Digicom II Digital Communications Part II

Based on book by Keshav

"An Engineering Approach to Computer Networking"

Jon Crowcroft

http://www.cl.cam.ac.uk/~jac22

Any Questions?

- …otherwise we can avoid all these 11am meetings☺

DC-II (=DC-I++) :- Contents

DCII Contents continued…

- Shared Media Networks (Ether/Radio etc) - some special problems
- Switched Networks - What does a switch do and how?
- Integrated Service Packet Networks for IP
- APIs to Quality of Service
- **Scheduling and Queue Management Algorithms for packet forwarding**
- What about routing with QoS
- The Big Picture for manageing traffic
- Economics, Policy and a little MPLS

The Telephone Network

An Engineering Approach to Computer Networking

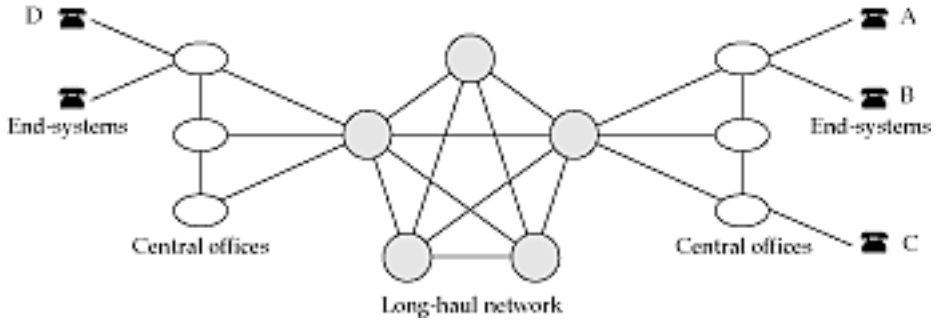
Is it a computer network?

- Specialized to carry voice
- Also carries
  - telemetry
  - video
  - fax
  - modem calls
- Internally, uses digital *samples*
- Switches and switch controllers are special purpose computers
- Principles in its design apply to more general computer networks

Concepts

- Single basic service: two-way voice
  - low end-to-end delay
  - guarantee that an accepted call will run to completion
- Endpoints connected by a *circuit*
  - like an electrical circuit
  - signals flow both ways (*full duplex*)
  - associated with bandwidth and buffer *resources*

The big picture



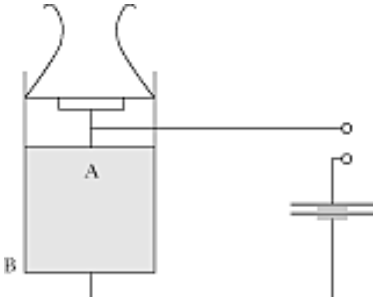
- Fully connected core
  - simple routing
  - telephone number is a hint about how to route a call
    - but not for 800/888/700/900 numbers
  - hierarchically allocated telephone number space

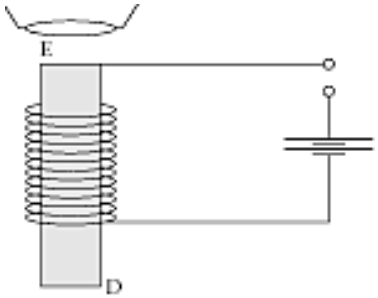
The pieces

1. End systems
2. Transmission
3. Switching
4. Signaling

1. End-systems

- Transducers
  - key to carrying voice on wires
- Dialer
- Ringer
- Switchhook





Sidetone

- Transmission circuit needs two wires
- And so does reception circuit
- => 4 wires from every central office to home
- Can we do better?
- Use *same* pair of wires for both transmission and reception
- Cancel out what is being said
- Ergonomics: leave in a little
  - *sidetone*
  - unavoidable

Echo

- Shared wires => received signal is also transmitted
- And not completely cancelled out!
- Leads to echo (why?)
- OK for short-distance calls
- For long distance calls, need to put in echo chancellors (why?)
- Expensive
- Lesson
  - keep end-to-end delays as short as possible

Dialing

- Pulse
  - sends a pulse per digit
  - collected by central office
- Tone
  - key press (feep) sends a pair of tones = digit
  - also called Dual Tone Mutifrequency (DTMF)

2. Transmission

- Link characteristics
  - information carrying capacity (bandwidth)
    - information sent as *symbols*
    - 1 symbol >= 1 bit
  - propagation delay
    - time for electromagnetic signal to reach other end
    - light travels at 0.7c in fiber ~8 microseconds/mile
    - NY to SF => 20 ms; NY to London => 27 ms
  - attenuation
    - degradation in signal quality with distance
    - long lines need regenerators
    - optical amplifiers are here

Transmission: Multiplexing

- *Trunks* between central offices carry hundreds of conversations
- Can't run thick bundles!
- Instead, send many calls on the same wire
  - *multiplexing*
- Analog multiplexing
  - bandlimit call to 3.4 KHz and frequency shift onto higher bandwidth trunk
  - obsolete
- Digital multiplexing
  - first convert voice to *samples*
  - 1 sample = 8 bits of voice
  - 8000 samples/sec => call = 64 Kbps

Transmission: Digital multiplexing

- How to choose a sample?
  - 256 *quantization levels*
    - logarithmically spaced (why?0
    - sample value = amplitude of nearest quantization level
  - two choices of levels (mu law and A law)
- Time division multiplexing
  - trunk carries bits at a faster bit rate than inputs
  - *n* input streams, each with a 1-byte buffer
  - output interleaves samples
  - need to serve all inputs in the time it takes one sample to arrive
  - => output runs *n* times faster than input
  - *overhead* bits mark end of *frame* (why?)

Transmission: Multiplexing

- Multiplexed trunks can be multiplexed further
- Need a standard! (why?)
- US/Japan standard is called *Digital Signaling* hierarchy (DS)

Digital Signal Number	Number of previous level circuits	Number of voice circuits	Bandwidth
DS0		1	64 Kbps
DS1	24	24	1.544Mbps
DS2	4	96	6.312 Mbps
DS3	7	672	44.736 Mbps

Transmission: Link technologies

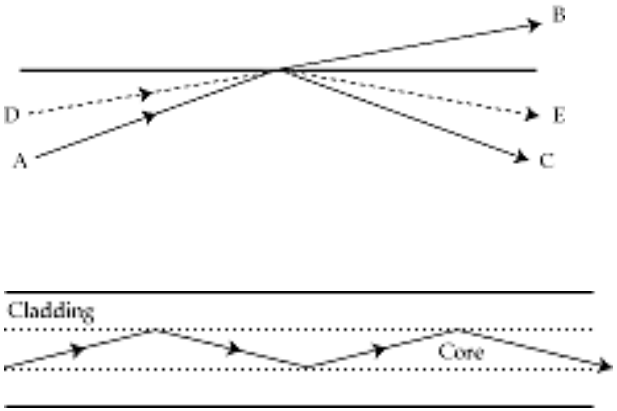
- Many in use today
  - twisted pair
  - coax cable
  - terrestrial microwave
  - satellite microwave
  - optical fiber
- Increasing amount of bandwidth and cost per foot
- Popular
  - fiber
  - satellite

The cost of a link

- Should you use the cheapest possible link?
- No!
- Cost is in installation, not in link itself
- Builders routinely install twisted pair (CAT 5), fiber, and coax to every room
- Even if only one of them used, still saves money
- Long distance
  - overprovision by up to ten times

Transmission: fiber optic links

- Wonderful stuff!
  - lots of capacity
  - nearly error free
  - very little attenuation
  - hard to tap
- A long thin strand of very pure glass



More on fibers

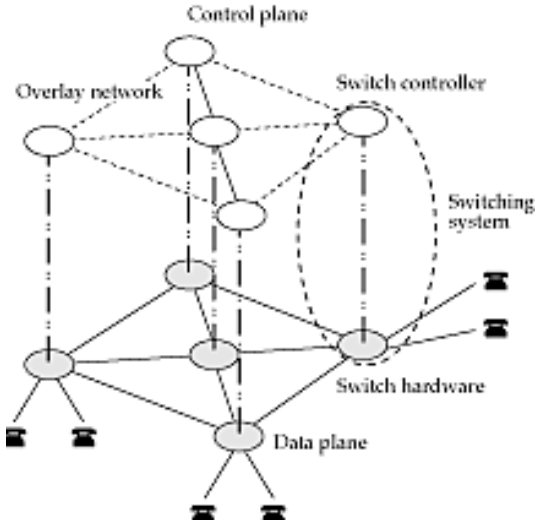
- Three types
  - step index (multimode)
  - graded index (multimode)
  - single mode
- Multimode
  - cheap
  - use LEDs
  - short distances (up to a few kilometers)
- Single mode
  - expensive
  - use lasers
  - long distances (up to hundreds of kilometers)

Transmission: satellites

- Long distances at high bandwidth
- Geosynchronous
  - 36,000 km in the sky
  - up-down propagation delay of 250 ms
  - bad for interactive communication
  - slots in space limited
- Nongeosynchronous (Low Earth Orbit)
  - appear to move in the sky
  - need more of them
  - handoff is complicated
  - e.g. Iridium

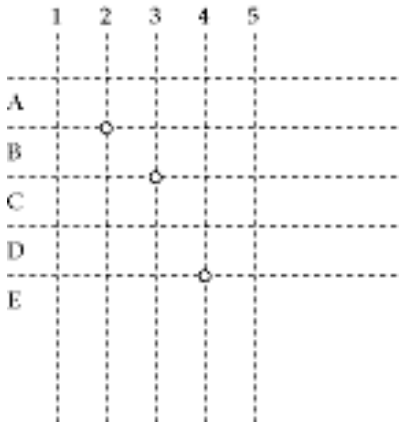
3. Switching

- Problem:
  - each user can potentially call any other user
  - can't have direct lines!
- Switches establish temporary *circuits*
- Switching systems come in two parts: switch and switch controller



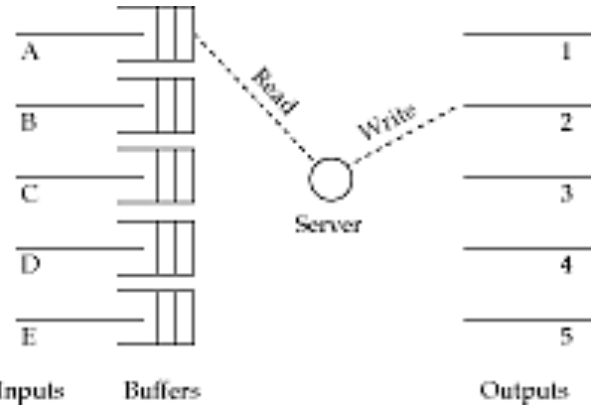
Switching: what does a switch do?

- Transfers data from an input to an output
  - many ports (up to200,000 simultaneous calls)`
  - need high speeds
- Some ways to switch:
  - *space division*
  - if inputs are multiplexed, need a *schedule* (why?)



Switching

- Another way to switch
  - *time division (time slot interchange* or *TSI)*
  - also needs a schedule (why?)



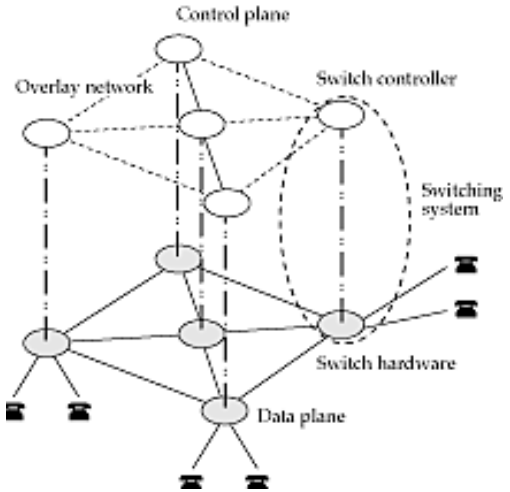
- To build larger switches we combine space and time division switching elements

4. Signaling

- Recall that a switching system has a switch and a switch controller
- Switch controller is in the *control* plane
  - does not touch voice samples
- Manages the network
  - call routing (collect *dialstring* and forward call)
  - alarms (ring bell at receiver)
  - billing
  - directory lookup (for 800/888 calls)

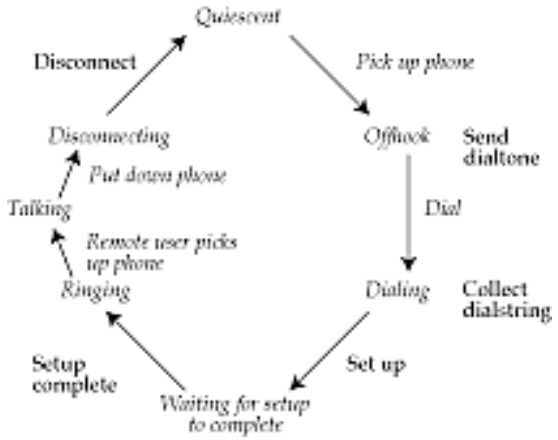
Signaling network

- Switch controllers are special purpose computers
- Linked by their own internal computer network
  - *Common Channel Interoffice Signaling (CCIS) network*
- Earlier design used *in-band* tones, but was severely hacked
- Also was very rigid (why?)
- Messages on CCIS conform to *Signaling System 7 (SS7)* spec.



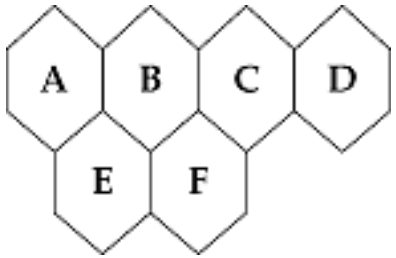
Signaling

- One of the main jobs of switch controller: keep track of *state* of every endpoint
- Key is *state transition diagram*



Cellular communication

- Mobile phone talks to a *base station* on a particular radio frequency
- Aren't enough frequencies to give each mobile a permanent frequency (like a wire)
- *Reuse*
  - temporal
    - if mobile is off, no frequency assigned to it
  - spatial
    - mobiles in non-adjacent *cells* can use the same frequency



Problems with cellular communication

- How to complete a call to a mobile?
  - need to *track* a mobile
  - on power on, mobile tells base of its ID and home
  - calls to home are forwarded to mobile over CCIS
- How to deal with a moving cell phone?
  - nearest base station changes
  - need to *hand off* existing call to new base station
  - a choice of several complicated protocols

Challenges for the telephone network

- Multimedia
  - simultaneously transmit voice/data/video over the network
  - people seem to want it
  - existing network can't handle it
    - bandwidth requirements
    - *burstiness* in traffic (TSI can't skip input)
    - change in statistical behavior
- Backward compatibility of new services
  - huge existing infrastructure
  - idiosyncrasies
- Regulation
  - stifles innovation

Challenges

- Competition
  - future telephone networks will no longer be monopolies
  - how to manage the transition?
- Inefficiencies in the system
  - an accumulation of cruft
  - special-purpose systems of the past
  - ‘legacy’ systems
  - need to change them without breaking the network

The Internet

An Engineering Approach to Computer Networking

My how you've grown!

- The Internet has doubled in size every year since 1969
- In 1996, 10 million computers joined the Internet
- By July 1997, 10 million more will join!
- Soon, everyone who has a phone is likely to also have an email account
  - already nearly true for Ithaca
  - PacTel telephone directories are planning to include email addresses in white pages

What does it look like?

- Loose collection of networks organized into a multilevel hierarchy
  - 10-100 machines connected to a *hub* or a *router*
    - service providers also provide direct dialup access
    - or over a wireless link
  - 10s of routers on a *department backbone*
  - 10s of department backbones connected to *campus backbone*
  - 10s of campus backbones connected to *regional service providers*
  - 100s of regional service providers connected by *national backbone*
  - 10s of national backbones connected by *international trunks*

Example of message routing

```
# traceroute henna.iitd.ernet.in
traceroute to henna.iitd.ernet.in (202.141.64.30), 30 hops max, 40 byte packets
 1  UPSON2-NP.CIT.CORNELL.EDU (128.84.154.1)  1 ms  1 ms  1 ms
 2  HOL1-MSS.CIT.CORNELL.EDU (132.236.230.189)  2 ms  3 ms  2 ms
 3  CORE1-MSS.CIT.CORNELL.EDU (128.253.222.1)  2 ms  2 ms  2 ms
 4  CORNELLNET1.CIT.CORNELL.EDU (132.236.100.10)  4 ms  3 ms  4 ms
 5  ny-ith-1-H1/0-T3.nysernet.net (169.130.61.9)  5 ms  5 ms  4 ms
 6  ny-ith-2-F0/0.nysernet.net (169.130.60.2)  4 ms  4 ms  3 ms
 7  ny-pen-1-H3/0-T3.nysernet.net (169.130.1.121)  21 ms  19 ms  16 ms
 8  sl-pen-21-F6/0/0.sprintlink.net (144.228.60.21)  16 ms  40 ms  36 ms
 9  core4-hssi5-0.WestOrange.mci.net (206.157.77.105)  20 ms  20 ms  24 ms
10  core2.WestOrange.mci.net (204.70.4.185)  21 ms  34 ms  26 ms
11  border7-fddi-0.WestOrange.mci.net (204.70.64.51)  21 ms  21 ms  21 ms
12  vsnl-poone-512k.WestOrange.mci.net (204.70.71.90)  623 ms  639 ms  621 ms
13  202.54.13.170 (202.54.13.170)  628 ms  629 ms  628 ms
14  144.16.60.2 (144.16.60.2)  1375 ms  1349 ms  1343 ms
15  henna.iitd.ernet.in (202.141.64.30)  1380 ms  1405 ms  1368 ms
```

Intranet, Internet, and Extranet

- Intranets are administered by a single entity
  - e.g. Cornell campus network
- Internet is administered by a coalition of entities
  - name services, backbone services, routing services etc.
- Extranet is a marketing term
  - refers to exterior customers who can access privileged Intranet services
  - e.g. Cornell could provide ‘extranet’ services to Ithaca college

What holds the Internet together?

- Addressing
  - how to refer to a machine on the Internet
- Routing
  - how to get there
- Internet Protocol (IP)
  - what to speak to be understood

Example: joining the Internet

- How can people talk to you?
  - get an IP address from your administrator
- How do you know where to send your data?
  - if you only have a single external connection, then no problem
  - otherwise, need to speak a routing protocol to decide next hop
- How to format data?
  - use the IP format so that intermediate routers can understand the destination address
- If you meet these criteria--you're on the Internet!
- Decentralized, distributed, and chaotic
  - but it scales (why?)

What lies at the heart?

- Two key technical innovations
  - packets
  - store and forward

Packets

- Self-descriptive data
  - packet = data + metadata (header)
- Packet vs. sample
  - samples are not self descriptive
  - to forward a sample, we have to know *where* it came from and *when*
  - can't store it!
  - hard to handle bursts of data

Store and forward

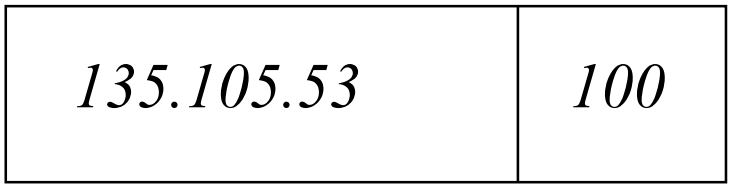
- Metadata allows us to forward packets when we want
- E.g. letters at a post office headed for main post office
  - address labels allow us to forward them in batches
- Efficient use of critical resources
- Three problems
  - hard to control delay within network
  - switches need memory for buffers
  - convergence of flows can lead to congestion

Key features of the Internet

- Addressing
- Routing
- Endpoint control

Addressing

- Internet addresses are called IP addresses
- Refer to a *host interface*: need one IP address per interface
- Addresses are structured as a two-part hierarchy
  - network number
  - host number



An interesting problem

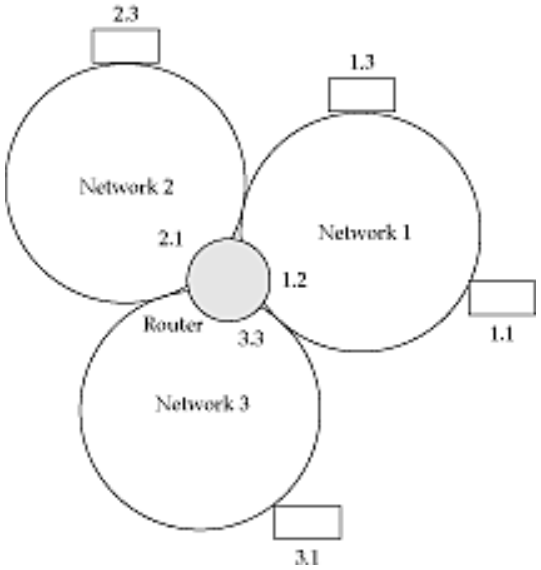
- How many bits to assign to host number and how many to network number?
- If many networks, each with a few hosts, then more bits to network number
- And *vice versa*
- But designer's couldn't predict the future
- Decided three sets of partitions of bits
  - class A: 8 bits network, 24 bits host
  - class B: 16 bits each
  - class C: 24 bits network, 8 bits host

Addressing (contd.)

- To distinguish among them
  - use leading bit
  - first bit = 0=> class A
  - first bits 10 => class B
  - first bits 110 => class C
  - (what class address is 135.104.53.100?)
- Problem
  - if you want more than 256 hosts in your network, need to get a class B, which allows 64K hosts => wasted address space
- Solution
  - associate *every* address with a *mask* that indicates partition point
  - *CIDR*

Routing

- How to get to a destination given its IP address?
- We need to know the next hop to reach a particular network number
  - this is called a *routing table*
  - computing routing tables is non-trivial
- Simplified example



Default routes

- Strictly speaking, need next hop information for every network in the Internet
  - > 80,000 now
- Instead, keep detailed routes only for local neighborhood
- For unknown destinations, use a *default* router
- Reduces size of routing tables at the expense of non-optimal paths

Endpoint control

- Key design philosophy
  - do as much as possible at the endpoint
  - dumb network
  - exactly the opposite philosophy of telephone network
- Layer above IP compensates for network defects
  - Transmission Control Protocol (TCP)
- Can run over any available link technology
    - but no quality of service
    - modification to TCP requires a change at every endpoint
    - (how does this differ from telephone network?)

Challenges

- IP address space shortage
  - because of free distribution of inefficient Class B addresses
  - decentralized control => hard to recover addresses, once handed out
- Decentralization
  - allows scaling, but makes *reliability* next to impossible
  - cannot guarantee that a route exists, much less bandwidth or buffer resources
  - single points of failure can cause a major disaster
    - and there is no control over who can join!
  - hard to guarantee security
    - end-to-end encryption is a partial solution
    - who manages keys?

Challenges (contd.)

- Decentralization (contd.)
  - no uniform solution for accounting and billing
    - can't even reliably identify individual users
  - no equivalent of white or yellow pages
    - hard to reliably discover a user's email address
  - nonoptimal routing
    - each administrative makes a locally optimal decision

Challenges (contd).

- Multimedia
  - requires network to support quality of service of some sort
    - hard to integrate into current architecture
    - store-and-forward => shared buffers => traffic interaction => hard to provide service quality
  - requires endpoint to signal to the network what it wants
    - but Internet does not have a simple way to identify streams of packets
    - nor are are routers required to cooperate in providing quality
    - and what about pricing!

ATM Networks

An Engineering Approach to Computer Networking

Why ATM networks?

- Different information types require different qualities of service from the network
  - stock quotes vs. USENET
- Telephone networks support a single quality of service
  - and is expensive to boot
- Internet supports no quality of service
  - but is flexible and cheap
- ATM networks are meant to support a range of service qualities at a reasonable cost
  - potentially can subsume both the telephone network and the Internet

Design goals

- Providing end-to-end quality of service
- High bandwidth
- Scalability
- Manageability
- Cost-effective

How far along are we?

- Basic architecture has been defined
- But delays have resulting in ceding desktop to IP
- Also, little experience in traffic specification, multicast, and fault tolerance
- We may never see end-to-end ATM
  - but its ideas continue to powerfully influence design of next-generation Internet
  - Internet technology + ATM philosophy
- Note--two standardization bodies
  - ATM Forum
  - International Telecommunications Union-Telecommunications Standardization Sector (ITU-T)

Concepts

1. Virtual circuits
2. Fixed-size packets (*cells*)
3. Small packet size
4. Statistical multiplexing
5. Integrated services

Together

can carry *multiple* types of traffic

with end-to-end quality of service

1. Virtual circuits

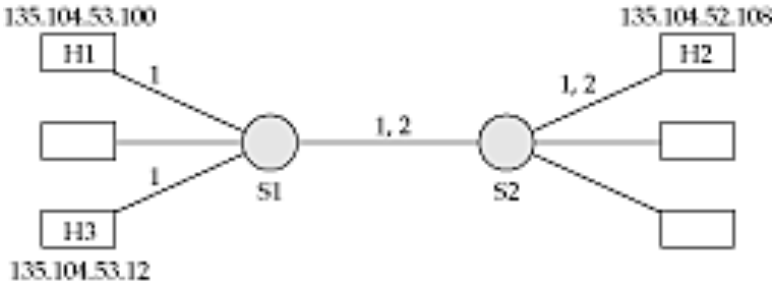
- Some background first
- Telephone network operates in *synchronous transmission mode*
  - the destination of a sample depends on where it comes from, and when it came
  - example--shared leased link
- Problems with STM
  - idle users consume bandwidth
  - links are shared with a fixed cyclical schedule => quantization of link capacity
    - can't 'dial' bandwidth

Virtual circuits (contd.)

- STM is easy to overcome
  - use *packets*
  - metadata indicates destination =>arbitrary schedule and no wasted bandwidth
- Two ways to use packets
  - carry entire destination address in header
  - carry only an identifier

Virtual circuits (contd.)

- Ids save on header space
- But need to be pre-established
- We also need to switch Ids at intermediate points (why?)
- Need *translation table* and *connection setup*



Features of virtual circuits

- All packets must follow the same path (why?)
- Switches store per-VCI state
  - can store QoS information
- Signaling => separation of *data* and *control*
- Virtual circuits do not automatically guarantee reliability
- Small Ids can be looked up quickly in hardware
  - harder to do this with IP addresses
- Setup must precede data transfer
  - delays short messages
- Switched vs. Permanent virtual circuits

More features

- Ways to reduce setup latency
  - preallocate a range of VCIs along a path
    - *Virtual Path*
  - send data cell along with setup packet
  - dedicate a VCI to carry datagrams, reassembled at each hop

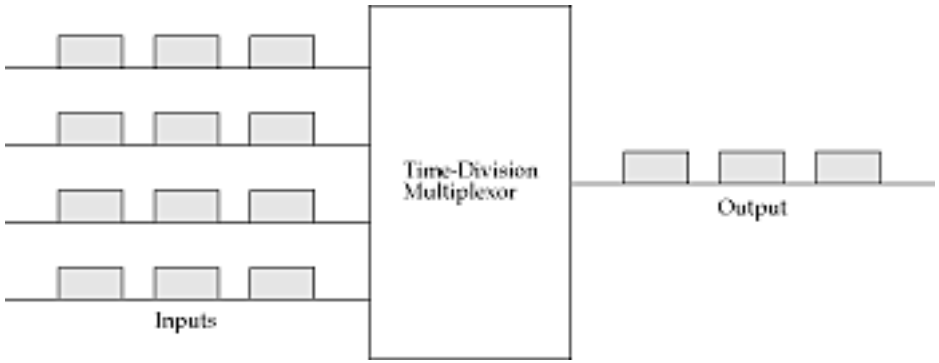
2. Fixed-size packets

- Pros
  - Simpler buffer hardware
    - packet arrival and departure requires us to manage fixed buffer sizes
  - Simpler line scheduling
    - each cell takes a constant chunk of bandwidth to transmit
  - Easier to build large parallel packet switches
- Cons
  - overhead for sending small amounts of data
  - segmentation and reassembly cost
  - last unfilled cell after segmentation wastes bandwidth

3. Small packet size

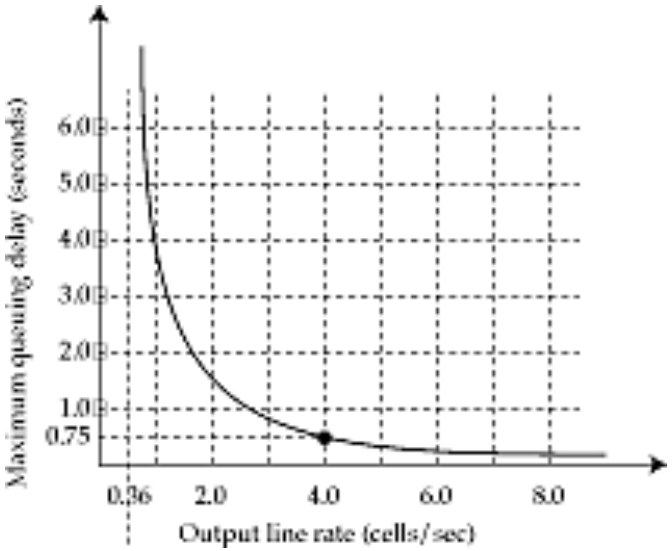
- At 8KHz, each byte is 125 microseconds
- The smaller the cell, the less an endpoint has to wait to fill it
  - *packetization delay*
- The smaller the packet, the larger the header overhead
- Standards body balanced the two to prescribe 48 bytes + 5 byte header = 53 bytes
  - => maximal efficiency of 90.57%

4. Statistical multiplexing



- Suppose cells arrive in bursts
  - each burst has 10 cells evenly spaced 1 second apart
  - gap between bursts = 100 seconds
- What should be service rate of output line?

Statistical multiplexing



- We can trade off worst-case delay against speed of output trunk
- SMG = sum of peak input/output rate
- Whenever long term average rate differs from peak, we can trade off service rate for delay
  - key to building packet-switched networks with QoS

5. Integrated service

- Traditionally, voice, video, and data traffic on separate networks
- Integration
  - easier to manage
  - innovative new services
- How do ATM networks allow for integrated service?
  - lots of bandwidth: hardware-oriented switching
  - support for different traffic types
    - signaling
    - admission control
    - easier scheduling
    - resource reservation

Challenges

- Quality of service
  - defined, but not used!
  - still needs research
- Scaling
  - little experience
- Competition from other LAN technologies
  - Fast Ethernet
  - FDDI
- Standardization
  - political
  - slow

Challenges

- IP
  - a vast, fast-growing, non-ATM infrastructure
  - interoperation is a pain in the neck, because of fundamentally different design philosophies
    - connectionless vs. connection-oriented
    - resource reservation vs. best-effort
    - different ways of expressing QoS requirements
    - routing protocols differ

System Design

An Engineering Approach to Computer Networking

What is system design?

- A computer network provides computation, storage and transmission resources
- System design is the art and science of putting together these resources into a harmonious whole
- Extract the most from what you have

Goal

- In any system, some resources are more freely available than others
  - high-end PC connected to Internet by a 28.8 modem
  - *constrained* resource is link bandwidth
  - PC CPU and and memory are *unconstrained*
- Maximize a set of performance metrics given a set of resource constraints
- Explicitly identifying constraints and metrics helps in designing efficient systems
- Example
  - maximize reliability and MPG for a car that costs less than $10,000 to manufacture

System design in real life

- Can’t always quantify and control all aspects of a system
- Criteria such as scalability, modularity, extensibility, and elegance are important, but unquantifiable
- Rapid technological change can add or remove resource constraints (example?)
  - an ideal design is ‘future proof’
- Market conditions may dictate changes to design halfway through the process
- International standards, which themselves change, also impose constraints
- Nevertheless, still possible to identify some principles

Some common resources

- Most resources are a combination of
  - time
  - space
  - computation
  - money
  - labor

Time

- Shows up in many constraints
  - deadline for task completion
  - time to market
  - mean time between failures
- Metrics
  - *response time*: mean time to complete a task
  - *throughput*: number of tasks completed per unit time
  - *degree of parallelism* = response time * throughput
    - 20 tasks complete in 10 seconds, and each task takes 3 seconds
    - => degree of parallelism = 3 * 20/10 = 6

Space

- Shows up as
  - limit to available memory (kilobytes)
  - bandwidth (kilobits)
    - 1 kilobit/s = 1000 bits/sec, but 1 kilobyte/s = 1024 bits/sec!

Computation

- Amount of processing that can be done in unit time
- Can increase computing power by
  - using more processors
  - waiting for a while!

Money

- Constrains
  - what components can be used
  - what price users are willing to pay for a service
  - the number of engineers available to complete a task

Labor

- Human effort required to design and build a system
- Constrains what can be done, and how fast

Social constraints

- Standards
  - force design to conform to requirements that may or may not make sense
  - underspecified standard can faulty and non-interoperable implementations
- Market requirements
  - products may need to be backwards compatible
  - may need to use a particular operating system
  - example
    - GUI-centric design

Scaling

- A design constraint, rather than a resource constraint
- Can use any centralized elements in the design
  - forces the use of complicated distributed algorithms
- Hard to measure
  - but necessary for success

Common design techniques

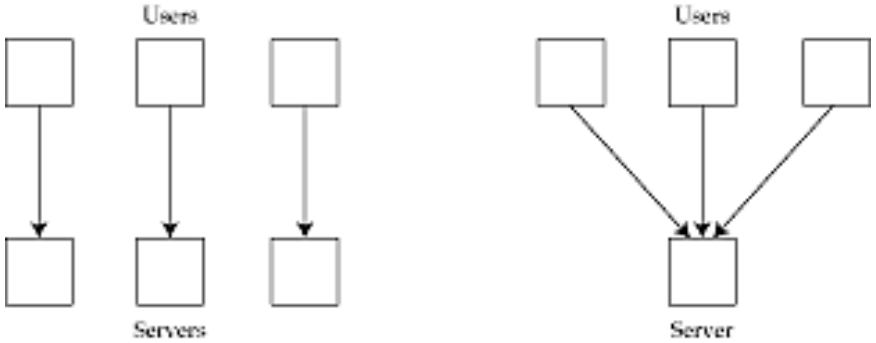
- Key concept: *bottleneck*
  - the most constrained element in a system
- System performance improves by removing bottleneck
  - but creates new bottlenecks
- In a *balanced* system, all resources are simultaneously bottlenecked
  - this is optimal
  - but nearly impossible to achieve
  - in practice, bottlenecks move from one part of the system to another
  - example: Ford Model T

Top level goal

- Use unconstrained resources to alleviate bottleneck
- How to do this?
- Several standard techniques allow us to trade off one resource for another

Multiplexing

- Another word for sharing
- Trades time and space for money
- Users see an increased response time, and take up space when waiting, but the system costs less
  - economies of scale



Multiplexing (contd.)

- Examples
  - multiplexed links
  - shared memory
- Another way to look at a shared resource
  - *unshared virtual resource*
- *Server* controls access to the shared resource
  - uses a *schedule* to resolve contention
  - choice of scheduling critical in proving quality of service guarantees

Statistical multiplexing

- Suppose resource has capacity C
- Shared by N identical tasks
- Each task requires capacity c
- If Nc <= C, then the resource is underloaded
- If at most 10% of tasks active, then C >= Nc/10 is enough
  - we have used statistical knowledge of users to reduce system cost
  - this is *statistical multiplexing gain*

Statistical multiplexing (contd.)

- Two types: spatial and temporal
- Spatial
  - we expect only a fraction of tasks to be simultaneously active
- Temporal
  - we expect a task to be active only part of the time
    - e.g silence periods during a voice call

Example of statistical multiplexing gain

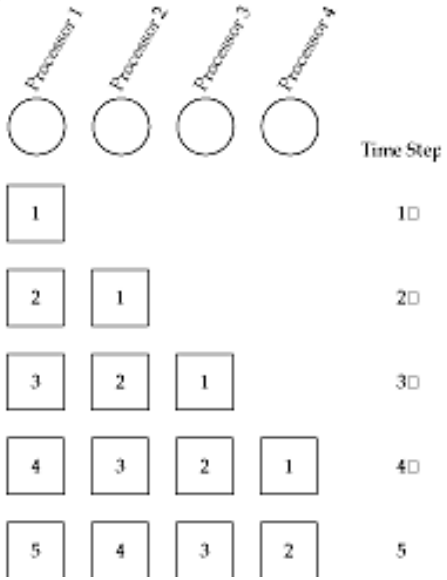
- Consider a 100 room hotel
- How many external phone lines does it need?
  - each line costs money to install and rent
  - tradeoff
- What if a voice call is active only 40% of the time?
  - can get both spatial and temporal statistical multiplexing gain
  - but only in a packet-switched network (why?)
- Remember
  - to get SMG, we need good statistics!
  - if statistics are incorrect or change over time, we're in trouble
  - example: road system

Pipelining

- Suppose you wanted to complete a task in less time
- Could you use more processors to do so?
- Yes, if you can break up the task into *independent* subtasks
  - such as downloading images into a browser
  - optimal if all subtasks take the same time
- What if subtasks are dependent?
  - for instance, a subtask may not begin execution before another ends
  - such as in cooking
- Then, having more processors doesn't always help (example?)

Pipelining (contd.)

- Special case of *serially dependent* subtasks
  - a subtask depends only on previous one in execution chain
- Can use a *pipeline*
  - think of an assembly



Pipelining (contd.)

- What is the best decomposition?
- If sum of times taken by all stages = R
- Slowest stage takes time S
- Throughput = 1/S
- Response time = R
- Degree of parallelism = R/S
- Maximize parallelism when R/S = N, so that S = R/N => equal stages
  - *balanced pipeline*

Batching

- Group tasks together to amortize overhead
- Only works when overhead for N tasks < N time overhead for one task (i.e. *nonlinear*)
- Also, time taken to accumulate a batch shouldn't be too long
- We're trading off reduced overhead for a longer worst case response time and increased throughput

Exploiting locality

- If the system accessed some data at a given time, it is likely that it will access the same or ‘nearby’ data ‘soon’
- Nearby => spatial
- Soon => temporal
- Both may coexist
- Exploit it if you can
  - caching
    - get the speed of RAM and the capacity of disk

Optimizing the common case

- 80/20 rule
  - 80% of the time is spent in 20% of the code
- Optimize the 20% that counts
  - need to measure first!
  - RISC
- How much does it help?
  - Amdahl's law
  - Execution time after improvement = (execution affected by improvement / amount of improvement) + execution unaffected
  - beyond a point, speeding up the common case doesn't help

Hierarchy

- Recursive decomposition of a system into smaller pieces that depend only on parent for proper execution
- No single point of control
- Highly scaleable
- Leaf-to-leaf communication can be expensive
  - shortcuts help

Binding and indirection

- Abstraction is good
  - allows generality of description
  - e.g. mail aliases
- Binding: translation from an abstraction to an instance
- If translation table is stored in a well known place, we can bind automatically
  - indirection
- Examples
  - mail alias file
  - page table
  - telephone numbers in a cellular system

Virtualization

- A combination of indirection and multiplexing
- Refer to a virtual resource that gets matched to an instance at run time
- Build system as if real resource were available
  - virtual memory
  - virtual modem
  - Santa Claus
- Can cleanly and dynamically reconfigure system

Randomization

- Allows us to break a tie fairly
- A powerful tool
- Examples
  - resolving contention in a broadcast medium
  - choosing multicast timeouts

Soft state

- State: memory in the system that influences future behavior
  - for instance, VCI translation table
- State is created in many different ways
  - signaling
  - network management
  - routing
- How to delete it?
- Soft state => delete on a timer
- If you want to keep it, refresh
- Automatically cleans up after a failure
  - but increases bandwidth requirement

Exchanging state explicitly

- Network elements often need to exchange state
- Can do this implicitly or explicitly
- Where possible, use explicit state exchange

Hysteresis

- Suppose system changes state depending on whether a variable is above or below a threshold
- Problem if variable fluctuates near threshold
  - rapid fluctuations in system state
- Use state-dependent threshold, or *hysteresis*

Separating data and control

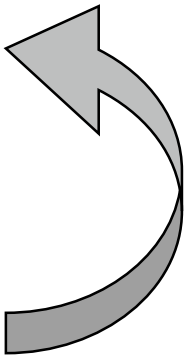
- Divide actions that happen once per data transfer from actions that happen once per packet
  - Data path and control path
- Can increase throughput by minimizing actions in data path
- Example
  - connection-oriented networks
- On the other hand, keeping control information in data element has its advantages
  - per-packet QoS

Extensibility

- Always a good idea to leave hooks that allow for future growth
- Examples
  - Version field in header
  - Modem negotiation

Performance analysis and tuning

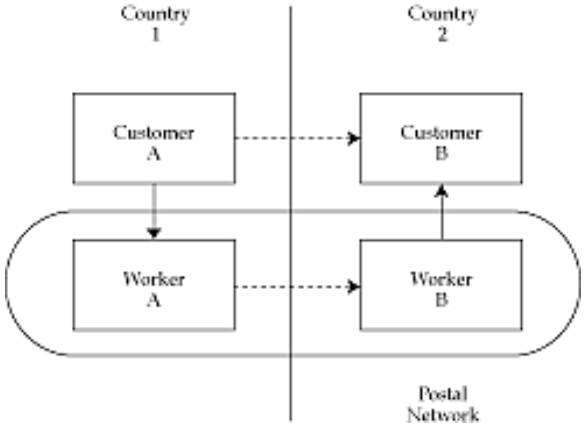
- Use the techniques discussed to tune existing systems
- Steps
  - measure
  - characterize workload
  - build a system model
  - analyze
  - implement



Protocol Layering

An Engineering Approach to Computer Networking

Peer entities



- Customer A and B are *peers*
- Postal worker A and B are *peers*

Protocols

- A *protocol* is a set of rules and formats that govern the communication between communicating peers
  - set of valid messages
  - meaning of each message
- A protocol is necessary for any function that requires cooperation between peers

Example

- Exchange a file over a network that corrupts packets
  - but doesn't lose or reorder them
- A simple protocol
  - send file as a series of packets
  - send a *checksum*
  - receiver sends OK or not-OK message
  - sender waits for OK message
  - if no response, resends entire file
- Problems
  - single bit corruption requires retransmission of entire file
  - what if link goes down?
  - what if not-OK message itself is corrupted?

What does a protocol tell us?

- *Syntax* of a message
  - what fields does it contain?
  - in what format?
- *Semantics* of a message
  - what does a message mean?
  - for example, not-OK message means receiver got a corrupted file
- *Actions* to take on receipt of a message
  - for example, on receiving not-OK message, retransmit the entire file

Another way to view a protocol

- As providing a *service*
- The example protocol provides *reliable file transfer service*
- Peer entities use a protocol to provide a service to a higher-level peer entity
  - for example, postal workers use a protocol to present customers with the abstraction of an *unreliable letter transfer* service

Protocol layering

- A network that provides many services needs many protocols
- Turns out that some services are independent
- But others depend on each other
- Protocol A may use protocol B as a *step* in its execution
  - for example, packet transfer is one step in the execution of the example reliable file transfer protocol
- This form of dependency is called *layering*
  - reliable file transfer is *layered* above packet transfer protocol
  - like a subroutine

Some terminology

- *Service access point (SAP)*
  - interface between an upper layer and a lower layer
- *Protocol data units (PDUs)*
  - packets exchanged between peer entities
- *Service data units (SDUs)*
  - packets handed to a layer by an upper layer
- PDU = SDU + optional header or trailer
- Example
  - letter transfer service
  - protocol data unit between customers = letter
  - service data unit for postal service = letter
  - protocol data unit = mailbag (aggregation of letters)
  - (what is the SDU header?)

Protocol stack

- A set of protocol layers
- Each layer uses the layer below and provides a service to the layer above
- Key idea
  - once we define a service provided by a layer, we need know nothing more about the details of *how* the layer actually implements the service
  - information hiding
  - decouples changes

The importance of being layered

- Breaks up a complex problem into smaller manageable pieces
  - can compose simple service to provide complex ones
  - for example, WWW (HTTP) is Java layered over TCP over IP (and uses DNS, ARP, DHCP, RIP, OSPF, BGP, PPP, ICMP)
- Abstraction of implementation details
  - separation of implementation and specification
  - can change implementation as long as service interface is maintained
- Can reuse functionality
  - upper layers can share lower layer functionality
  - example: WinSock on Microsoft Windows

Problems with layering

- Layering hides information
  - if it didn't then changes to one layer could require changes everywhere
    - *layering violation*
- But sometimes hidden information can be used to improve performance
  - for example, flow control protocol may think packet loss is always because of network congestion
  - if it is, instead, due to a lossy link, the flow control breaks
  - this is because we hid information about reason of packet loss from flow control protocol

Layering

- There is a tension between information-hiding (abstraction) and achieving good performance
- Art of protocol design is to leak enough information to allow good performance
  - but not so much that small changes in one layer need changes to other layers

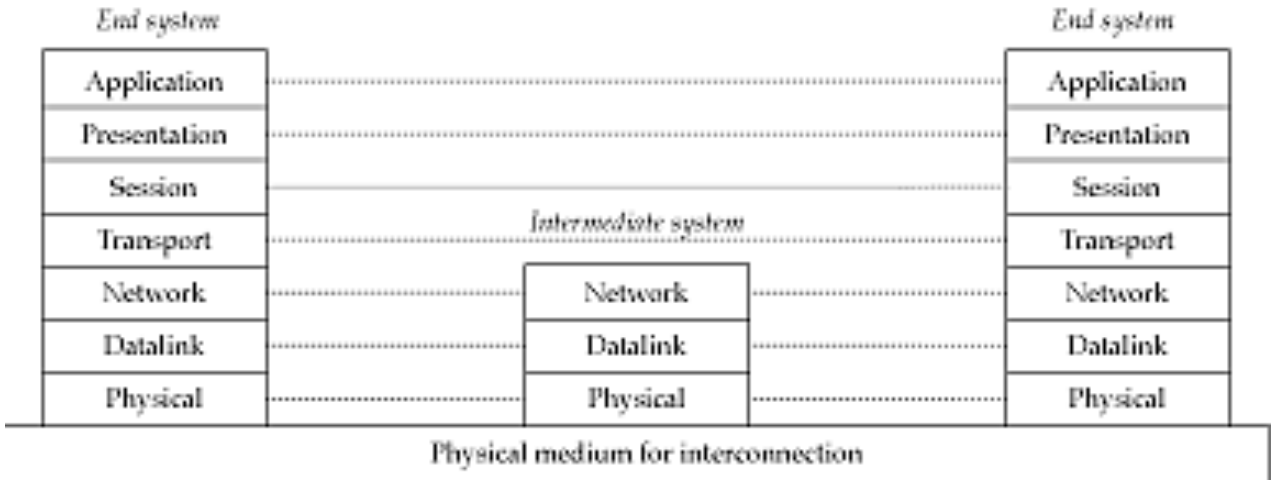
ISO OSI reference model

- A set of protocols is *open* if
  - protocol details are publicly available
  - changes are managed by an organization whose membership and transactions are open to the public
- A system that implements open protocols is called an *open system*
- International Organization for Standards (ISO) prescribes a standard to connect open systems
  - *open system interconnect (OSI)*
- Has greatly influenced thinking on protocol stacks

ISO OSI

- *Reference model*
  - formally defines what is meant by a layer, a service etc.
- *Service architecture*
  - describes the services provided by each layer and the service access point
- *Protocol architecture*
  - set of protocols that implement the service architecture
  - compliant service architectures may still use non-compliant protocol architectures

The seven layers



Physical layer

- Moves bits between physically connected end-systems
- Standard prescribes
  - coding scheme to represent a bit
  - shapes and sizes of connectors
  - bit-level synchronization
- Postal network
  - technology for moving letters from one point to another (trains, planes, vans, bicycles, ships…)
- Internet
  - technology to move bits on a wire, wireless link, satellite channel etc.

Datalink layer

- Introduces the notion of a *frame*
  - set of bits that belong together
- *Idle* markers tell us that a link is not carrying a frame
- *Begin* and *end* markers delimit a frame
- On a broadcast link (such as Ethernet)
  - end-system must receive only bits meant for it
  - need datalink-layer address
  - also need to decide who gets to speak next
  - these functions are provided by *Medium Access sublayer (MAC)*
- Some data links also retransmit corrupted packets and pace the rate at which frames are placed on a link
  - part of *logical link control sublayer*
  - layered over MAC sublayer

Datalink layer (contd.)

- Datalink layer protocols are the first layer of software
- Very dependent on underlying physical link propeties
- Usually bundle both physical and datalink layer on *host adaptor card*
  - example: Ethernet
- Postal service
  - mail bag ‘frames’ letters
- Internet
  - a variety of datalink layer protocols
  - most common is Ethernet
  - others are FDDI, SONET, HDLC

Network layer

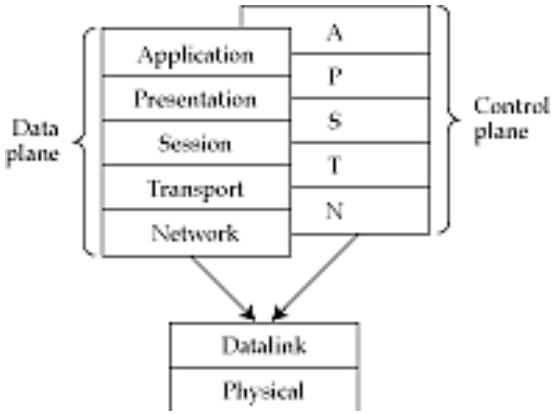
- Logically concatenates a set of links to form the abstraction of an end-to-end link
- Allows an end-system to communicate with any other end-system by computing a route between them
- Hides idiosyncrasies of datalink layer
- Provides unique network-wide addresses
- Found both in end-systems and in intermediate systems
- At end-systems primarily hides details of datalink layer
  - segmentation and reassembly
  - error detection

Network layer (contd.)

- At intermediate systems
  - participates in routing protocol to create routing tables
  - responsigle for forwarding packets
  - scheduling the transmission order of packets
  - choosing which packets to drop

Two types of network layers

- In datagram networks
  - provides both routing and data forwarding
- In connection-oriented network
  - we distinguish between data plane and control plane
  - data plane only forwards and schedules data (touches every byte)
  - control plane responsible for routing, call-establishment, call-teardown (doesn't touch data bytes)



Network layer

- Postal network
  - set up internal routing tables
  - forward letters from source to destination
  - static routing
  - multiple qualities of service
- Internet
  - network layer is provided by Internet Protocol
  - found in all end-systems and intermediate systems
  - provides abstraction of end-to-end link
  - segmentation and reassembly
  - packet-forwarding, routing, scheduling
  - unique IP addresses
  - can be layered over anything, but only best-effort service

Transport layer

- Network provides a ‘raw’ end-to-end service
- Transport layer creates the abstraction of an *error-controlled*, *flow-controlled* and *multiplexed* end-to-end link
- Error control
  - message will reach destination despite packet loss, corruption and duplication
  - retransmit lost packets; detect, discard, and retransmit corrupted packets; detect and discard duplicated packets
- Flow control
  - match transmission rat to rate currently sustainable on the path to destination, and at the destination itself

Transport layer (contd.)

- Multiplexes multiple applications to the same end-to-end connection
  - adds an application-specific identifier (*port number)* so that receiving end-system can hand in incoming packet to the correct application
- Some transport layers provide fewer services
  - e.g. simple error detection, no flow control, and no retransmission
  - *lightweight transport layer*

Transport layer (contd.)

- Postal system
  - doesn't have a transport layer
  - implemented, if at all, by customers
  - detect lost letters (how?) and retransmit them
- Internet
  - two popular protocols are TCP and UDP
  - TCP provides error control, flow control, multiplexing
  - UDP provides only multiplexing

Session layer

- Not common
- Provides *full-duplex service, expedited data delivery,* and *session synchronization*
- Duplex
  - if transport layer is simplex, concatenates two transport endpoints togeter
- Expedited data delivery
  - allows some messages to skip ahead in end-system queues, by using a separate low-delay transport layer endpoint
- Synchronization
  - allows users to place marks in data stream and to roll back to a prespecified mark

Example

- Postal network
  - suppose a company has separate shipping and receiving clerks
  - chief clerk can manage both to provide abstraction of a duplex service
  - chief clerk may also send some messages using a courier (expedited service)
  - chief clerk can arrange to have a set of messages either delivered all at once, or not at all
- Internet
  - doesn’t have a standard session layer

Presentation layer

- Unlike other layers which deal with *headers* presentation layer touches the application data
- Hides data representation differences between applications
  - e.g. *endian-ness*
- Can also encrypt data
- Usually *ad hoc*
- Postal network
  - translator translates contents before giving it to chief clerk
- Internet
  - no standard presentation layer
  - only defines network byte order for 2- and 4-byte integers

Application layer

- The set of applications that use the network
- Doesn't provide services to any other layer
- Postal network
  - the person who uses the postal system
  - suppose manager wants to send a set of recall letters
  - translator translates letters going abroad
  - chief clerk sends some priority mail, and some by regular mail
  - mail clerk sends a message, retransmits if not acked
  - postal system computes a route and forwards the letters
  - datalink layer: letters carried by planes, trains, automobiles
  - physical layer: the letter itself

Layering

- We have broken a complex problem into smaller, simpler pieces
- Provides the application with *sophisticated* services
- Each layer provides a clean abstraction to the layer above

Why seven layers?

- Need a top and a bottom -- 2
- Need to hide physical link, so need datalink -- 3
- Need both end-to-end and hop-by-hop actions; so need at least the network and transport layers -- 5
- Session and presentation layers are not so important, and are often ignored
- So, we need at least 5, and 7 seems to be excessive
- Note that we can place functions in different layers

Protocol Implementation

An Engineering Approach to Computer Networking

Protocol implementation

- Depends on *structure* and *environment*
- Structure
  - *partitioning* of functionality between user and kernel
  - separation of layer processing (*interface*)
- Environment
  - data copy cost
  - interrupt overhead
  - context switch time
  - latency in accessing memory
  - cache effects

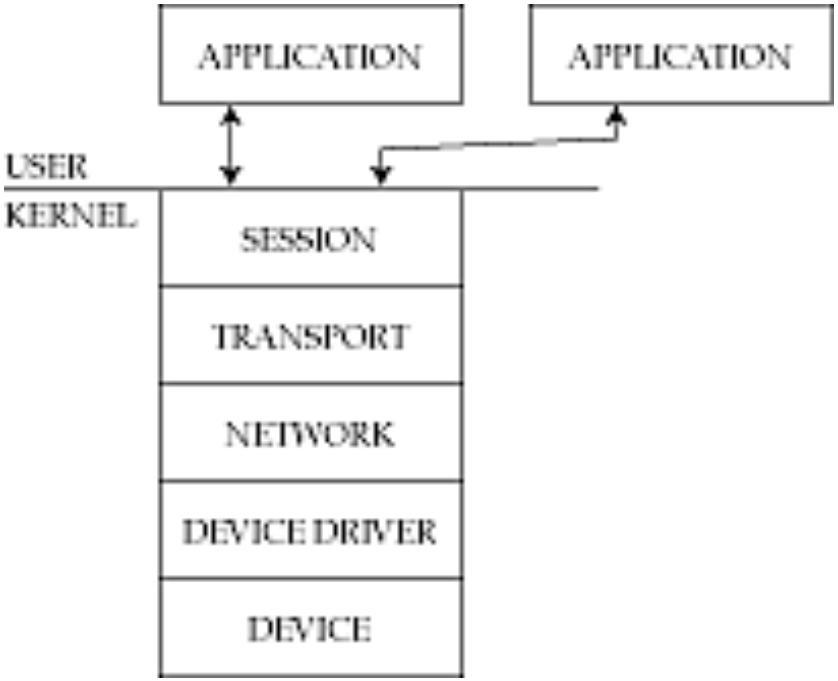
Partitioning strategies

- How much to put in user space, and how much in kernel space?
  - tradeoff between
    - software engineering
    - customizability
    - security
    - performance
- Monolithic in kernel space
- Monolithic in user space
- Per-process in user space

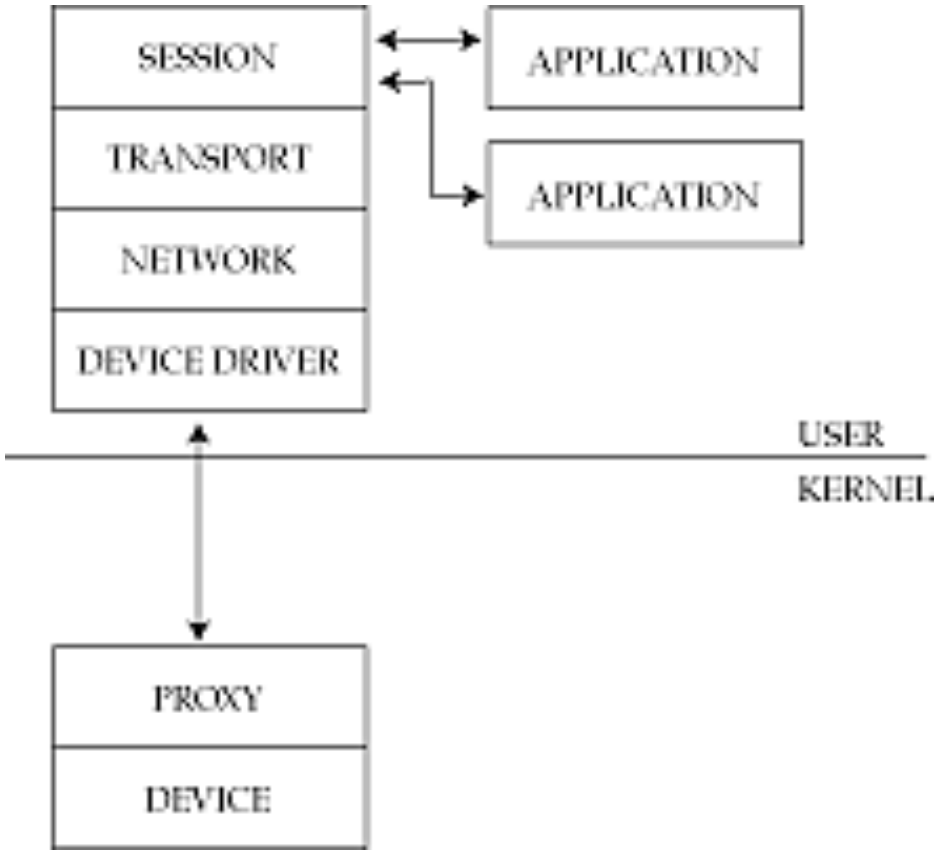
Interface strategies

- Single-context
- Tasks
- Upcalls

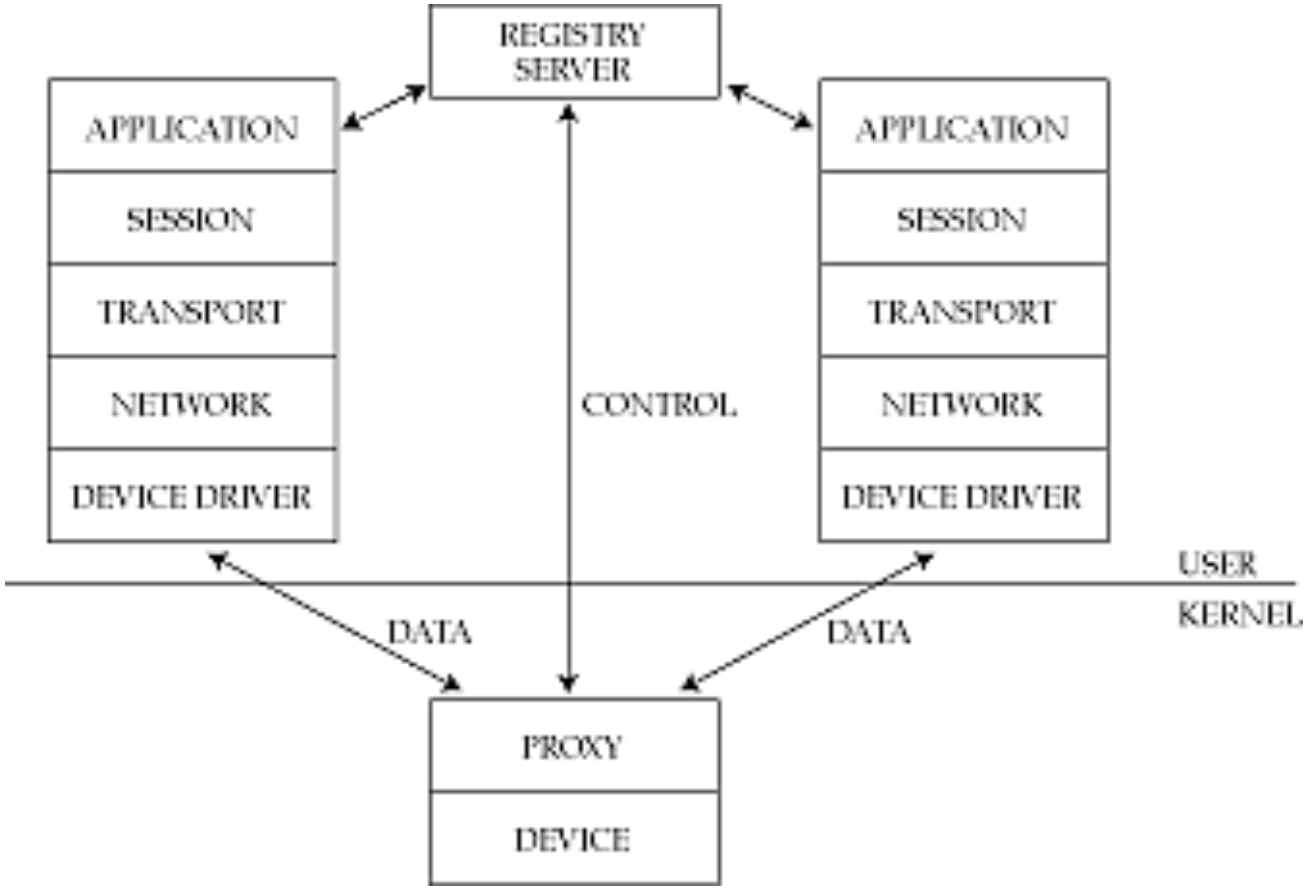
Monolithic in kernel



Monolithic in user space



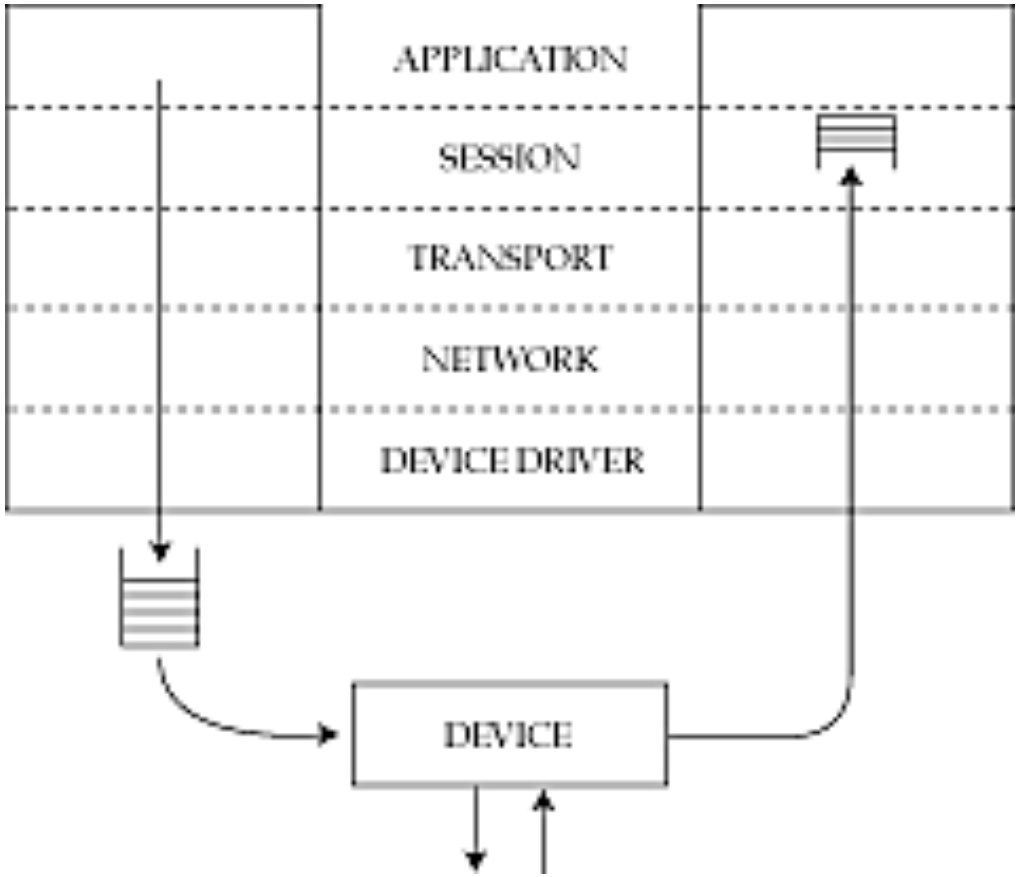
Per-process in user space



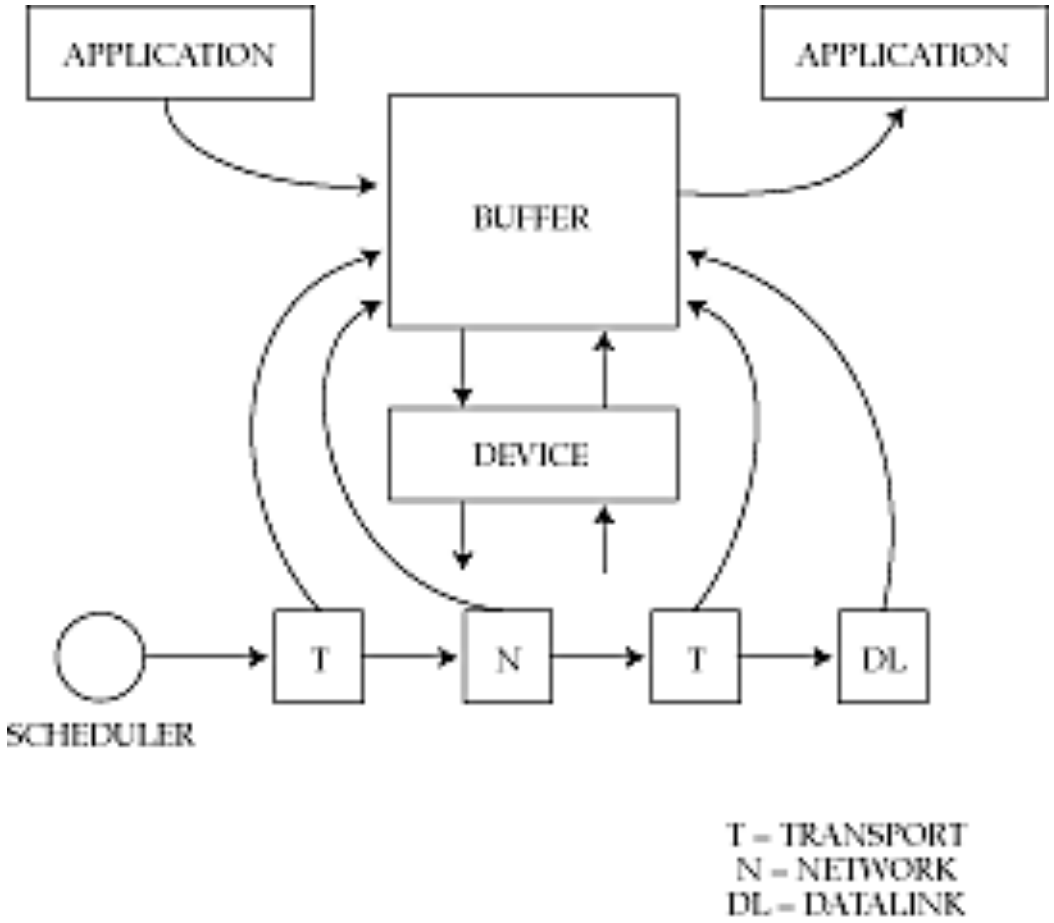
Interfaces

- Single-context
- Tasks
- Upcalls

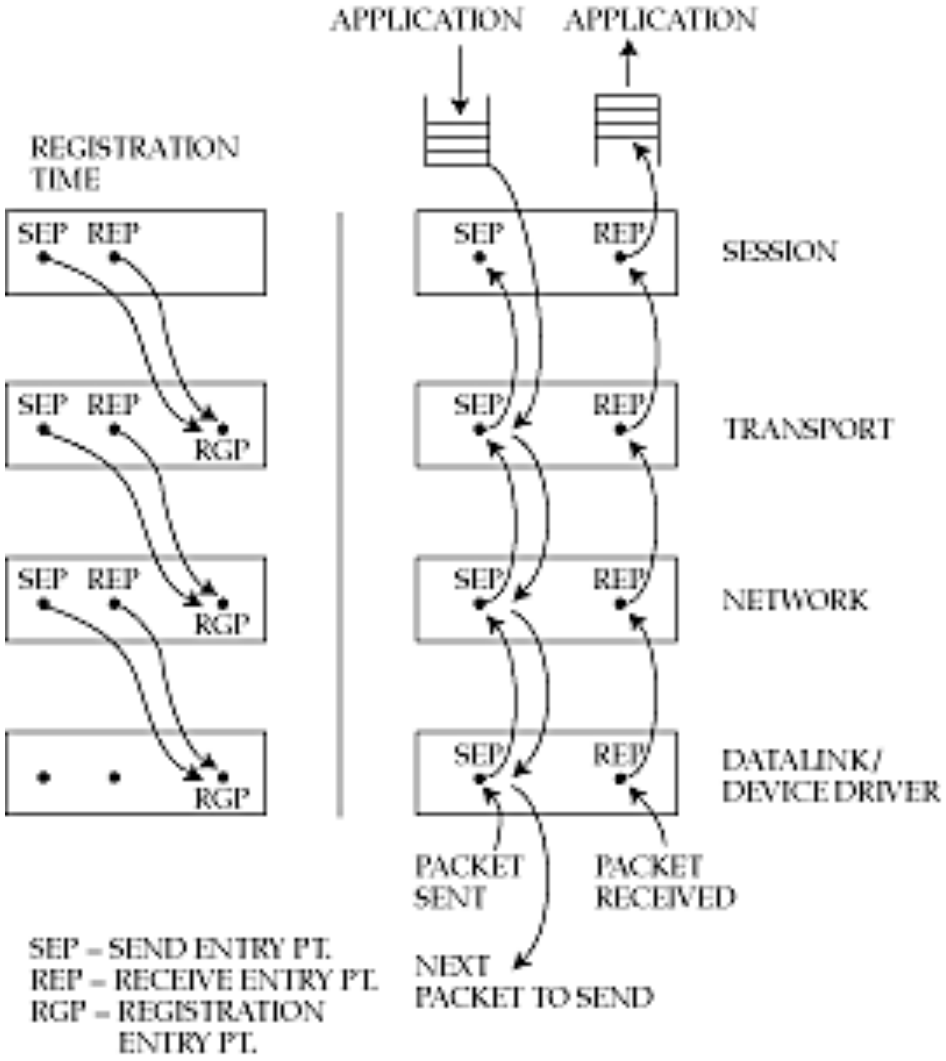
Single context



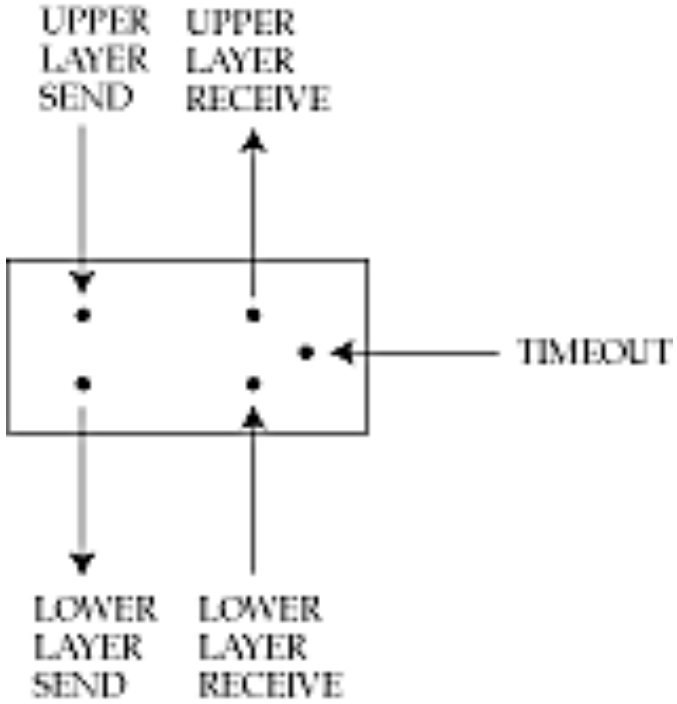
Tasks


APPLICATION
APPLICATION
BUFFER
DEVICE
T
N
T
DL
SCHEDULER
T = TRANSPORT
N = NETWORK
DL = DATALINK

Upcalls


APPLICATION
APPLICATION
REGISTRATION TIME
SEP
REP
RGP
SESSION
TRANSPORT
NETWORK
DATALINK/ DEVICE DRIVER
PACKET SENT
PACKET RECEIVED
NEXT PACKET TO SEND
SEP – SEND ENTRY PT.
REP – RECEIVE ENTRY PT.
RGP – REGISTRATION ENTRY PT.

Protocol implementation



Some numbers

- 10 Kbps 400 ms
- 100 Kbps, 40 ms
- 1 Mbps, 4 ms
- 100 Mbps, 40 μs
- User-to-kernel context switch ~40 μs
- Copying the packet ~25 μs
- Checksum in software ~40 μs
- Scheduling delays ~150 μs (depends on workload)
- Interrupt handling ~10-50 μs (depends on the bus)
- Protocol processing ~15 -100 μs (depends on protocol complexity)

Rules of thumb

- Optimize common case
- Watch out for bottlenecks
- Fine tune inner loops
- Choose good data structures
- Beware of data touching
- Minimize # packets sent
- Send largest packets possible
- Cache hints
- Use hardware
- Exploit application properties

Common Protocols

An Engineering Approach to Computer Networking

The grand finale

- Previous chapters presented principles, but not protocol details
  - these change with time
  - real protocols draw many things together
- Overview of real protocols
  - standards documents are the final resort
- Three sets of protocols
  - telephone
  - Internet
  - ATM

Telephone network protocols

	Data Plane	*Control Plane (SS7)*
App	Voice/Fax	ASE/ISDN-UP TCAP
Session		
Transport		
Network		SCCP/MTP-3
Datalink	Sonet/PDH	MTP-2
Physical	Many	MTP-1

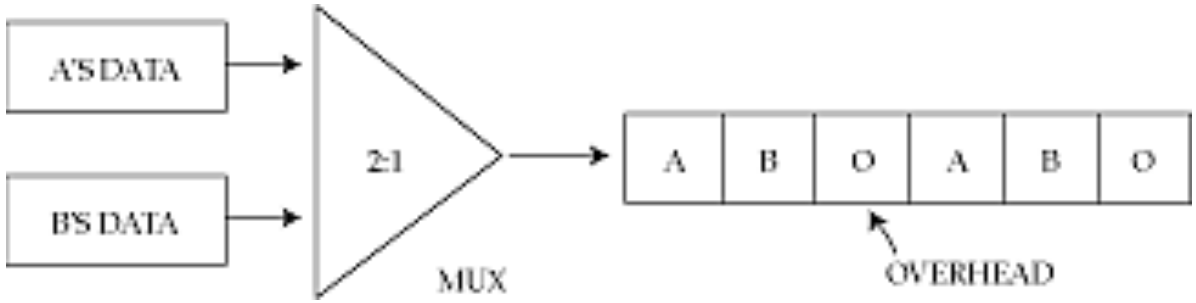
Traditional digital transmission

- Long distance trunks carry multiplexed calls
- Standard multiplexing levels
- Digital transmission hierarchy

	US and Japan		
Multiplexing level	Name	# calls	Rate (Mbps)
1	DS1	24	1.544
2	DS2	96	6.312
3	DS3	672	44.736
4	DS4	4032	274.176

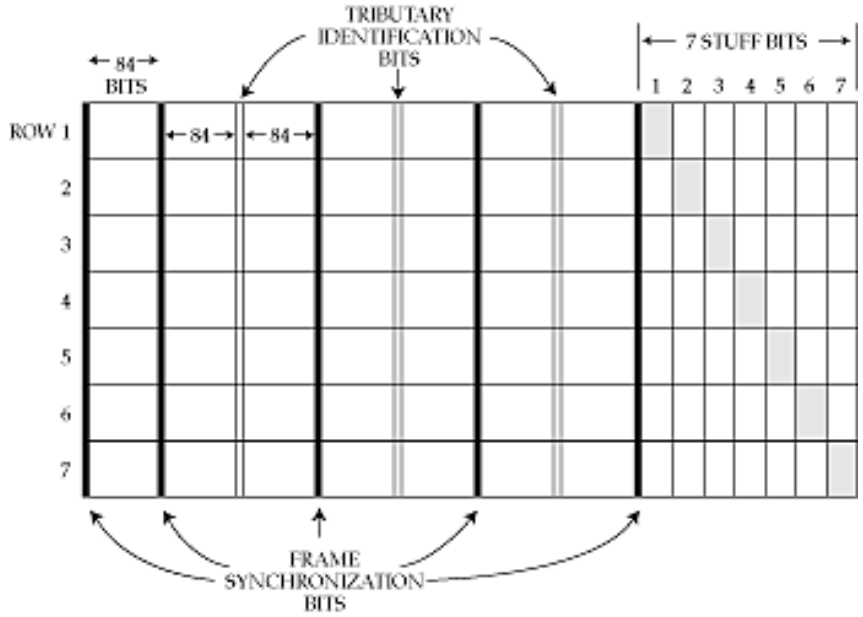
Plesiochronous hierarchy

- Plesiochronous = nearly synchronous
- Tight control on deviation from synchrony
- What if stream runs a little faster or slower?
- Need *justification*



Justification

- Output runs a bit faster always
- Overhead identifies bits from a particular stream
- If a stream runs faster, use overhead to identify it
- Overhead used everywhere except at first level (DS1)



Problems with plesiochrony

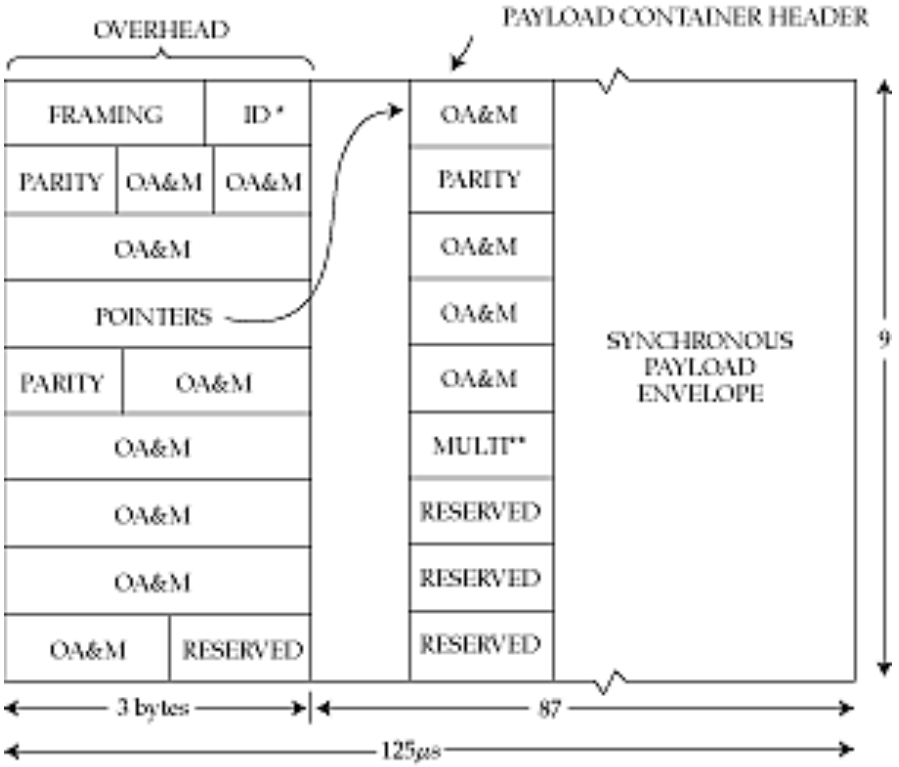
- Incompatible hierarchies around the world
- Data is spread out! Hard to extract a single call
- Cannot switch bundles of calls

Synchronous Digital Hierarchy

- All levels are synchronous
- Justification uses pointers

	Data Rate (Mbps)	US Name
1	51.84	OC-1
2	155.52	OC-3
3	466.56	OC-9
4	622.08	OC-12
5	933.12	OC-18
6	1244.16	OC-24
8	1866.24	OC-36
9	2488.32	OC-48
	9953.28	OC-192

SDH (SONET) frame



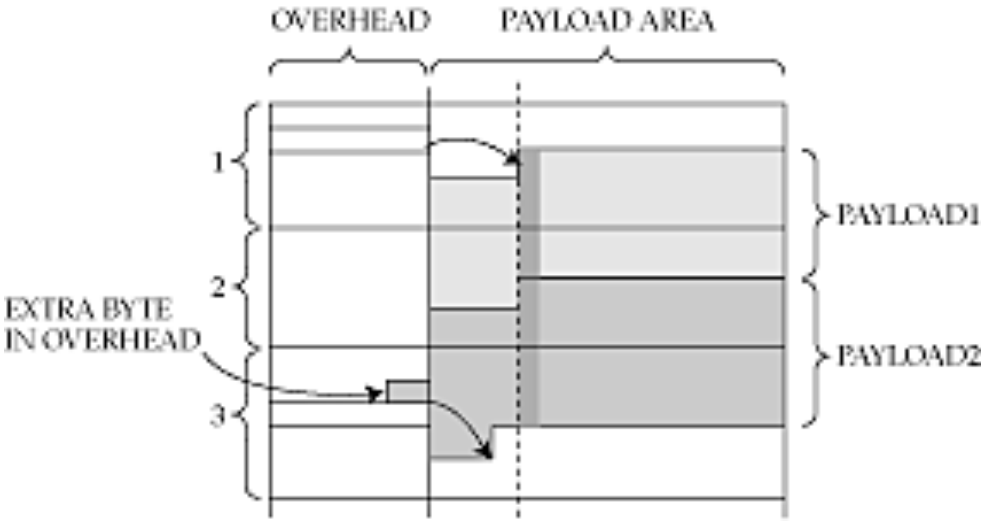
* ID – IDENTIFIES THE OC-1 NUMBER (1 . . N) IN AN OC-N FRAME

** MULTI – INDICATES THAT PAYLOAD SPANS MULTIPLE PAYLOAD ENVELOPES

SDH

- 9 rows, 90 columns
- Each payload container (SPE) served in 125 microseconds
- One byte = 1 call
- All overhead is in the headers
- Pointers for justification
  - if sending too fast, use a byte in the overhead, increasing sending rate
  - if sending too slow, skip a byte and move the pointer
  - can always locate a payload envelope, and thus a call within it => cheaper add drop mux

SDH justification



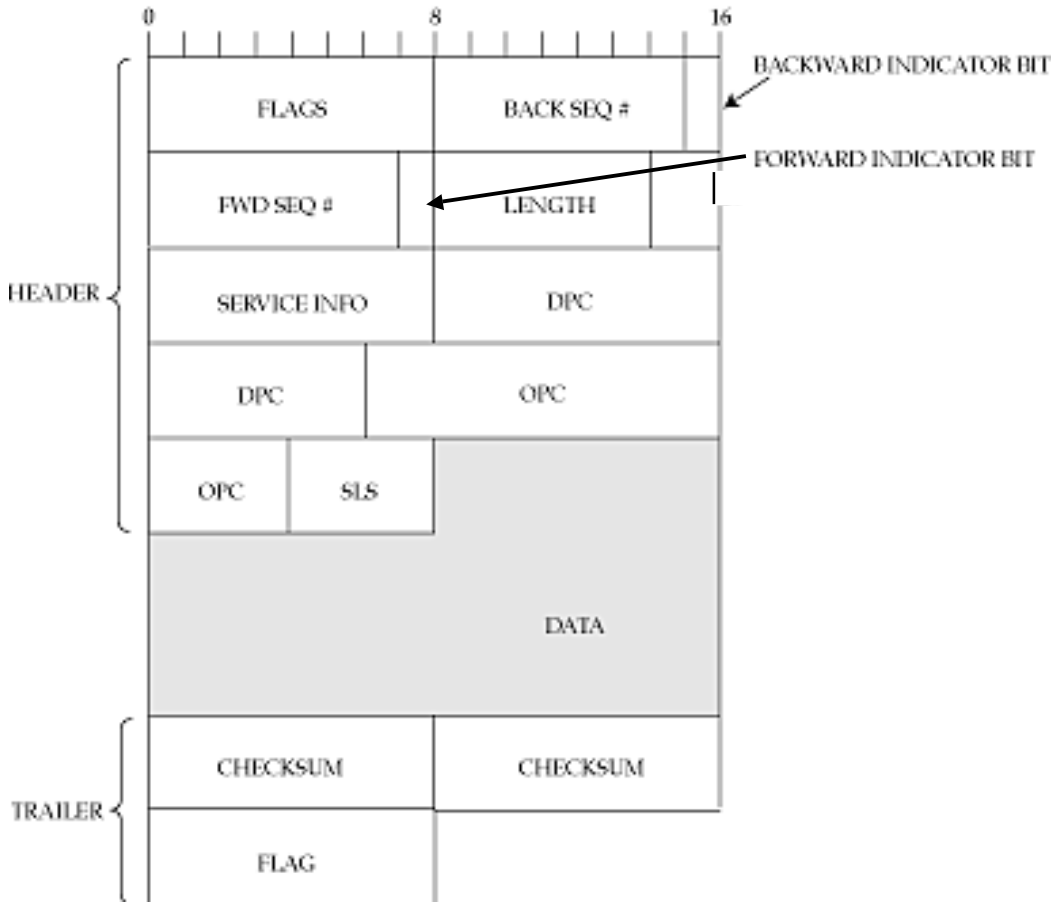
Signaling System 7 (SS7)

OSI layer name	SS7 layer name	Functionality	Internet example
Application	Application Service Element	Application	FTP
	Transaction Capabilities Application part	RPC	RPC
Transport	Signaling Connection Control Part	Connections, sequence numbers, segmentation and reassembly, flow control	TCP
Network	Message Transfer Part 3 (MTP-3)	Routing	IP
Datalink	MTP-2	Framing , link-level error detection and retransmission	Ethernet
Physical	MTP-1	Physical bit transfer	Ethernet

SS7 example

- Call forwarding
- To register
  - call special number
  - connects to ASE
  - authenticates user, stores forwarding number in database
- On call arrival
  - call setup protocol checks database for forwarding number
  - if number present, reroutes call
- SS7 provides all the services necessary for communication and coordination between registry ASE, database, and call setup entity

MTP Header


0
8
16
BACKWARD INDICATOR BIT
FORWARD INDICATOR BIT
FLAGS
BACK SEQ #
FWD SEQ #
LENGTH
HEADER
SERVICE INFO
DPC
DPC
OPC
OPC
SLS
DATA
CHECKSUM
CHECKSUM
TRAILER
FLAG

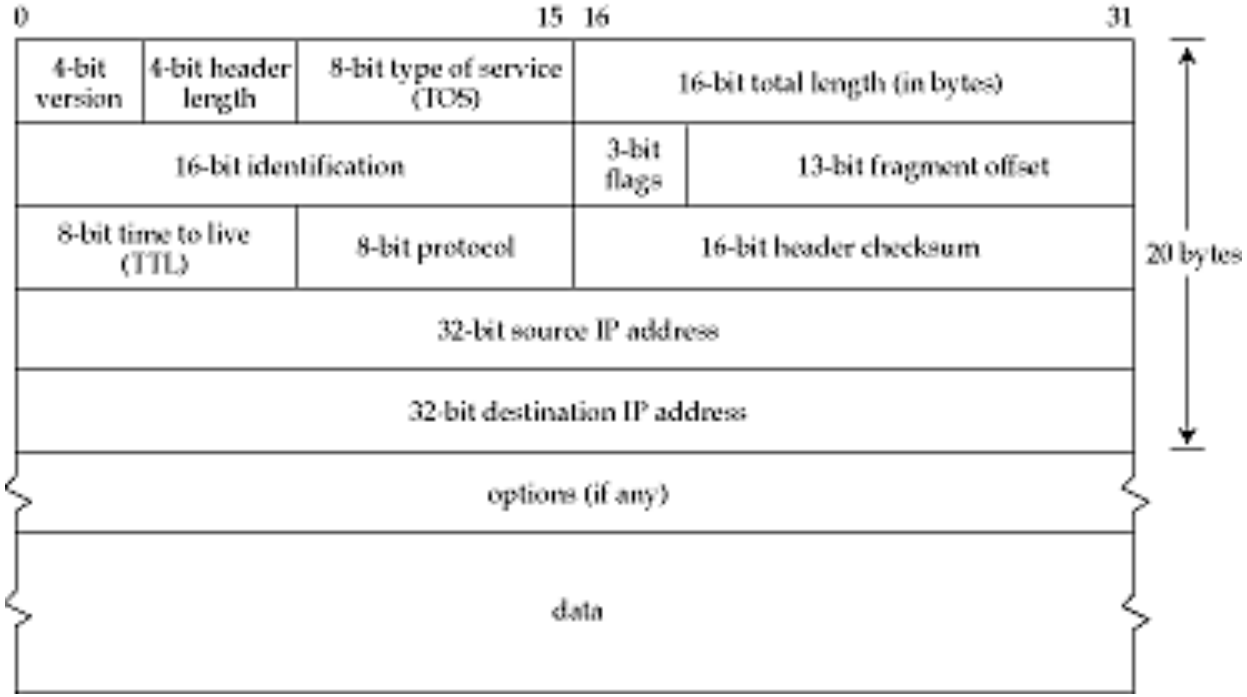
Internet stack

	Data Plane	*Control Plane*
App	HTTP	RSVP/OSPF
Session	Sockets/Streams	
Transport	TCP/UDP	
Network	IP	IP/ICMP
Datalink	Many	Many
Physical	Many	Many

IP

- Unreliable
- Best effort
- End-to-end
- IP on everything- interconnect the world

IP


0
15
16
31
4-bit version
4-bit header length
8-bit type of service (TOS)
16-bit total length (in bytes)
16-bit identification
3-bit flags
13-bit fragment offset
8-bit time to live (TTL)
8-bit protocol
16-bit header checksum
20 bytes
32-bit source IP address
32-bit destination IP address
options (if any)
data

Fragmentation

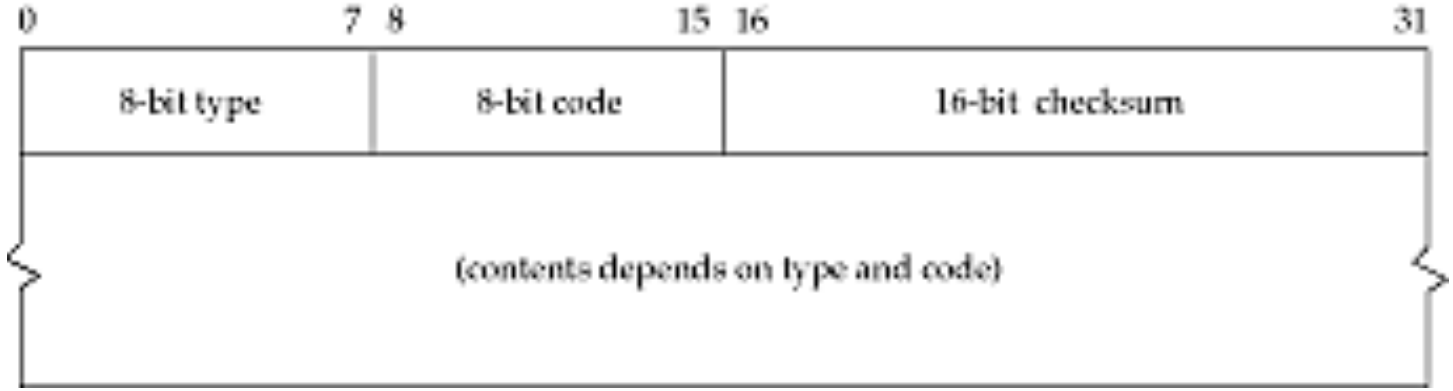
- IP can fragment, reassemble at receiver
- Fragment offset field
- More fragments flag and Don’t fragment flag
- Reassembly lockup
  - decrement timer and drop when it reaches 0
- Fragmentation is harmful
  - extra work
  - lockup
  - error multiplication
- Path MTU discovery
  - send large pkt with Don’t fragment set
  - if error, try smaller

IP fields

- TTL
  - decremented on each hop
  - decremented every 500 ms at endpt
  - terminates routing loops
- Traceroute
  - if router decrements to 0, send ICMP error packet
  - source sends packets with increasing TTL and waits for errors
- Options
  - record route
  - timestamp
  - loose source routing

ICMP

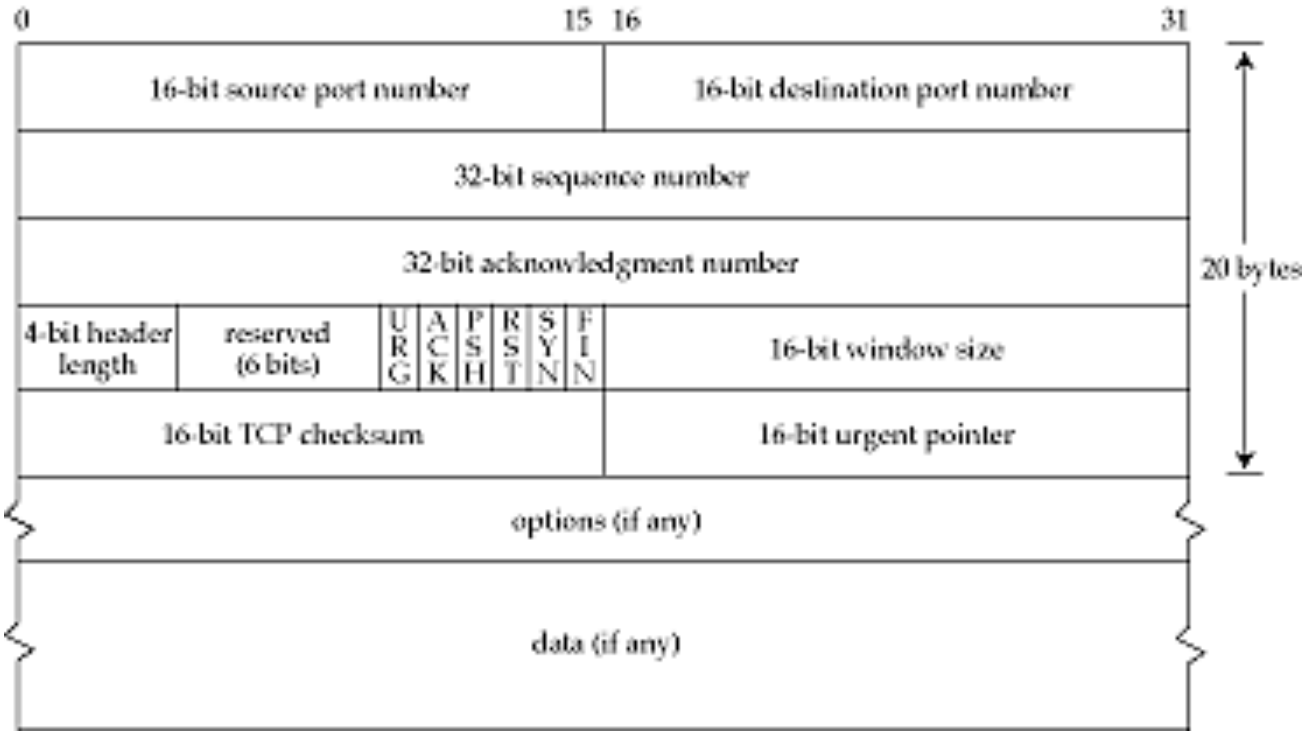
- Destination unreachable
- Source quench
- Redirect
- Router advertisement
- Time exceeded (TTL)
- Fragmentation needed, but Dont frag flag set



TCP

- Multiplexed
- Duplex
- Connection-oriented
- Reliable
- Flow-controlled
- Byte-stream

TCP


0
15
16
31
16-bit source port number
16-bit destination port number
32-bit sequence number
32-bit acknowledgment number
20 bytes
4-bit header length
reserved (6 bits)
URG
ACK
PSH
RST
SYN
FIN
16-bit window size
16-bit TCP checksum
16-bit urgent pointer
options (if any)
data (if any)

Fields

- Port numbers
- Sequence and ack number
- Header length
- Window size
  - 16 bits => 64 Kbytes (more with scaling)
  - receiver controls the window size
  - if zero, need sender persistence
  - silly window syndrome
- Checksum
- Urgent pointer
- Options
  - max segment size

HTTP

- Request response
- Protocol is simple, browser is complex
- Address space encapsulation
- Request types
  - GET
  - HEAD
  - POST
- Response
  - status
  - headers
  - body

ATM stack

	Data Plane	*Control Plane*
Application		UNI/PNNI
Application		Q.2931
Session		
Transport		SSCOP
Network	AAL1-5	S-AAL (AAL5)
Data Link	ATM	ATM
Physical	Many	Many

ATM

- Connection-oriented
- In-sequence
- Unreliable
- Quality of service assured

Virtual paths

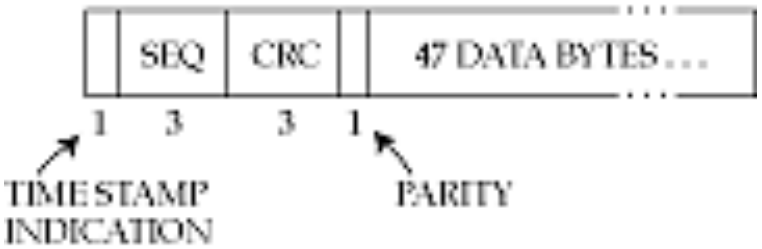
- High order bits of VCI
- All VCIs in a VP share path and resource reservation
- Saves table space in switches
  - faster lookup
- Avoids signaling
- May waste resources
- Dynamic renegotiation of VP capacity may help
- Set of virtual paths defines a *virtual private network*

AAL

- Was supposed to provide “rest of stack”
- Scaled back
- 4 versions: 1, 2, 3/4, 5
- Only 1, 3/4 and 5 important in practice

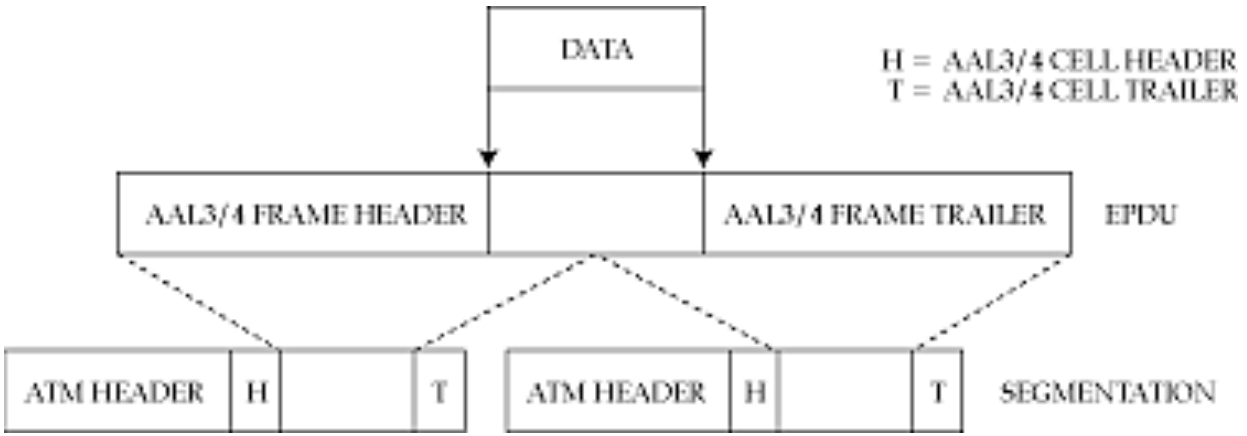
AAL 1

- For synchronous apps
  - provides timestamps and clocking
  - sequencing
  - always CBR
  - FEC in data bytes



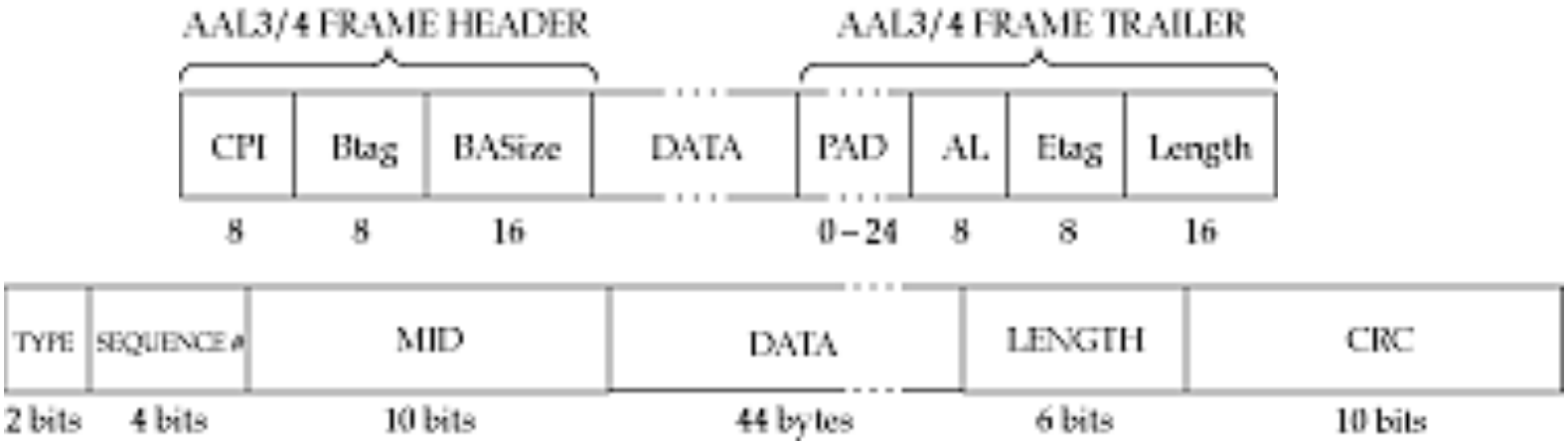
AAL 3/4

- For data traffic (from a telco perspective!)
- First create an encapsulated protocol data unit EPDU
  - (common part convergence sublayer-protocol data unit CPCS-PDU)
- Then fragment it and add ATM headers



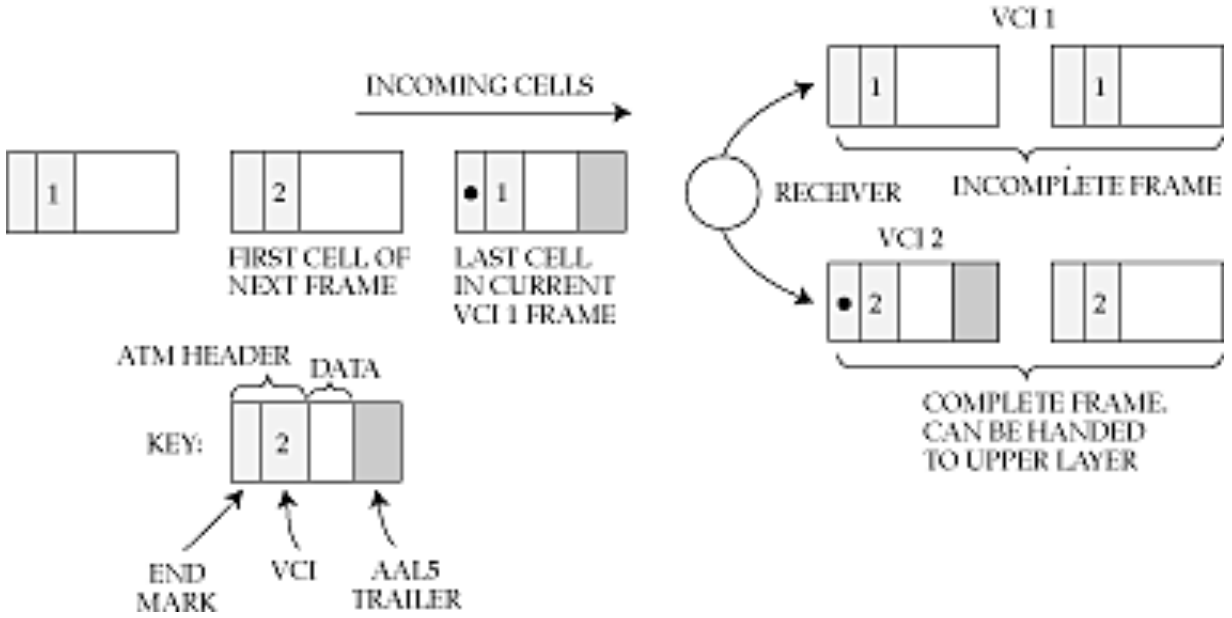
AAL 3/4

- Error detection, segmentation, reassembly
- Header and trailer per EPDU *and* per-cell header!

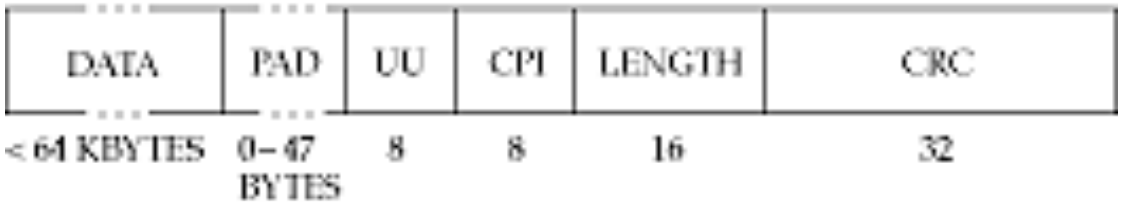


AAL 5

- Violates layering, but efficient
- Bit in header marks end of frame



AAL5 frame format



SSCOP

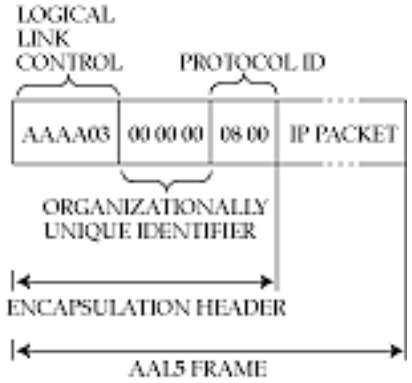
- Reliable transport for signaling messages
- Functionality similar to TCP
  - error control (described below)
  - flow control (static window)
- Four packet types
  - sequenced data / poll / stat / ustat
- No acks!
- Sender polls, receiver sends status
  - includes cumulative ack and window size
- If out of order, sends unsolicited status (ustat)
- Key variable is poll interval

IP-over-ATM

- Key idea: treat ATM as a link-level technology
  - ignore routing and QoS aspects
- Key problems
  - ATM is connection-oriented and IP is not
  - different addressing schemes
  - ATM LAN is point-to-point while IP assumes broadcast
- Basic technologies
  - IP encapsulation in ATM
  - Resolving IP addresses to ATM addresses
  - Creating an ATM-based IP subnet
  - Mapping multicast groups to ATM

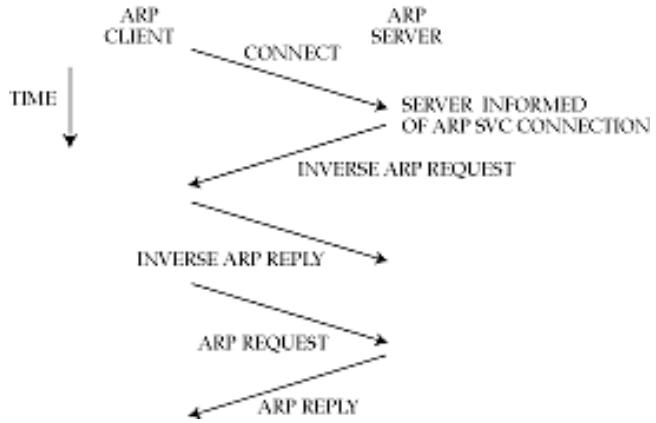
IP encapsulation in ATM

- Put data portion of IP packets in AAL5 frame
  - works only if endpoints understand AAL5
- Instead, place entire IP packet with AAL5 frame
- General solution allows *multiprotocol encapsulation*



Resolving IP addresses to ATM addresses

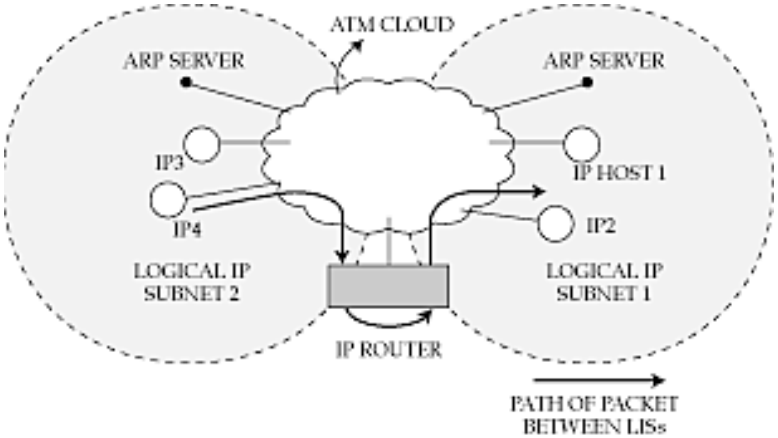
- Need something like ARP, but can't use broadcast
- Designate one of the ATM hosts as an ARP server



- Inverse ARP automatically creates database

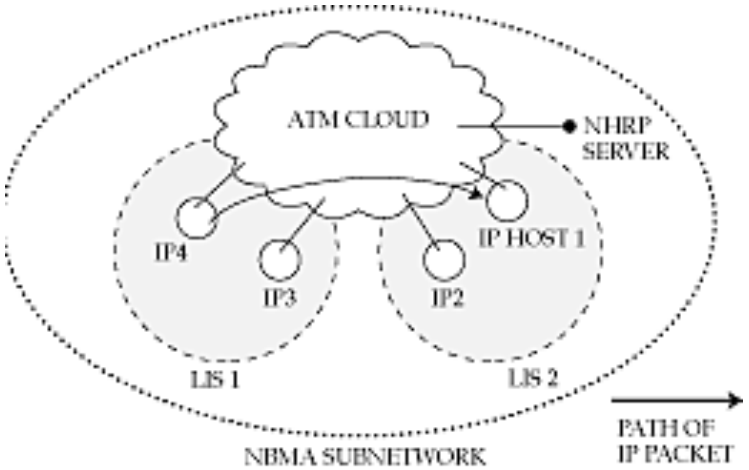
Creating an ATM-based IP subnet

- IP assumes free availability of bandwidth within a subnet
- If all hosts on ATM are on same IP subnet, broadcast reaches all => congestion
- Partition into *logical IP subnets*
  - at the cost of longer paths between ATM-attached hosts



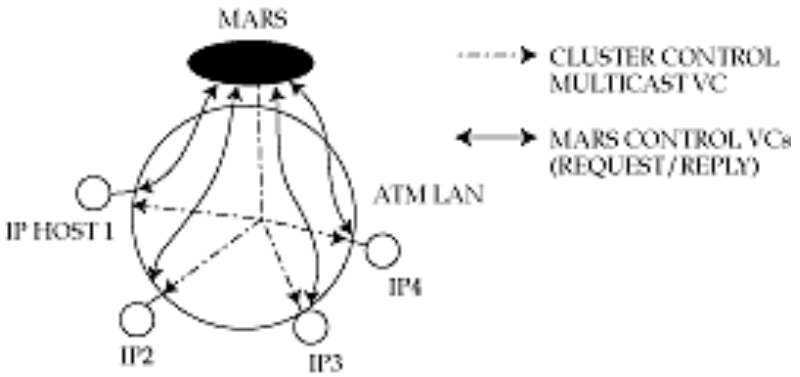
Next-hop routing

- Avoids long paths
- Next-hop server stores IP-to-ATM translations independent of subnet boundaries
  - like DNS



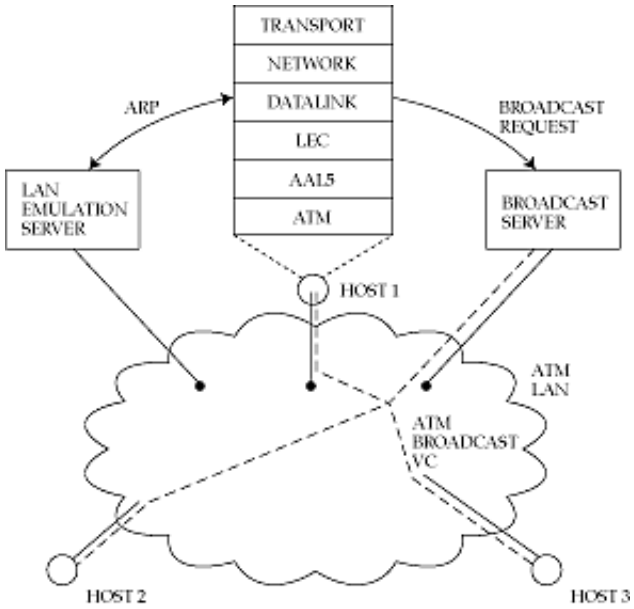
Resolving multicast addresses

- ARP server cannot resolve multicast addresses (why?)
- Actively maintain set of endpoints that correspond to a particular Class D address
- *Multicast Address Resolution Server* provides and updates this translation



LAN emulation

- If destination is on same LAN, can use ATM underneath datalink layer
- Need to translate from MAC address to ATM address
- Also need to emulate broadcast for Ethernet/FDDI



Cells in Frame (CIF)

- Solutions so far require expensive ATM host-adapter card
- Can we reuse Ethernet card?
- Encapsulate AAL5 frame in Ethernet header on point-to-point Ethernet link
- CIF-Attachment Device at other end decapsulates and injects the frame into an ATM network
- Software on end-system thinks that it has a local host adapter
- *Shim* between ATM stack and Ethernet driver inserts CIF header with VCI and ATM cell header
  - may need to fragment AAL5 frame
  - can also forward partial frames
- Cheaper
  - also gives endpoints QoS guarantees, unlike LANE

Holding time problem

- After resolution, open an ATM connection, and send IP packet
- When to close it?
- Locality
  - more packets likely
  - hold the connection for a while to avoid next call setup
  - but pay per-second holding time cost
- Optimal solution depends on pricing policy and packet arrival characteristics
- Measurement-based heuristic works nearly optimally
  - create the inter-arrival time histogram
  - expect future arrivals to conform to measured distribution
  - close connection if expected cost exceeds expected benefit

Routing

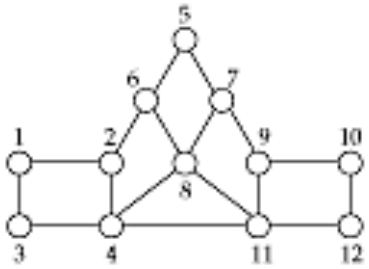
An Engineering Approach to Computer Networking

What is it?

- Process of finding a path from a source to every destination in the network
- Suppose you want to connect to Antarctica from your desktop
  - what route should you take?
  - does a shorter route exist?
  - what if a link along the route goes down?
  - what if you're on a mobile wireless link?
- Routing deals with these types of issues

Basics

- A *routing protocol* sets up a *routing table* in *routers* and *switch controllers*



ROUTING TABLE AT 1

Destination	Next hop	Destination	Next hop
1	—	7	2
2	2□	8□	2□
3	3□	9□	2□
4	3□	10□	2□
5	2□	11□	3□
6	2	12	3

- A node makes a *local* choice depending on *global* topology: this is the fundamental problem

Key problem

- How to make correct local decisions?
  - each router must know *something* about global state
- Global state
  - inherently large
  - dynamic
  - hard to collect
- *A routing protocol must intelligently summarize relevant information*

Requirements

- Minimize routing table space
  - fast to look up
  - less to exchange
- Minimize number and frequency of control messages
- Robustness: avoid
  - black holes
  - loops
  - oscillations
- Use optimal path

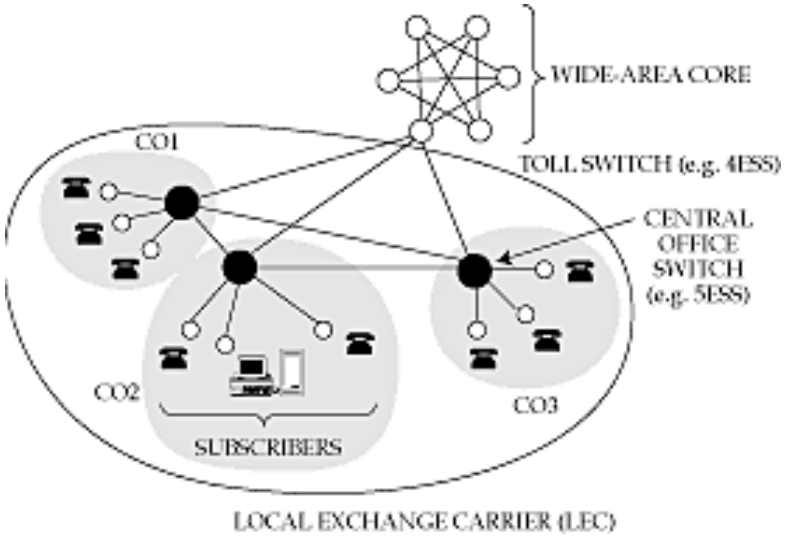
Choices

- Centralized vs. distributed routing
  - centralized is simpler, but prone to failure and congestion
- Source-based vs. hop-by-hop
  - how much is in packet header?
  - Intermediate: *loose source route*
- Stochastic vs. deterministic
  - stochastic spreads load, avoiding oscillations, but misorders
- Single vs. multiple path
  - primary and alternative paths (compare with stochastic)
- State-dependent vs. state-independent
  - do routes depend on current network state (e.g. delay)

Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Telephone network topology



- 3-level hierarchy, with a fully-connected core
- AT&T: 135 core switches with nearly 5 million circuits
- LECs may connect to multiple cores

Routing algorithm

- If endpoints are within same CO, directly connect
- If call is between COs in same LEC, use one-hop path between COs
- Otherwise send call to one of the cores
- Only major decision is at toll switch
  - one-hop or two-hop path to the destination toll switch
  - (why don't we need longer paths?)
- Essence of problem
  - which two-hop path to use if one-hop path is full

Features of telephone network routing

- Stable load
  - can predict pairwise load throughout the day
  - can choose optimal routes in advance
- Extremely reliable switches
  - downtime is less than a few minutes per year
  - can assume that a chosen route is available
  - can’t do this in the Internet
- Single organization controls entire core
  - can collect global statistics and implement global changes
- Very highly connected network
- Connections require resources (but all need the same)

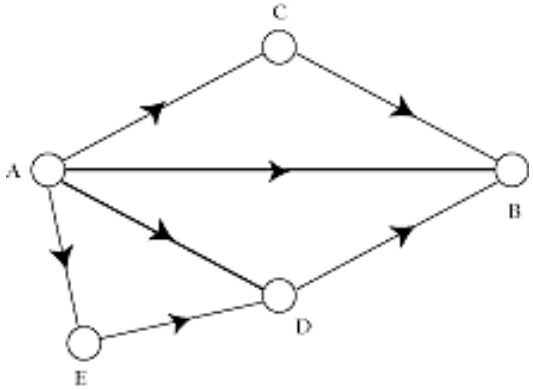
The cost of simplicity

- Simplicity of routing a historical necessity
- But requires
  - reliability in every component
  - logically fully-connected core
- Can we build an alternative that has same features as the telephone network, but is cheaper because it uses more sophisticated routing?
  - Yes: that is one of the motivations for ATM
  - But 80% of the cost is in the local loop
    - not affected by changes in core routing
  - Moreover, many of the software systems assume topology
    - too expensive to change them

Dynamic nonhierarchical routing (DNHR)

- Simplest core routing protocol
  - accept call if one-hop path is available, else drop
- DNHR
  - divides day into around 10-periods
  - in each period, each toll switch is assigned a primary one-hop path and a list of alternatives
  - can overflow to alternative if needed
  - drop only if all alternate paths are busy
    - *crankback*
- Problems
  - does not work well if actual traffic differs from prediction

Metastability



- Burst of activity can cause network to enter metastable state
  - high blocking probability even with a low load
- Removed by trunk reservation
  - prevents spilled traffic from taking over direct path

Trunk status map routing (TSMR)

- DNHR measures traffic once a week
- TSMR updates measurements once an hour or so
  - only if it changes “significantly”
- List of alternative paths is more up to date

Real-time network routing (RTNR)

- No centralized control
- Each toll switch maintains a list of lightly loaded links
- Intersection of source and destination lists gives set of lightly loaded paths
- Example
  - At A, list is C, D, E => links AC, AD, AE lightly loaded
  - At B, list is D, F, G => links BD, BF, BG lightly loaded
  - A asks B for its list
  - Intersection = D => AD and BD lightly loaded => ADB lightly loaded => it is a good alternative path
- Very effective in practice: only about a couple of calls blocked in core out of about 250 million calls attempted every day

Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

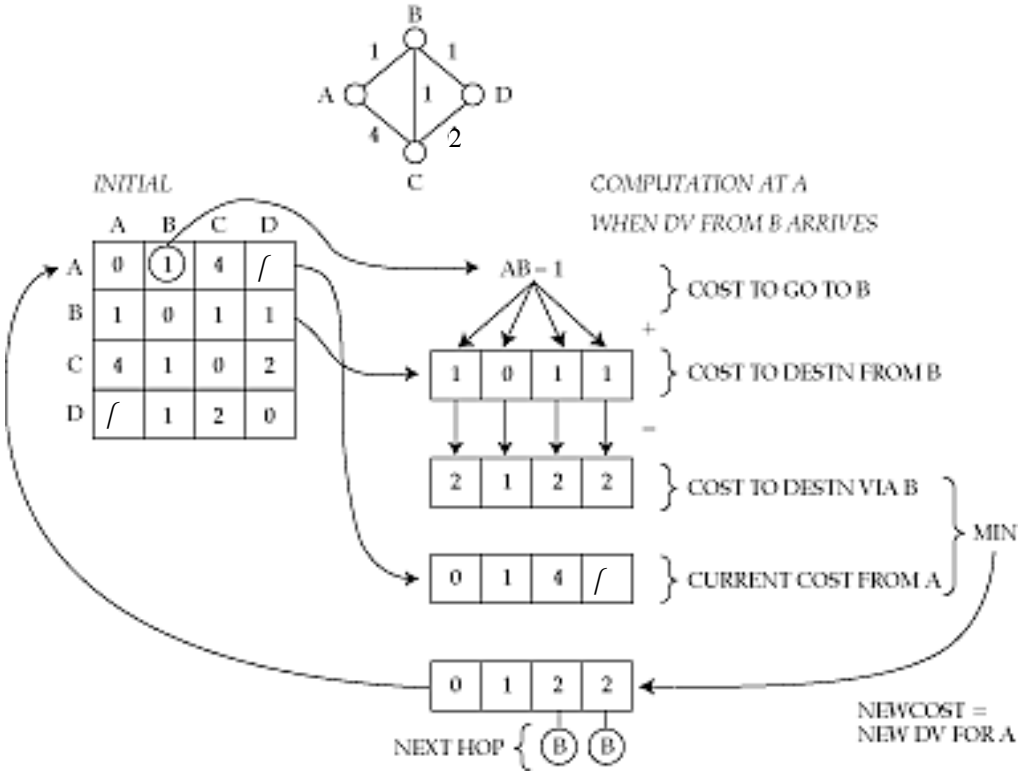
Distance vector routing

- Environment
  - links and routers unreliable
  - alternative paths scarce
  - traffic patterns can change rapidly
- Two key algorithms
  - distance vector
  - link-state
- Both assume router knows
  - address of each neighbor
  - cost of reaching each neighbor
- Both allow a router to determine global routing information by talking to its neighbors

Basic idea

- Node tells its neighbors its best idea of distance to *every* other node in the network
- Node receives these *distance vectors* from its neighbors
- Updates its notion of best path to each destination, and the next hop for this destination
- Features
  - distributed
  - adapts to traffic changes and link failures
  - suitable for networks with multiple administrative entities

Example

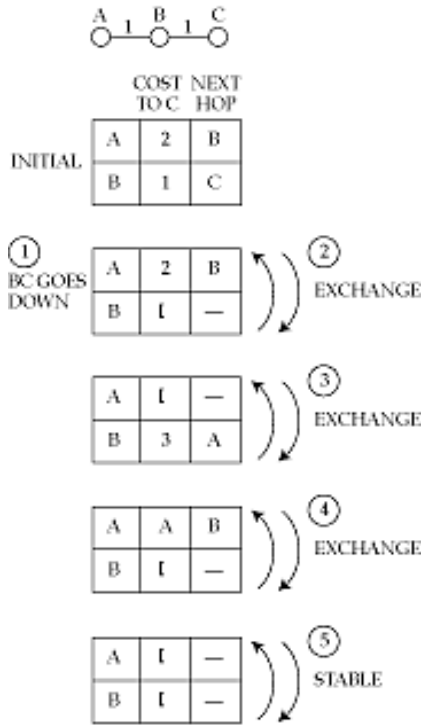

B
1
1
A
1
D
4
2
C
INITIAL
A B C D
A 0 1 4 ∫
B 1 0 1 1
C 4 1 0 2
D ∫ 1 2 0
COMPUTATION AT A
WHEN DV FROM B ARRIVES
AB = 1
COST TO GO TO B
+
1 0 1 1
COST TO DESTN FROM B
−
2 1 2 2
COST TO DESTN VIA B
MIN
0 1 4 ∫
CURRENT COST FROM A
0 1 2 2
NEXT HOP { B B
NEWCOST =
NEW DV FOR A

Why does it work

- Each node knows its true cost to its neighbors
- This information is spread to its neighbors the first time it sends out its distance vector
- Each subsequent dissemination spreads the truth one hop
- Eventually, it is incorporated into routing table everywhere in the network
- Proof: Bellman and Ford, 1957

Problems with distance vector

- Count to infinity



Dealing with the problem

- Path vector
  - DV carries path to reach each destination
- Split horizon
  - never tell neighbor cost to X if neighbor is next hop to X
  - doesn't work for 3-way count to infinity (see exercise)
- Triggered updates
  - exchange routes on change, instead of on timer
  - faster count up to infinity
- More complicated
  - source tracing
  - DUAL

Outline

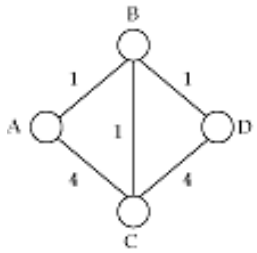
- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Link state routing

- In distance vector, router knows only *cost* to each destination
  - hides information, causing problems
- In link state, router knows entire network topology, and computes shortest path by itself
  - independent computation of routes
  - potentially less robust
- Key elements
  - topology dissemination
  - computing shortest routes

Link state: topology dissemination

- A router describes its neighbors with a *link state packet (LSP)*



LSPs CREATED BY A

A'S ID	B'S ID	COST
A	B	1
A	C	4

- Use *controlled flooding* to distribute this everywhere
  - store an LSP in an *LSP database*
  - if new, forward to every interface other than incoming one
  - a network with E edges will copy at most 2E times

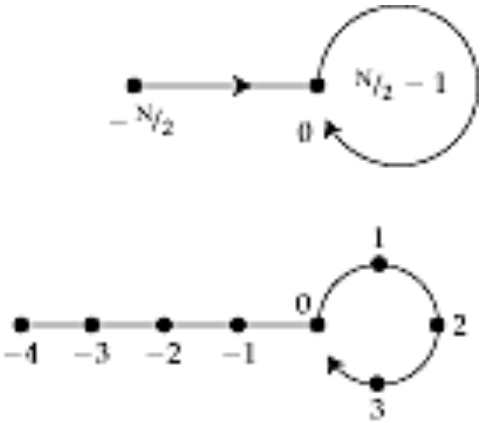
Sequence numbers

- How do we know an LSP is new?
- Use a sequence number in LSP header
- Greater sequence number is newer
- What if sequence number wraps around?
  - smaller sequence number is now newer!
  - (hint: use a large sequence space)
- On boot up, what should be the initial sequence number?
  - have to somehow purge old LSPs
  - two solutions
    - aging
    - lollipop sequence space

Aging

- Creator of LSP puts timeout value in the header
- Router removes LSP when it times out
  - also floods this information to the rest of the network (why?)
- So, on booting, router just has to wait for its old LSPs to be purged
- But what age to choose?
  - if too small
    - purged before fully flooded (why?)
    - needs frequent updates
  - if too large
    - router waits idle for a long time on rebooting

A better solution



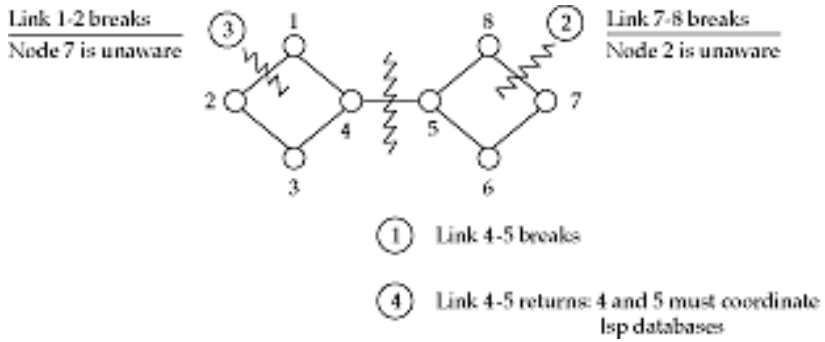
- Need a *unique* start sequence number
- a is older than b if:
  - a < 0 and a < b
  - a > o, a < b, and b-a < N/4
  - a > 0, b > 0, a > b, and a-b > N/4

More on lollipops

- If a router gets an older LSP, it tells the sender about the newer LSP
- So, newly booted router quickly finds out its most recent sequence number
- It jumps to one more than that
- -N/2 is a *trigger* to evoke a response from community memory

Recovering from a partition

- On partition, LSP databases can get out of synch



- Databases described by database descriptor records
- Routers on each side of a newly restored link talk to each other to update databases (determine missing and out-of-date LSPs)

Router failure

- How to detect?
  - HELLO protocol
- HELLO packet may be corrupted
  - so age anyway
  - on a timeout, flood the information

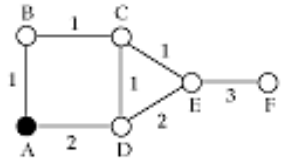
Securing LSP databases

- LSP databases *must* be consistent to avoid routing loops
- Malicious agent may inject spurious LSPs
- Routers must actively protect their databases
  - checksum LSPs
  - ack LSP exchanges
  - passwords

Computing shortest paths

- Basic idea
  - maintain a set of nodes P to whom we know shortest path
  - consider every node one hop away from nodes in P = T
  - find every way in which to reach a given node in T, and choose shortest one
  - then add this node to P

Example



B(A,1) means B was reached by A, cost 1

PERMANENT	TEMPORARY	COMMENTS
A	B(A,1), D(A,2)	ROOT AND ITS NEIGHBORS
A, B(A,1)	D(A,2), C(B,2)	ADD C(B,2)
A, B(A,1) D(A,2)	E(D,4), C(B,2)	C(D,3) DIDN'T MAKE IT
A, B(A,1) D(A,2), C(B,2)	E(C,3)	E(D,4) TOO LONG
A, B(A,1) D(A,2), C(B,2) E(C,3)	F(E,6)	
A, B(A,1) C(B,2), D(A,2) E(C,3), F(E,6)	NULL	STOP

Link state vs. distance vector

- Criteria
  - stability
  - multiple routing metrics
  - convergence time after a change
  - communication overhead
  - memory overhead
- Both are evenly matched
- Both widely used

Outline

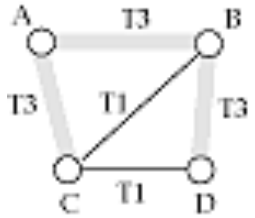
- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Choosing link costs

- Shortest path uses link costs
- Can use either static of dynamic costs
- In both cases: cost determine amount of traffic on the link
  - lower the cost, more the expected traffic
  - if dynamic cost depends on load, can have oscillations (why?)

Static metrics

- Simplest: set all link costs to 1 => min hop routing
  - but 28.8 modem link is not the same as a T3!
- Give links weight proportional to capacity



WEIGHTS
T3 – 1 T1 – 10

Dynamic metrics

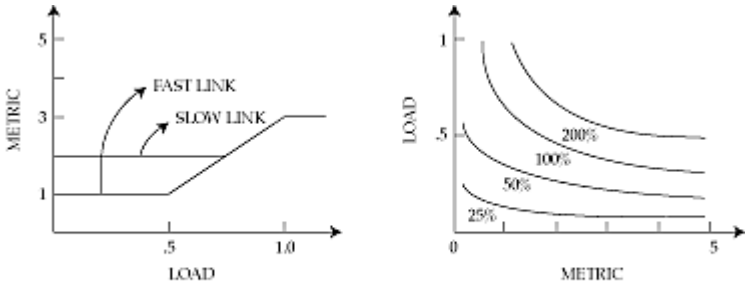
- A first cut (ARPAnet original)
- Cost proportional to length of router queue
  - independent of link capacity
- Many problems when network is loaded
  - queue length averaged over a small time => transient spikes caused major rerouting
  - wide dynamic range => network completely ignored paths with high costs
  - queue length assumed to predict future loads => opposite is true (why?)
  - no restriction on successively reported costs => oscillations
  - all tables computed simultaneously => low cost link flooded

Modified metrics

- queue length averaged over a small time
- wide dynamic range queue
- queue length assumed to predict future loads
- no restriction on successively reported costs
- all tables computed simultaneously

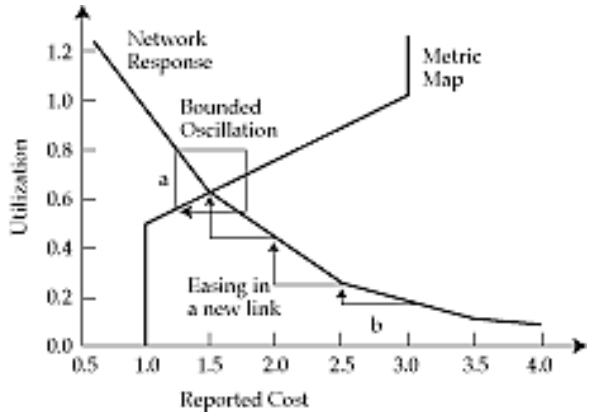
- queue length averaged over a longer time
- dynamic range restricted
- cost also depends on intrinsic link capacity
- restriction on successively reported costs
- attempt to stagger table computation

Routing dynamics



(a) METRIC MAP

(b) NETWORK RESPONSE MAP



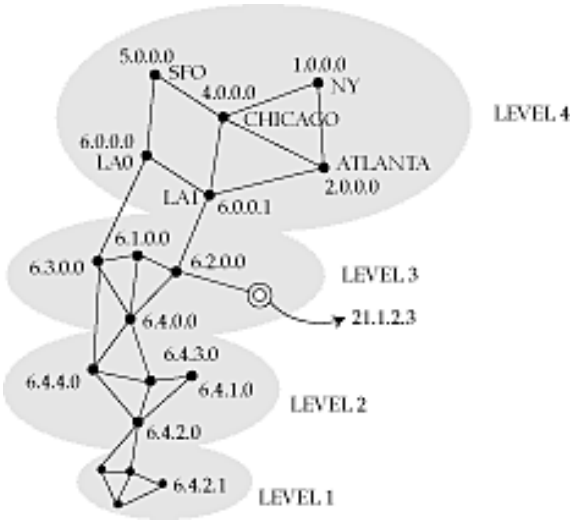
Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Hierarchical routing

- Large networks need large routing tables
  - more computation to find shortest paths
  - more bandwidth wasted on exchanging DVs and LSPs
- Solution:
  - hierarchical routing
- Key idea
  - divide network into a set of domains
  - gateways connect domains
  - computers within domain unaware of outside computers
  - gateways know only about other gateways

Example



- Features
  - only a few routers in each level
  - not a strict hierarchy
  - gateways participate in multiple routing protocols
  - non-aggregable routers increase core table space

Hierarchy in the Internet

- Three-level hierarchy in addresses
  - network number
  - subnet number
  - host number
- Core advertises routes only to networks, not to subnets
  - e.g. 135.104.*, 192.20.225.*
- Even so, about 80,000 networks in core routers (1996)
- Gateways talk to backbone to find best next-hop to every other network in the Internet

External and summary records

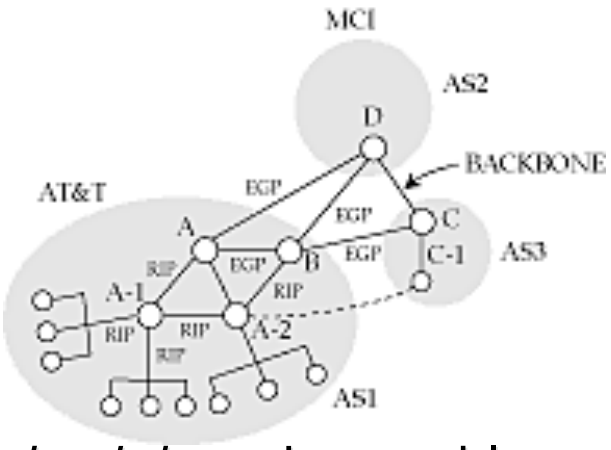
- If a domain has multiple gateways
  - *external* records tell hosts in a domain which one to pick to reach a host in an external domain
    - e.g allows 6.4.0.0 to discover shortest path to 5.* is through 6.0.0.0
  - *summary* records tell backbone which gateway to use to reach an internal node
    - e.g. allows 5.0.0.0 to discover shortest path to 6.4.0.0 is through 6.0.0.0
- External and summary records contain distance from gateway to external or internal node
  - unifies distance vector and link state algorithms

Interior and exterior protocols

- Internet has three levels of routing
  - highest is at *backbone* level, connecting *autonomous systems (AS)*
  - next level is within AS
  - lowest is within a LAN
- Protocol between AS gateways: exterior gateway protocol
- Protocol within AS: interior gateway protocol

Exterior gateway protocol

- Between untrusted routers
  - mutually suspicious
- Must tell a *border gateway* who can be trusted and what paths are allowed



- *Transit* over *backdoors* is a problem

Interior protocols

- Much easier to implement
- Typically partition an AS into *areas*
- Exterior and summary records used between areas

Issues in interconnection

- May use different schemes (DV vs. LS)
- Cost metrics may differ
- Need to:
  - convert from one scheme to another (how?)
  - use the lowest common denominator for costs
  - manually intervene if necessary

Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Common routing protocols

- Interior
  - RIP
  - OSPF
- Exterior
  - EGP
  - BGP
- ATM
  - PNNI

RIP

- Distance vector
- Cost metric is hop count
- Infinity = 16
- Exchange distance vectors every 30 s
- Split horizon
- Useful for small subnets
  - easy to install

OSPF

- Link-state
- Uses areas to route packets hierarchically within AS
- Complex
  - LSP databases to be protected
- Uses *designated routers* to reduce number of endpoints

EGP

- Original exterior gateway protocol
- Distance-vector
- Costs are either 128 (reachable) or 255 (unreachable) => reachability protocol => backbone must be loop free (why?)
- Allows administrators to pick neighbors to peer with
- Allows backdoors (by setting backdoor cost < 128)

BGP

- Path-vector
  - distance vector annotated with entire path
  - also with policy attributes
  - guaranteed loop-free
- Can use non-tree backbone topologies
- Uses TCP to disseminate DVs
  - reliable
  - but subject to TCP flow control
- Policies are complex to set up

PNNI

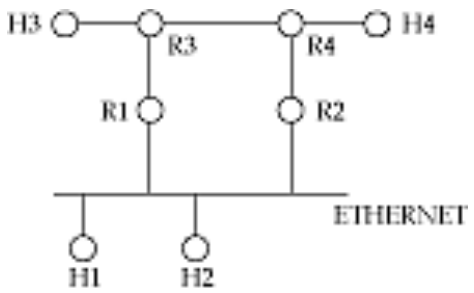
- Link-state
- Many levels of hierarchy
- Switch controllers at each level form a peer group
- Group has a group leader
- Leaders are members of the next higher level group
- Leaders summarize information about group to tell higher level peers
- All records received by leader are flooded to lower level
- LSPs can be annotated with per-link QoS metrics
- Switch controller uses this to compute source routes for call-setup packets

Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Routing within a broadcast LAN

- What happens at an endpoint?
- On a point-to-point link, no problem
- On a broadcast LAN
  - is packet meant for destination within the LAN?
  - if so, what is the datalink address ?
  - if not, which router on the LAN to pick?
  - what is the router's datalink address?



Internet solution

- All hosts on the LAN have the same subnet address
- So, easy to determine if destination is on the same LAN
- Destination's datalink address determined using ARP
  - broadcast a request
  - owner of IP address replies
- To discover routers
  - routers periodically sends router advertisements
    - with preference level and time to live
  - pick most preferred router
  - delete overage records
  - can also force routers to reply with *solicitation message*

Redirection

- How to pick the best router?
- Send message to arbitrary router
- If that router's next hop is another router on the same LAN, host gets a *redirect* message
- It uses this for subsequent messages

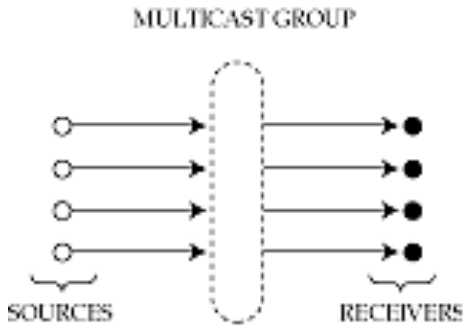
Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Multicast routing

- Unicast: single source sends to a single destination
- Multicast: hosts are part of a *multicast group*
  - packet sent by *any* member of a group are received by *all*
- Useful for
  - multiparty videoconference
  - distance learning
  - resource location

Multicast group



- Associates a set of senders and receivers with each other
  - but independent of them
  - created either when a sender starts sending from a group
  - or a receiver expresses interest in receiving
  - even if no one else is there!
- Sender does not need to know receivers' identities
  - *rendezvous point*

Addressing

- Multicast group in the Internet has its own Class D address
  - looks like a host address, but isn't
- Senders send to the address
- Receivers anywhere in the world request packets from that address
- "Magic" is in associating the two: *dynamic directory service*
- Four problems
  - which groups are currently active
  - how to express interest in joining a group
  - discovering the set of receivers in a group
  - delivering data to members of a group

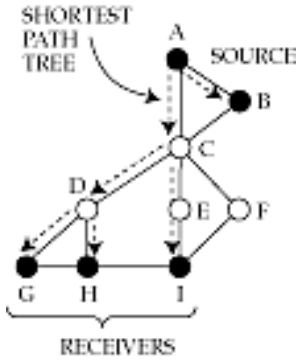
Expanding ring search

- A way to use multicast groups for resource discovery
- Routers decrement TTL when forwarding
- Sender sets TTL and multicasts
  - reaches all receivers <= TTL hops away
- Discovers local resources first
- Since heavily loaded servers can keep quiet, automatically distributes load

Multicast flavors

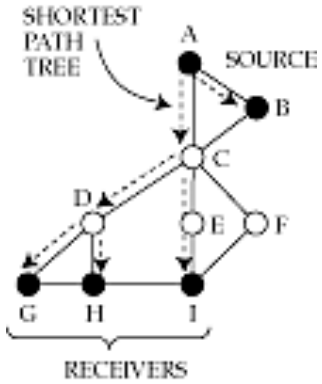
- Unicast: point to point
- Multicast:
  - point to multipoint
  - multipoint to multipoint
- Can simulate point to multipoint by a set of point to point unicasts
- Can simulate multipoint to multipoint by a set of point to multipoint multicasts
- The difference is efficiency

Example



- Suppose A wants to talk to B, G, H, I, B to A, G, H, I
- With unicast, 4 messages sent from each source
  - links AC, BC carry a packet in triplicate
- With point to multipoint multicast, 1 message sent from each source
  - but requires establishment of two separate multicast groups
- With multipoint to multipoint multicast, 1 message sent from each source,
  - single multicast group

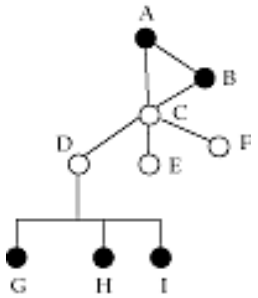
Shortest path tree



- Ideally, want to send exactly one multicast packet per link
  - forms a *multicast tree* rooted at sender
- Optimal multicast tree provides *shortest* path from sender to every receiver
  - *shortest-path* tree rooted at sender

Issues in wide-area multicast

- Difficult because
  - sources may join and leave dynamically
    - need to dynamically update shortest-path tree
  - leaves of tree are often members of broadcast LAN
    - would like to exploit LAN broadcast capability



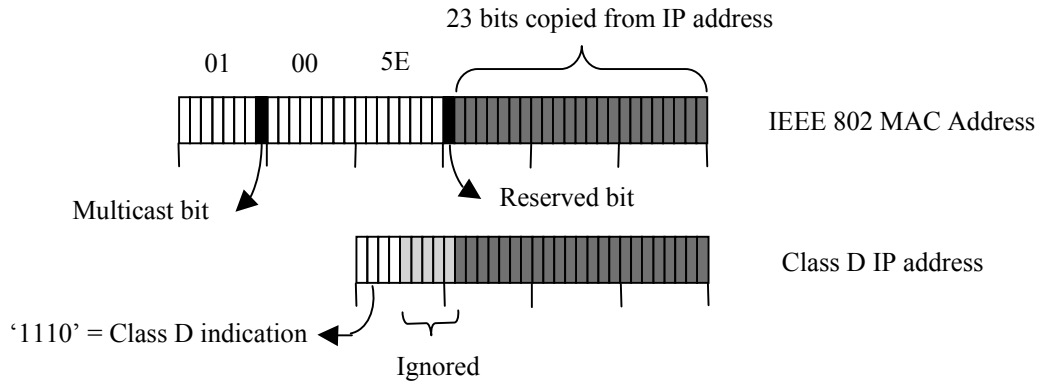
  - would like a receiver to join or leave without explicitly notifying sender
    - otherwise it will not scale

Multicast in a broadcast LAN

- Wide area multicast can exploit a LAN's broadcast capability
- E.g. Ethernet will multicast all packets with multicast bit set on destination address

- Two problems:
  - what multicast MAC address corresponds to a given Class D IP address?
  - does the LAN have contain any members for a given group (why do we need to know this?)

Class D to MAC translation



- Multiple Class D addresses map to the same MAC address
- Well-known translation algorithm => no need for a translation table

Internet Group Management Protocol

- Detects if a LAN has any members for a particular group
  - If no members, then we can *prune* the shortest path tree for that group by telling parent
- Router periodically broadcasts a *query* message
- Hosts reply with the list of groups they are interested in
- To suppress traffic
  - reply after random timeout
  - broadcast reply
  - if someone else has expressed interest in a group, drop out
- To receive multicast packets:
  - translate from class D to MAC and configure adapter

Wide area multicast

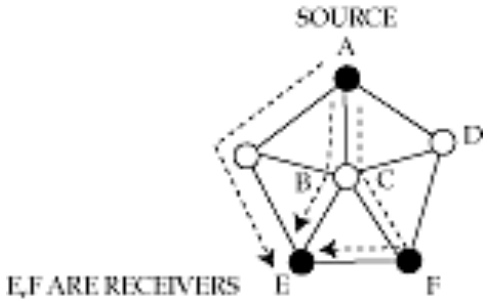
- Assume
  - each endpoint is a router
  - a router can use IGMP to discover all the members in its LAN that want to subscribe to each multicast group
- Goal
  - distribute packets coming from any sender directed to a given group to all routers on the path to a group member

Simplest solution

- Flood packets from a source to entire network
- If a router has not seen a packet before, forward it to all interfaces except the incoming one
- Pros
  - simple
  - always works!
- Cons
  - routers receive duplicate packets
  - detecting that a packet is a duplicate requires storage, which can be expensive for long multicast sessions

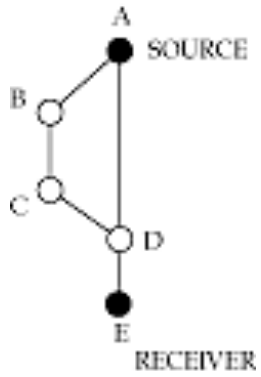
A clever solution

- *Reverse path forwarding*
- Rule
  - forward packet from S to all interfaces if and only if packet arrives on the interface that corresponds to the shortest path *to* S
  - no need to remember past packets
  - C need not forward packet received from D



Cleverer

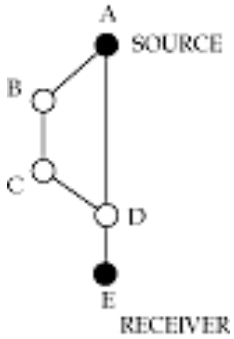
- Don’t send a packet downstream if you are not on the shortest path from the downstream router to the source
- C need not forward packet from A to E



- Potential confusion if downstream router has a choice of shortest paths to source (see figure on previous slide)

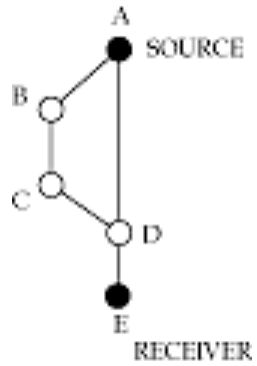
Pruning

- RPF does not completely eliminate unnecessary transmissions



- B and C get packets even though they do not need it
- Pruning => router tells parent in tree to stop forwarding
- Can be associated either with a multicast group or with a source *and* group
  - trades selectivity for router memory

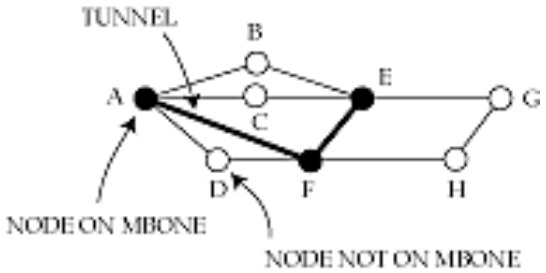
Rejoining



- What if host on C's LAN wants to receive messages from A after a previous prune by C?
  - IGMP lets C know of host's interest
  - C can send a *join(group, A)* message to B, which propagates it to A
  - or, periodically flood a message; C refrains from pruning

A problem

- Reverse path forwarding requires a router to know shortest path to a source
  - known from routing table
- Doesn't work if some routers do not support multicast
  - *virtual links* between multicast-capable routers
  - shortest path to A from E is not C, but F

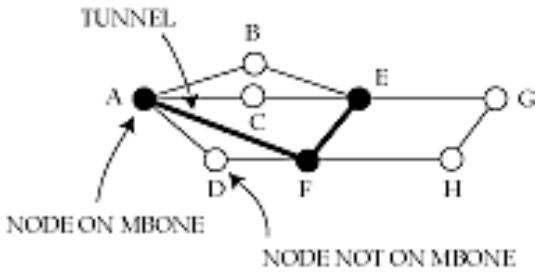


A problem (contd.)

- Two problems
  - how to build virtual links
  - how to construct routing table for a network with virtual links

Tunnels

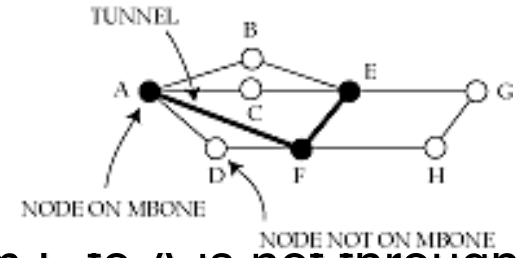
- Why do we need them?



- Consider packet sent from A to F via multicast-incapable D
- If packet's destination is Class D, D drops it
- If destination is F's address, F doesn't know multicast address!
- So, put packet destination as F, but carry multicast address internally
- Encapsulate IP in IP => set protocol type to IP-in-IP

Multicast routing protocol

- Interface on “shortest path” to source depends on whether path is real or virtual



- Shortest path from E to A is not through C, but F
  - so packets from F will be flooded, but not from C
- Need to discover shortest paths only taking multicast-capable routers into account
  - DVMRP

DVMRP

- Distance-vector Multicast routing protocol
- Very similar to RIP
  - distance vector
  - hop count metric
- Used in conjunction with
  - flood-and-prune (to determine memberships)
    - prunes store per-source and per-group information
  - reverse-path forwarding (to decide where to forward a packet)
  - explicit join messages to reduce join latency (but no source info, so still need flooding)

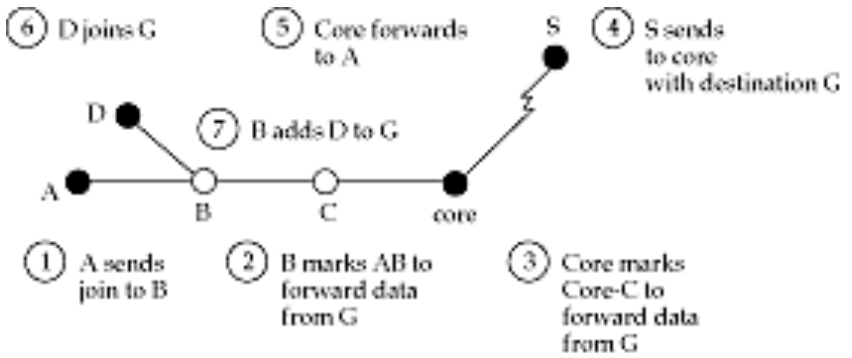
MOSPF

- Multicast extension to OSPF
- Routers flood group membership information with LSPs
- Each router independently computes shortest-path tree that only includes multicast-capable routers
  - no need to flood and prune
- Complex
  - interactions with external and summary records
  - need storage per group per link
  - need to compute shortest path tree per source and group

Core-based trees

- Problems with DVMRP-oriented approach
  - need to periodically flood and prune to determine group members
  - need to source per-source and per-group prune records at each router
- Key idea with core-based tree
  - coordinate multicast with a *core* router
  - host sends a join request to core router
  - routers along path mark incoming interface for forwarding

Example

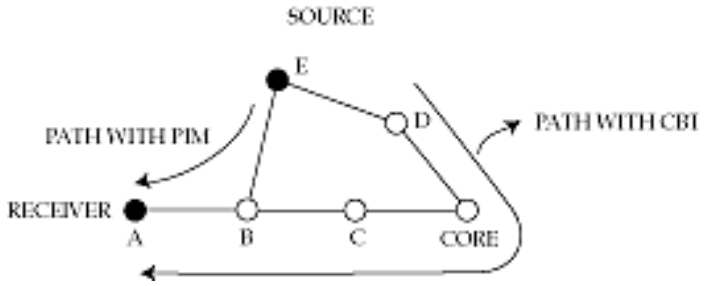


- Pros
  - routers not part of a group are not involved in pruning
  - explicit join/leave makes membership changes faster
  - router needs to store only one record per group
- Cons
  - all multicast traffic traverses core, which is a bottleneck
  - traffic travels on non-optimal paths

Protocol independent multicast (PIM)

- Tries to bring together best aspects of CBT and DVMRP
- Choose different strategies depending on whether multicast tree is *dense* or *sparse*
  - flood and prune good for dense groups
    - only need a few prunes
    - CBT needs explicit join per source/group
  - CBT good for sparse groups
- Dense mode PIM == DVMRP
- Sparse mode PIM is similar to CBT
  - but receivers can switch from CBT to a shortest-path tree

PIM (contd.)



- In CBT, E must send to core
- In PIM, B discovers shorter path to E (by looking at unicast routing table)
  - sends join message directly to E
  - sends prune message towards core
- Core no longer bottleneck
- Survives failure of core

More on core

- Renamed a *rendezvous point*
  - because it no longer carries all the traffic like a CBT core
- Rendezvous points periodically send “I am alive” messages downstream
- Leaf routers set timer on receipt
- If timer goes off, send a join request to alternative rendezvous point
- Problems
  - how to decide whether to use dense or sparse mode?
  - how to determine “best” rendezvous point?

Outline

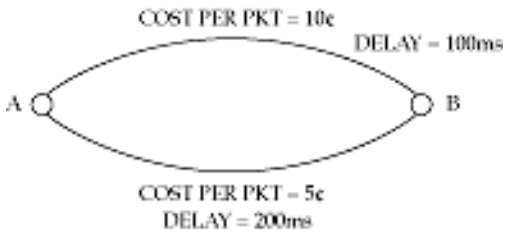
- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

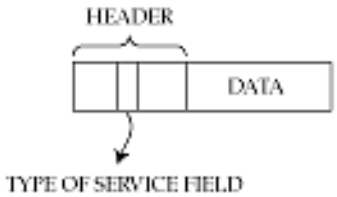
Routing vs. policy routing

- In standard routing, a packet is forwarded on the ‘best’ path to destination
  - choice depends on load and link status
- With policy routing, routes are chosen depending on *policy* directives regarding things like
  - source and destination address
  - transit domains
  - quality of service
  - time of day
  - charging and accounting
- The general problem is still open
  - fine balance between correctness and information hiding

Multiple metrics

- Simplest approach to policy routing
- Advertise multiple costs per link
- Routers construct multiple shortest path trees





Problems with multiple metrics

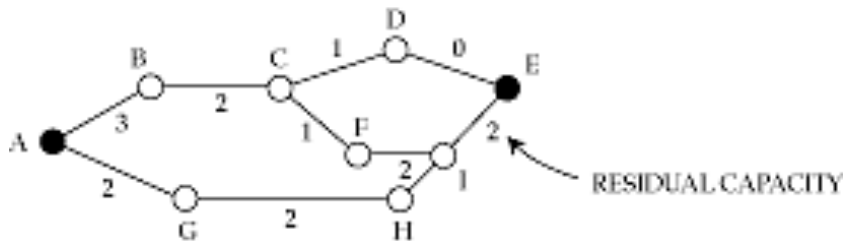
- All routers must use the same rule in computing paths
- Remote routers may misinterpret policy
  - source routing may solve this
  - but introduces other problems (what?)

Provider selection

- Another simple approach
- Assume that a single service provider provides almost all the path from source to destination
  - e.g. AT&T or MCI
- Then, choose policy simply by choosing provider
  - this could be dynamic (agents!)
- In Internet, can use a loose source route through service provider's access point
- Or, multiple addresses/names per host

Crankback

- Consider computing routes with QoS guarantees
- Router returns packet if no next hop with sufficient QoS can be found
- In ATM networks (PNNI) used for the call-setup packet
- In Internet, may need to be done for _every_ packet!
  - Will it work?



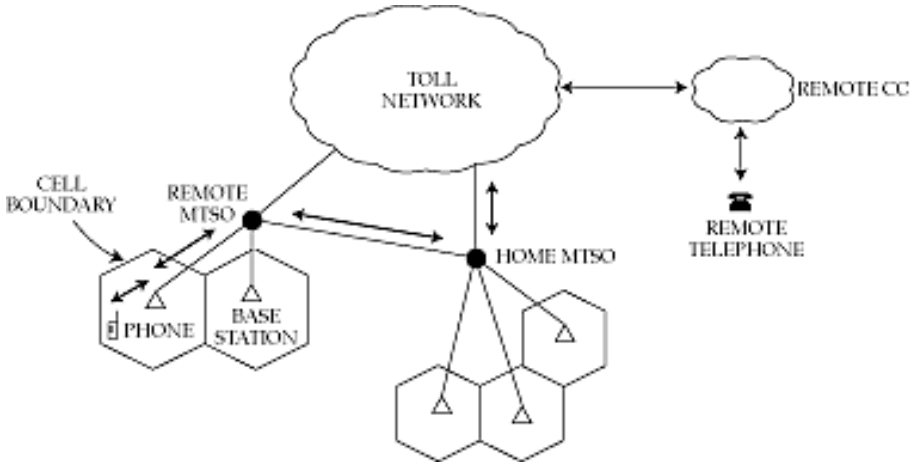
Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Mobile routing

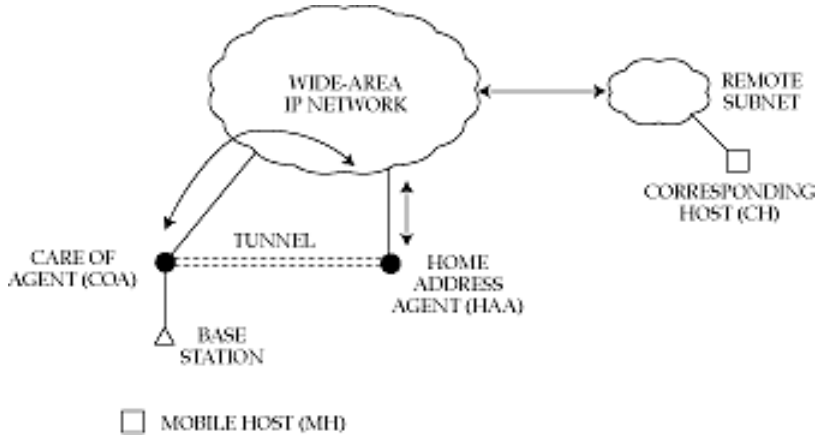
- How to find a mobile host?
- Two sub-problems
  - location (where is the host?)
  - routing (how to get packets to it?)
- We will study mobile routing in the Internet and in the telephone network

Mobile routing in the telephone network



- Each cell phone has a global ID that it tells remote MTSO when turned on (using slotted ALOHA up channel)
- Remote MTSO tells home MTSO
- *To* phone: call forwarded to remote MTSO to closest base
- *From* phone: call forwarded to home MTSO from closest base
- New MTSOs can be added as load increases

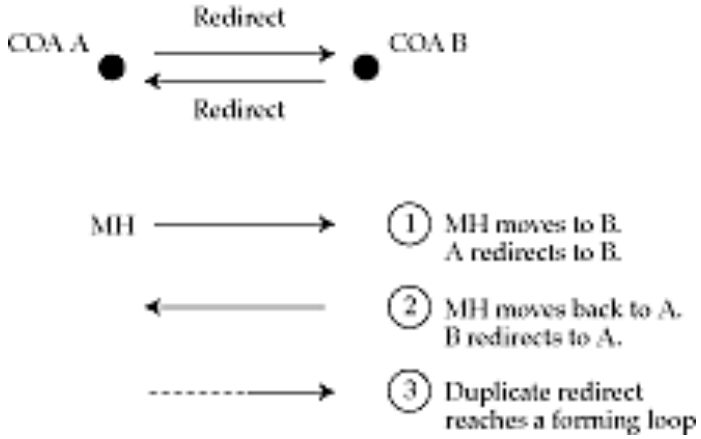
Mobile routing in the Internet



- Very similar to mobile telephony
  - but outgoing traffic does not go through home
  - and need to use tunnels to forward data
- Use *registration* packets instead of slotted ALOHA
  - passed on to home address agent
- Old care-of-agent forwards packets to new care-of-agent until home address agent learns of change

Problems

- Security
  - mobile and home address agent share a common secret
  - checked before forwarding packets to COA
- Loops



Error control

An Engineering Approach to Computer Networking

CRC

- Detects
  - all single bit errors
  - almost all 2-bit errors
  - any odd number of errors
  - all bursts up to M, where generator length is M
  - longer bursts with probability 2^-m

Implementation

- Hardware
  - on-the-fly with a shift register
  - easy to implement with ASIC/FPGA
- Software
  - precompute remainders for 16-bit words
  - add remainders to a running sum
  - needs only one lookup per 16-bit block

Software schemes

- Efficiency is important
  - touch each data byte only once
- CRC
- TCP/UDP/IP
  - all use same scheme
  - treat data bytes as 16-bit integers
  - add with end-around carry
  - one’s complement = checksum
  - catches all 1-bit errors
  - longer errors with prob 1/65536

Packet errors

- Different from bit errors
  - types
    - not just erasure, but also duplication, insertion,etc.
  - correction
    - retransmission, instead of redundancy

Types of packet errors

- Loss
  - due to uncorrectable bit errors
  - buffer loss on overflow
    - especially with bursty traffic
      - for the same load, the greater the burstiness, the more the loss
    - loss rate depends on burstiness, load, and buffer size
  - fragmented packets can lead to error multiplication
    - longer the packet, more the loss

Types of packet errors (cont.)

- Duplication
  - same packet received twice
    - usually due to retransmission
- Insertion
  - packet from some other conversation received
    - header corruption
- Reordering
  - packets received in wrong order
    - usually due to retransmission
    - some routers also reorder

Packet error detection and correction

- Detection
  - Sequence numbers
  - Timeouts
- Correction
  - Retransmission

Sequence numbers

- In each header
- Incremented for non-retransmitted packets
- *Sequence space*
  - set of all possible sequence numbers
  - for a 3-bit seq #, space is {0,1,2,3,4,5,6,7}

Using sequence numbers

- Loss
  - gap in sequence space allows *receiver* to detect loss
    - e.g. received 0,1,2,5,6,7 => lost 3,4
  - acks carry *cumulative* seq #
  - redundant information
  - if no ack for a while, *sender* suspects loss
- Reordering
- Duplication
- Insertion
  - if the received seq # is “very different” from what is expected
    - more on this later

Sequence number size

- Long enough so that sender does not confuse sequence numbers on acks
- E.g, sending at < 100 packets/sec (R)
  - wait for 200 secs before giving up (T)
  - receiver may dally up to 100 sec (A)
  - packet can live in the network up to 5 minutes (300 s) (*maximum packet lifetime)*
  - can get an ack as late as 900 seconds after packet sent out
  - sent out 900*100 = 90,000 packets
  - if seqence space smaller, then can have confusion
  - so, sequence number > log (90,000), at least 17 bits
- In general 2^seq_size > R(2 MPL + T + A)

MPL

- How can we bound it?
- Generation time in header
  - too complex!
- Counter in header decremented per hop
  - crufty, but works
  - used in the Internet
  - assumes max. diameter, and a limit on forwarding time

Sequence number size (cont.)

- If no acks, then size depends on two things
  - reordering span: how much packets can be reordered
    - e.g. span of 128 => seq # > 7 bits
  - burst loss span: how many consecutive pkts. can be lost
    - e.g. possibility of 16 consecutive lost packets => seq # > 4 bits
  - In practice, hope that technology becomes obselete before worst case hits!

Packet insertion

- Receiver should be able to distinguish packets from other connections
- Why?
  - receive packets on VCI 1
  - connection closes
  - new connection also with VCI 1
  - delayed packet arrives
  - could be accepted
- Solution
  - flush packets on connection clos
  - can’t do this for connectionless networks like the Internet

Packet insertion in the Internet

- Packets carry source IP, dest IP, *source port number, destination port number*
- How we can have insertion?
  - host A opens connection to B, source port 123, dest port 456
  - transport layer connection terminates
  - new connection opens, A and B assign the same port numbers
  - delayed packet from old connection arrives
  - insertion!

Solutions

- Per-connection *incarnation number*
  - incremented for each connection from each host
  - - takes up header space
  - - on a crash, we may repeat
    - need stable storage, which is expensive
- Reassign port numbers only after 1 MPL
  - - needs stable storage to survive crash

Solutions (cont.)

- Assign port numbers serially: new connections have new ports
  - Unix starts at 1024
  - this fails if we wrap around within 1 MPL
  - also fails of computer crashes and we restart with 1024
- Assign initial sequence numbers serially
  - new connections may have same port, but seq # differs
  - fails on a crash
- Wait 1 MPL after boot up (30s to 2 min)
  - this flushes old packets from network
  - used in most Unix systems

3-way handshake

- Standard solution, then, is
  - choose port numbers serially
  - choose initial sequence numbers from a clock
  - wait 1 MPL after a crash
- Needs communicating ends to tell each other initial sequence number
- Easiest way is to tell this in a *SYNchronize* packet (TCP) that starts a connection
- 2-way handshake

3-way handshake

- Problem really is that SYNs themselves are not protected with sequence numbers
- 3-way handshake protects against delayed SYNs

Loss detection

- At receiver, from a gap in sequence space
  - send a *nack* to the sender
- At sender, by looking at cumulative acks, and timeing out if no ack for a while
  - need to choose timeout interval

Nacks

- Sounds good, but does not work well
  - extra load during loss, even though in reverse direction
- If nack is lost, receiver must retransmit it
  - moves timeout problem to receiver
- So we need timeouts anyway

Timeouts

- Set timer on sending a packet
- If timer goes off, and no ack, resend
- How to choose timeout value?
- Intuition is that we expect a reply in about one round trip time (RTT)

Timeout schemes

- Static scheme
  - know RTT *a priori*
  - timer set to this value
  - works well when RTT changes little
- Dynamic scheme
  - measure RTT
  - timeout is a function of measured RTTs

Old TCP scheme

- RTTs are measured periodically
- Smoothed RTT (*srtt*)
- *srtt = a * srtt + (1-a) * RTT*
- timeout = *b * srtt*
- a = 0.9, b = 2
- sensitive to choice of a
  - a = 1 => timeout = 2 * initial srtt
  - a = 0 => no history
- doesn't work too well in practice

New TCP scheme (Jacobson)

- introduce new term = mean deviation from mean (m)
- *m = | srtt - RTT |*
- *sm = a * sm + (1-a) * m*
- *timeout = srtt + b * sm*

Intrinsic problems

- Hard to choose proper timers, even with new TCP scheme
  - What should initial value of srtt be?
  - High variability in R
  - Timeout => loss, delayed ack, or lost ack
    - hard to distinguish
- Lesson: use timeouts rarely

Retransmissions

- Sender detects loss on timeout
- Which packets to retransmit?
- Need to first understand concept of error control window

Error control window

- Set of packets sent, but not acked
- 1 2 3 4 5 6 7 8 9 (original window)
- 1 2 3 4 5 6 7 8 9 (recv ack for 3)
- 1 2 3 4 5 6 7 8 9 (send 8)
- May want to restrict max size = window size
- Sender blocked until ack comes back

Go back N retransmission

- On a timeout, retransmit the entire error control window
- Receiver only accepts in-order packets
- + simple
- + no buffer at receiver
- - can add to congestion
- - wastes bandwidth
- used in TCP
- if packet loss rate is *p*, and

Selective retransmission

- Somehow find out which packets lost, then only retransmit them
- How to find lost packets?
  - each ack has a bitmap of received packets
    - e.g. cum_ack = 5, bitmap = 101 => received 5 and 7, but not 6
    - wastes header space
  - sender periodically asks receiver for bitmap
  - fast retransmit

Fast retransmit

- Assume cumulative acks
- If sender sees repeated cumulative acks, packet likely lost
- 1, 2, 3, 4, 5 , 6
- 1, 2, 3 3 3
- Send cumulative_ack + 1 = 4
- Used in TCP

SMART

- Ack carries cumulative sequence number
- Also sequence number of packet causing ack
- 1 2 3 4 5 6 7
- 1 2 3 3 3 3
- 1 2 3 5 6 7
- Sender creates bitmap
- No need for timers!
- If retransmitted packet lost, periodically check if cumulative ack increased.

Flow Control

An Engineering Approach to Computer Networking

Flow control problem

- Consider file transfer
- Sender sends a stream of packets representing fragments of a file
- Sender should try to match rate at which receiver and network can process data
- Can’t send too slow or too fast
- Too slow
  - wastes time
- Too fast
  - can lead to buffer overflow
- How to find the correct rate?

Other considerations

- Simplicity
- Overhead
- Scaling
- Fairness
- Stability

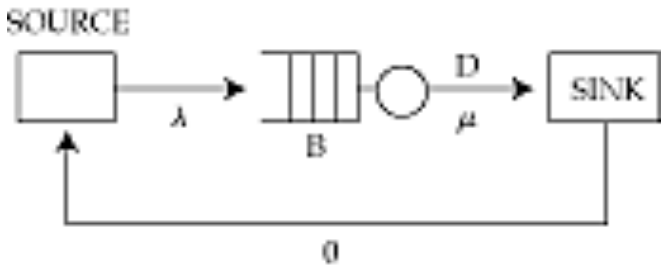
- Many interesting tradeoffs
  - overhead for stability
  - simplicity for unfairness

Where?

- Usually at transport layer
- Also, in some cases, in datalink layer

Model

- Source, sink, server, service rate, bottleneck, round trip time



Classification

- Open loop
  - Source describes its desired flow rate
  - Network *admits* call
  - Source sends at this rate
- Closed loop
  - Source monitors available service rate
    - Explicit or implicit
  - Sends at this rate
  - Due to speed of light delay, errors are bound to occur
- Hybrid
  - Source asks for some minimum rate
  - But can send more, if available

Open loop flow control

- Two phases to flow
  - Call setup
  - Data transmission
- Call setup
  - Network prescribes parameters
  - User chooses parameter values
  - Network admits or denies call
- Data transmission
  - User sends within parameter range
  - Network *polices* users
  - Scheduling policies give user QoS

Hard problems

- Choosing a descriptor at a source
- Choosing a scheduling discipline at intermediate network elements
- Admitting calls so that their performance objectives are met (*call admission control*).

Traffic descriptors

- Usually an *envelope*
  - Constrains worst case behavior
- Three uses
  - Basis for traffic contract
  - Input to *regulator*
  - Input to *policer*

Descriptor requirements

- Representativity
  - adequately describes flow, so that network does not reserve too little or too much resource
- Verifiability
  - verify that descriptor holds
- Preservability
  - Doesn’t change inside the network
- Usability
  - Easy to describe and use for admission control

Examples

- Representative, verifiable, but not useble
  - Time series of interarrival times
- Verifiable, preservable, and useable, but not representative
  - peak rate

Some common descriptors

- Peak rate
- Average rate
- Linear bounded arrival process

Peak rate

- Highest ‘rate’ at which a source can send data
- Two ways to compute it
- For networks with fixed-size packets
  - min inter-packet spacing
- For networks with variable-size packets
  - highest rate over *all* intervals of a particular duration
- Regulator for fixed-size packets
  - timer set on packet transmission
  - if timer expires, send packet, if any
- Problem
  - sensitive to extremes

Average rate

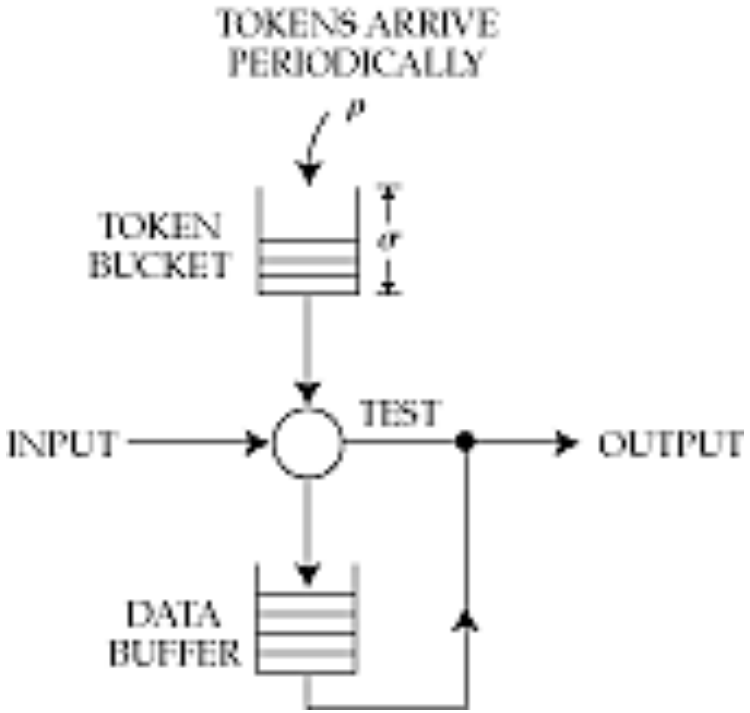
- Rate over some time period (*window*)
- Less susceptible to outliers
- Parameters: *t* and *a*
- Two types: jumping window and moving window
- Jumping window
  - over consecutive intervals of length *t*, only *a* bits sent
  - regulator reinitializes every interval
- Moving window
  - over all intervals of length *t,* only *a* bits sent
  - regulator forgets packet sent more than *t* seconds ago

Linear Bounded Arrival Process

- Source bounds # bits sent in any time interval by a linear function of time
- the number of bits transmitted in any active interval of length t is less than $rt + s$
- r is the long term rate
- s is the burst limit
- insensitive to outliers

Leaky bucket

- A regulator for an LBAP
- Token bucket fills up at rate *r*
- Largest # tokens < *s*



Variants

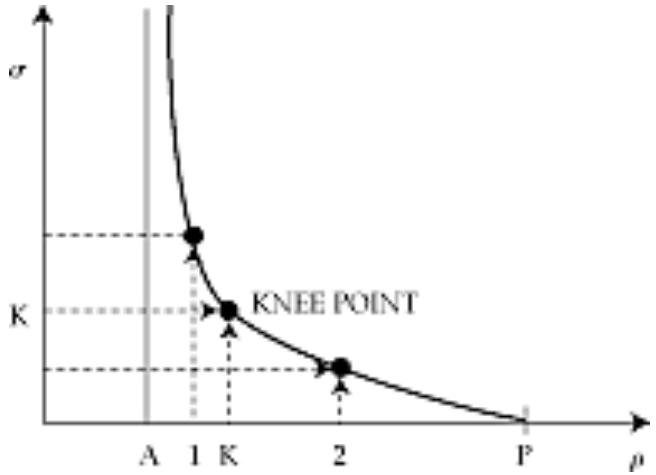
- Token and data buckets
  - Sum is what matters
- Peak rate regulator

Choosing LBAP parameters

- Tradeoff between *r* and *s*
- Minimal descriptor
  - doesn't simultaneously have smaller *r* and *s*
  - presumably costs less
- How to choose minimal descriptor?
- Three way tradeoff
  - choice of *s* (data bucket size)
  - loss rate
  - choice of *r*

Choosing minimal parameters

- Keeping loss rate the same
  - if *s* is more, *r* is less (smoothing)
  - for each *r* we have least *s*
- Choose knee of curve



LBAP

- Popular in practice and in academia
  - sort of representative
  - verifiable
  - sort of preservable
  - sort of usable
- Problems with multiple time scale traffic
  - large burst messes up things

Open loop vs. closed loop

- Open loop
  - describe traffic
  - network admits/reserves resources
  - regulation/policing
- Closed loop
  - can't describe traffic or network doesn't support reservation
  - monitor available bandwidth
    - perhaps allocated using GPS-emulation
  - adapt to it
  - if not done properly either
    - too much loss
    - unnecessary delay

Taxonomy

- First generation
  - ignores network state
  - only match receiver
- Second generation
  - responsive to state
  - three choices
    - State measurement
      - explicit or implicit
    - Control
      - flow control window size or rate
    - Point of control
      - endpoint or within network

Explicit vs. Implicit

- Explicit
  - Network tells source its current rate
  - Better control
  - More overhead
- Implicit
  - Endpoint figures out rate by looking at network
  - Less overhead
- Ideally, want overhead of implicit with effectiveness of explicit

Flow control window

- Recall error control window
- Largest number of packet outstanding (sent but not acked)
- If endpoint has sent all packets in window, it must wait => slows down its rate
- Thus, window provides *both* error control and flow control
- This is called *transmission* window
- Coupling can be a problem
  - Few buffers are receiver => slow rate!

Window vs. rate

- In adaptive rate, we directly control rate
- Needs a timer per connection
- Plusses for window
  - no need for fine-grained timer
  - self-limiting
- Plusses for rate
  - better control (finer grain)
  - no coupling of flow control and error control
- Rate control must be careful to avoid overhead and sending too much

Hop-by-hop vs. end-to-end

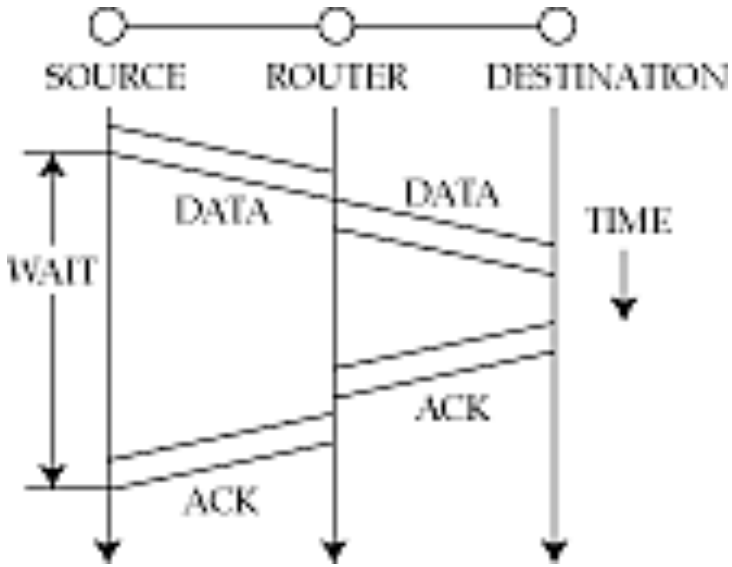
- Hop-by-hop
  - first generation flow control at each link
    - next server = sink
  - easy to implement
- End-to-end
  - sender matches all the servers on its path
- Plusses for hop-by-hop
  - simpler
  - distributes overflow
  - better control
- Plusses for end-to-end
  - cheaper

On-off

- Receiver gives ON and OFF signals
- If ON, send at full speed
- If OFF, stop
- OK when RTT is small
- What if OFF is lost?
- Bursty
- Used in serial lines or LANs

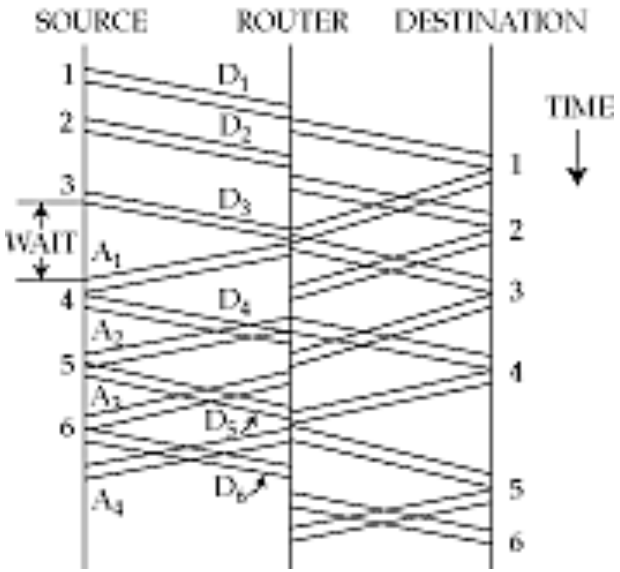
Stop and Wait

- Send a packet
- Wait for ack before sending next packet



Static window

- Stop and wait can send at most one pkt per RTT
- Here, we allow multiple packets per RTT (= transmission window)



What should window size be?

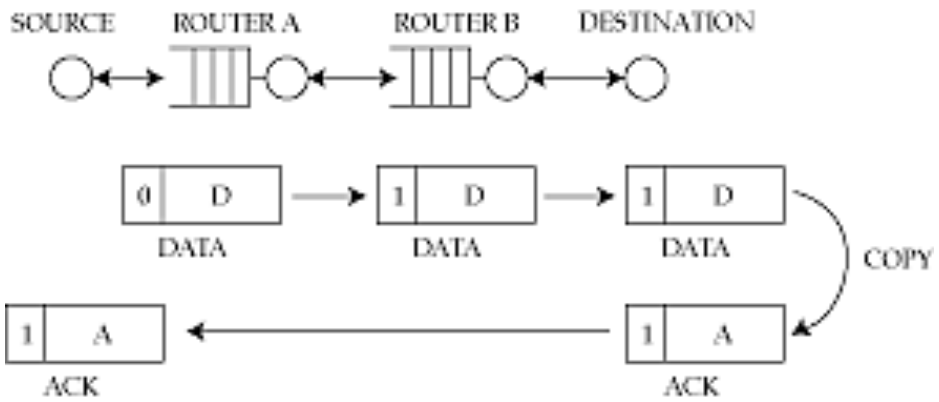
- Let bottleneck service rate along path = b pkts/sec
- Let round trip time = R sec
- Let flow control window = w packet
- Sending rate is w packets in R seconds = w/R
- To use bottleneck w/R > b => w > bR
- This is the *bandwidth delay product* or *optimal window size*

Static window

- Works well if b and R are fixed
- But, bottleneck rate changes with time!
- Static choice of w can lead to problems
  - too small
  - too large
- So, need to adapt window
- Always try to get to the *current* optimal value

DECbit flow control

- Intuition
  - every packet has a bit in header
  - intermediate routers set bit if queue has built up => source window is too large
  - sink copies bit to ack
  - if bits set, source reduces window size
  - in steady state, oscillate around optimal size

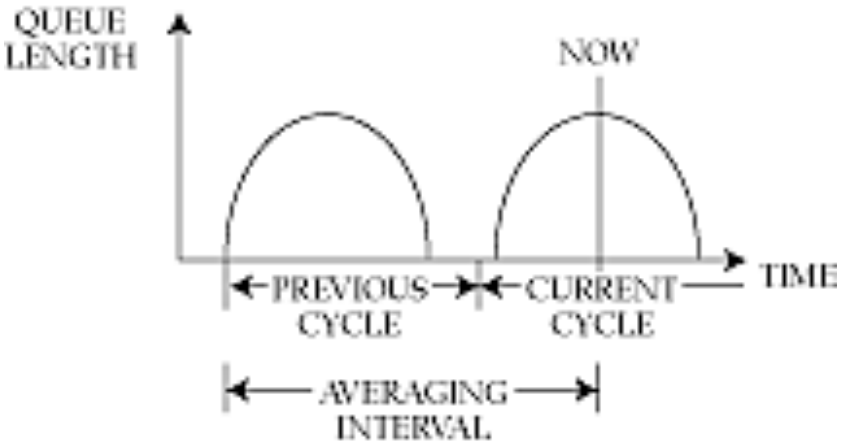


DECbit

- When do bits get set?
- How does a source interpret them?

DECbit details: router actions

- Measure *demand* and *mean queue length* of each source
- Computed over queue regeneration cycles
- Balance between sensitivity and stability



Router actions

- If mean queue length > 1.0
  - set bits on sources whose demand exceeds fair share
- If it exceeds 2.0
  - set bits on everyone
  - panic!

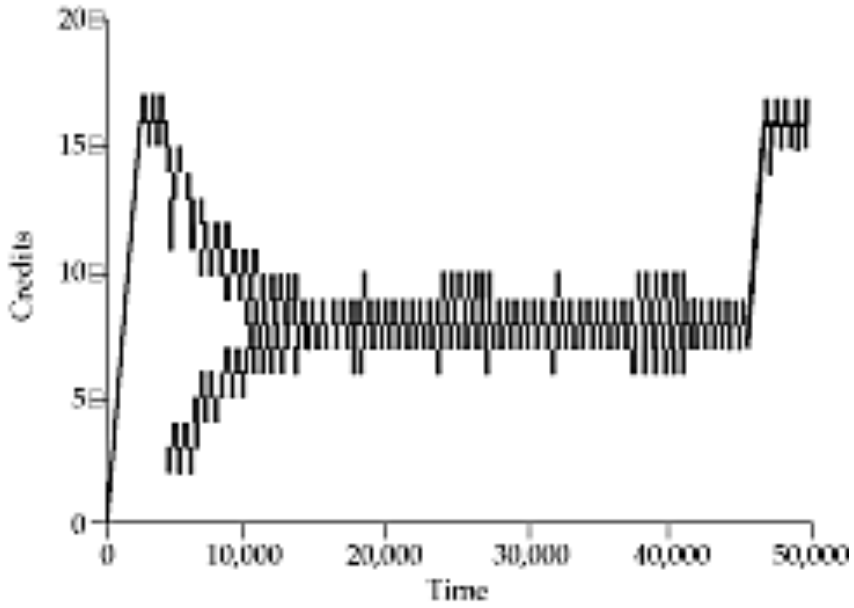
Source actions

- Keep track of bits
- Can’t take control actions too fast!
- Wait for past change to take effect
- Measure bits over past + present window size
- If more than 50% set, then decrease window, else increase
- Additive increase, multiplicative decrease

Evaluation

- Works with FIFO
  - but requires per-connection state (demand)
- Software
- But
  - assumes cooperation!
  - conservative window increase policy

Sample trace



TCP Flow Control

- Implicit
- Dynamic window
- End-to-end
- Very similar to DECbit, but
  - no support from routers
  - increase if no loss (usually detected using timeout)
  - window decrease on a timeout
  - additive increase multiplicative decrease

TCP details

- Window starts at 1
- Increases exponentially for a while, then linearly
- Exponentially => doubles every RTT
- Linearly => increases by 1 every RTT
- During exponential phase, every ack results in window increase by 1
- During linear phase, window increases by 1 when # acks = window size
- Exponential phase is called *slow start*
- Linear phase is called *congestion avoidance*

More TCP details

- On a loss, current window size is stored in a variable called *slow start threshold* or *ssthresh*
- Switch from exponential to linear (slow start to congestion avoidance) when window size reaches threshold
- Loss detected either with timeout or *fast retransmit* (duplicate cumulative acks)
- Two versions of TCP
  - Tahoe: in both cases, drop window to 1
  - Reno: on timeout, drop window to 1, and on fast retransmit drop window to half previous size (also, increase window on subsequent acks)

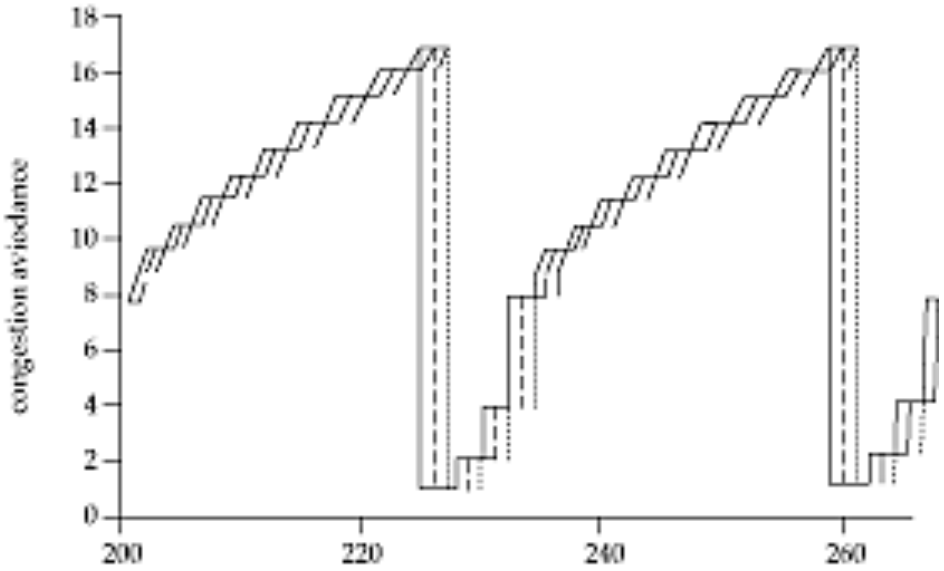
TCP vs. DECbit

- Both use dynamic window flow control and additive-increase multiplicative decrease
- TCP uses implicit measurement of congestion
  - probe a black box
- Operates at the *cliff*
- Source does not filter information

Evaluation

- Effective over a wide range of bandwidths
- A lot of operational experience
- Weaknesses
  - loss => overload? (wireless)
  - overload => self-blame, problem with FCFS
  - ovelroad detected only on a loss
    - in steady state, source *induces* loss
  - needs at least bR/3 buffers per connection

Sample trace



TCP Vegas

- Expected throughput = transmission_window_size/propagation_delay
- Numerator: known
- Denominator: measure *smallest* RTT
- Also know *actual* throughput
- Difference = how much to reduce/increase rate
- Algorithm
  - send a special packet
  - on ack, compute expected and actual throughput
  - (expected - actual)* RTT packets in bottleneck buffer
  - adjust sending rate if this is too large
- Works better than TCP Reno

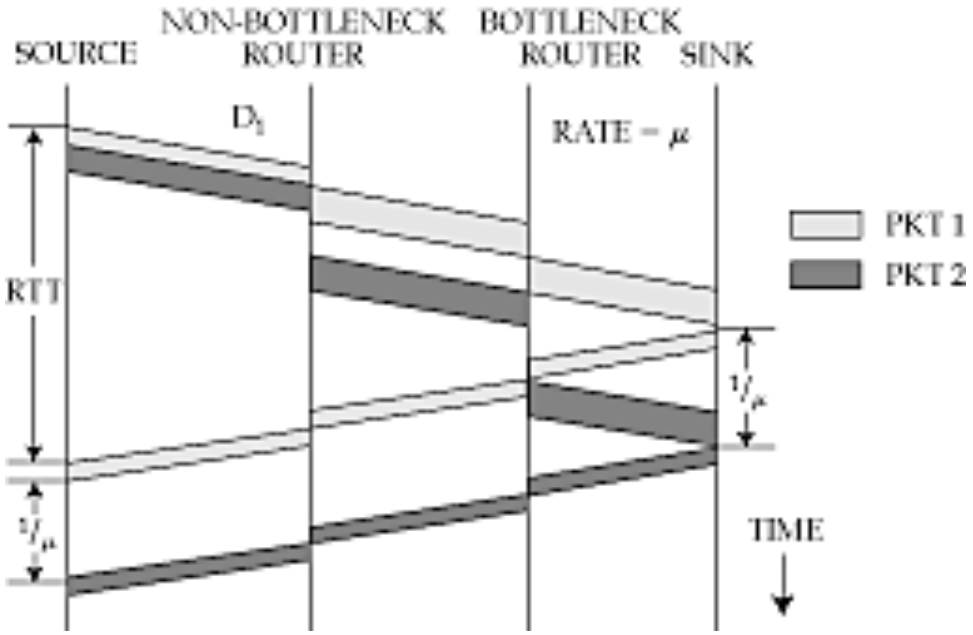
NETBLT

- First rate-based flow control scheme
- Separates error control (window) and flow control (no *coupling*)
- So, losses and retransmissions do not affect the flow rate
- Application data sent as a series of buffers, each at a particular rate
- Rate = (burst size + burst rate) so granularity of control = burst
- Initially, no adjustment of rates
- Later, if received rate < sending rate, multiplicatively decrease rate
- Change rate only once per buffer => slow

Packet pair

- Improves basic ideas in NETBLT
  - better measurement of bottleneck
  - control based on prediction
  - finer granularity
- Assume all bottlenecks serve packets in round robin order
- Then, spacing between packets at receiver (= ack spacing) = 1/(rate of slowest server)
- If *all* data sent as paired packets, no distinction between data and probes
- Implicitly determine service rates if servers are round-robin-like

Packet pair


SOURCE
NON-BOTTLENECK ROUTER
BOTTLENECK ROUTER
SINK
D_1
RATE = μ
PKT 1
PKT 2
RTT
$1/\mu$
$1/\mu$
TIME

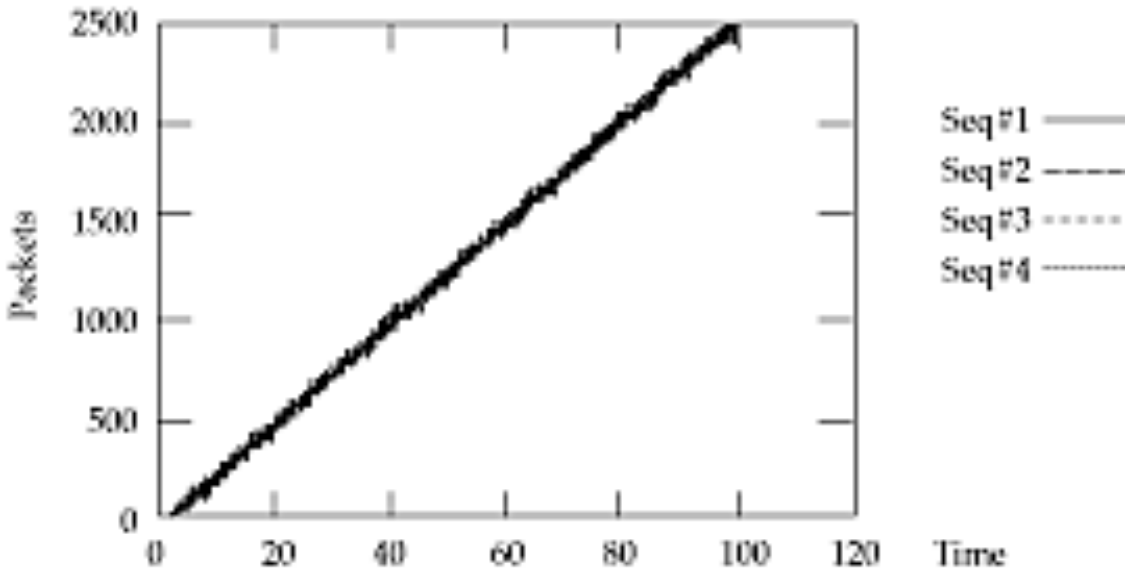
Packet-pair details

- Acks give time series of service rates in the past
- We can use this to predict the next rate
- Exponential averager, with fuzzy rules to change the averaging factor
- Predicted rate feeds into flow control equation

Packet-pair flow control

- Let X = # packets in bottleneck buffer
- S = # outstanding packets
- R = RTT
- b = bottleneck rate
- Then, X = S - Rb (assuming no losses)
- Let l = source rate
- l(k+1) = b(k+1) + (setpoint -X)/R

Sample trace



ATM Forum EERC

- Similar to DECbit, but send a whole cell's worth of info instead of one bit
- Sources periodically send a Resource Management (RM) cell with a *rate request*
  - typically once every 32 cells
- Each server fills in RM cell with current share, if less
- Source sends at this rate

ATM Forum EERC details

- Source sends Explicit Rate (ER) in RM cell
- Switches compute source share in an unspecified manner (allows competition)
- Current rate = allowed cell rate = ACR
- If ER > ACR then ACR = ACR + RIF * PCR else ACR = ER
- If switch does not change ER, then use DECbit idea
  - If CI bit set, ACR = ACR (1 - RDF)
- If ER < AR, AR = ER
- Allows interoperability of a sort
- If idle 500 ms, reset rate to Initial cell rate
- If no RM cells return for a while, ACR *= (1-RDF)

Comparison with DECbit

- Sources know exact rate
- Non-zero Initial cell-rate => conservative increase can be avoided
- Interoperation between ER/CI switches

Problems

- RM cells in data path a mess
- Updating sending rate based on RM cell can be hard
- Interoperability comes at the cost of reduced efficiency (as bad as DECbit)
- Computing ER is hard

Comparison among closed-loop schemes

- On-off, stop-and-wait, static window, DECbit, TCP, NETBLT, Packet-pair, ATM Forum EERC
- Which is best? No simple answer
- Some rules of thumb
  - flow control easier with RR scheduling
    - otherwise, assume cooperation, or police rates
  - explicit schemes are more robust
  - hop-by-hop schemes are more resposive, but more comples
  - try to separate error control and flow control
  - rate based schemes are inherently unstable unless well-engineered

Hybrid flow control

- Source gets a minimum rate, but can use more
- All problems of both open loop and closed loop flow control
- Resource partitioning problem
  - what fraction can be reserved?
  - how?

Multiple Access

An Engineering Approach to Computer Networking

What is it all about?

- Consider an audioconference where
  - if one person speaks, all can hear
  - if more than one person speaks at the same time, both voices are garbled
- How should participants coordinate actions so that
  - the number of messages exchanged per second is maximized
  - time spent waiting for a chance to speak is minimized
- This is the *multiple access problem*

Some simple solutions

- Use a moderator
  - a speaker must wait for moderator to call on him or her, even if no one else wants to speak
  - what if the moderator's connection breaks?
- Distributed solution
  - speak if no one else is speaking
  - but if two speakers are waiting for a third to finish, guarantee collision
- Designing good schemes is surprisingly hard!

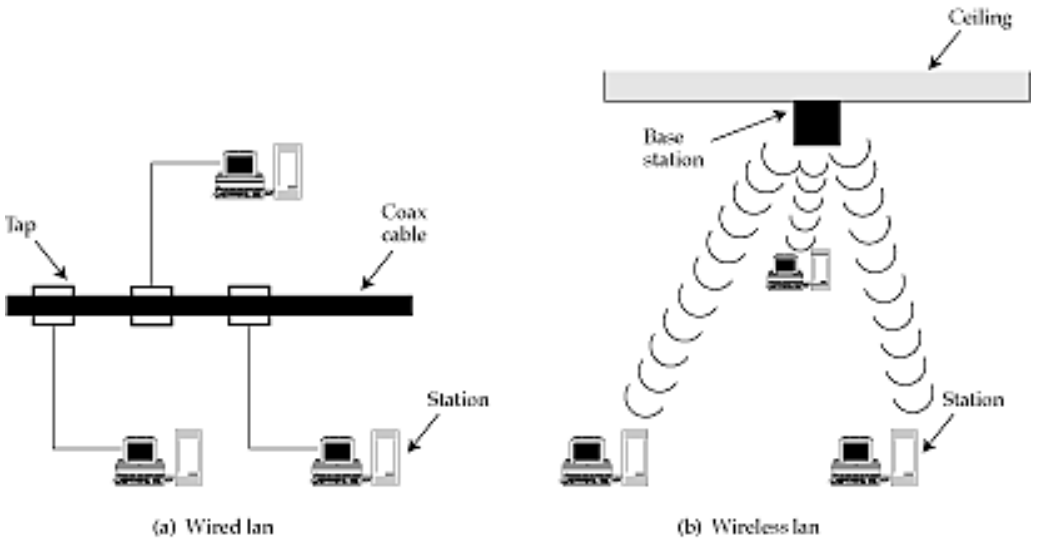
Outline

- Contexts for the problem
- Choices and constraints
- Performance metrics
- Base technologies
- Centralized schemes
- Distributed schemes

Contexts for the multiple access problem

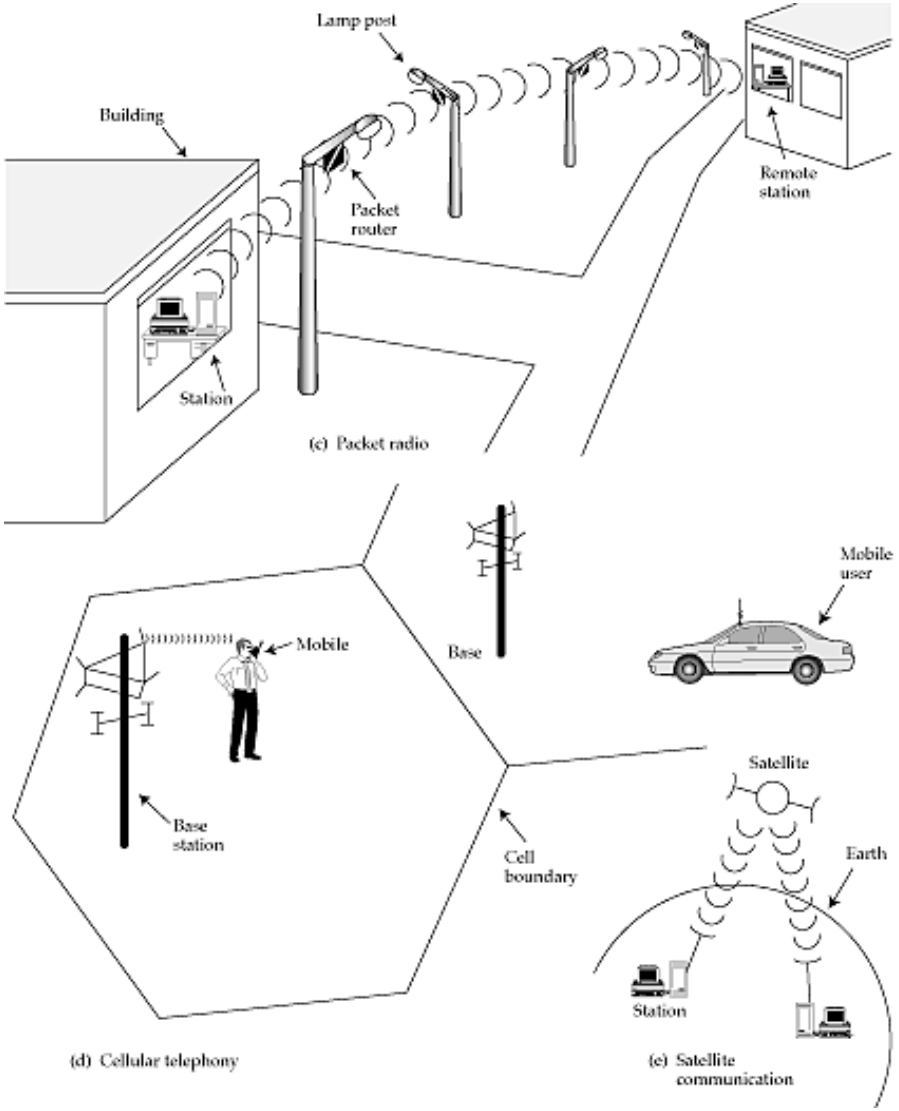
- *Broadcast* transmission medium
  - message from any transmitter is received by all receivers
- Colliding messages are garbled
- Goal
  - maximize message throughput
  - minimize mean waiting time
- Shows up in five main contexts

Contexts



(a) Wired lan (b) Wireless lan

Contexts


Lamp post
Building
Packet router
Remote station
Station
(c) Packet radio
Mobile user
Base
Mobile
Base station
Cell boundary
Satellite
Earth
Station
(d) Cellular telephony
(e) Satellite communication

Solving the problem

- First, choose a *base technology*
  - to isolate traffic from different stations
  - can be in time domain or frequency domain
- Then, choose how to allocate a limited number of transmission resources to a larger set of contending users

Outline

- Contexts for the problem
- Choices and constraints
- Performance metrics
- Base technologies
- Centralized schemes
- Distributed schemes

Choices

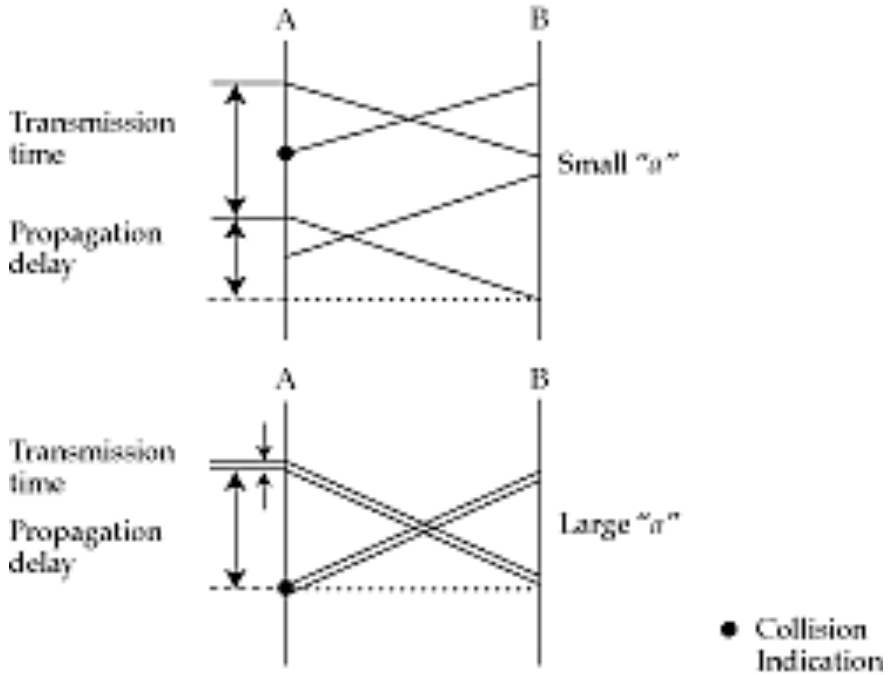
- Centralized vs. distributed design
  - is there a moderator or not?
  - in a centralized solution one of the stations is a *master* and the others are *slaves*
    - master->slave = downlink
    - slave->master = uplink
  - in a distributed solution, all stations are peers
- Circuit-mode vs. packet-mode
  - do stations send steady streams or bursts of packets?
  - with streams, doesn't make sense to contend for every packet
  - allocate resources to streams
  - with packets, makes sense to contend for every packet to avoid wasting bandwidth

Constraints

- Spectrum scarcity
  - radio spectrum is hard to come by
  - only a few frequencies available for long-distance communication
  - multiple access schemes must be careful not to waste bandwidth
- Radio link properties
  - radio links are error prone
    - fading
    - multipath interference
  - hidden terminals
    - transmitter heard only by a subset of receivers
  - capture
    - on collision, station with higher power overpowers the other
    - lower powered station may never get a chance to be heard

The parameter ‘a’

- The number of packets sent by a source before the farthest station receives the first bit



Outline

- Contexts for the problem
- Choices and constraints
- Performance metrics
- Base technologies
- Centralized schemes
- Distributed schemes

Performance metrics

- Normalized throughput
  - fraction of link capacity used to carry non-retransmitted packets
  - example
    - with no collisions, 1000 packets/sec
    - with a particular scheme and workload, 250 packets/sec
    - => goodput = 0.25
- Mean delay
  - amount of time a station has to wait before it successfully transmits a packet
    - depends on the load and the characteristics of the medium

Performance metrics

- Stability
  - with heavy load, is all the time spent on resolving contentions?
  - => unstable
  - with a stable algorithm, throughput does not decrease with offered load
  - if infinite number of uncontrolled stations share a link, then instability is guaranteed
  - but if sources reduce load when overload is detected, can achieve stability
- Fairness
  - no single definition
  - ‘no-starvation’: source eventually gets a chance to send
  - max-min fair share: will study later

Outline

- Contexts for the problem
- Choices and constraints
- Performance metrics
- Base technologies
- Centralized schemes
- Distributed schemes

Base technologies

- Isolates data from different sources
- Three basic choices
  - Frequency division multiple access (FDMA)
  - Time division multiple access (TDMA)
  - Code division multiple access (CDMA)

FDMA

- Simplest
- Best suited for analog links
- Each station has its own frequency band, separated by guard bands
- Receivers tune to the right frequency
- Number of frequencies is limited
  - reduce transmitter power; reuse frequencies in non-adjacent cells
  - example: voice channel = 30 KHz
  - 833 channels in 25 MHz band
  - with hexagonal cells, partition into 118 channels each
  - but with N cells in a city, can get 118N calls => win if N > 7

TDMA

- All stations transmit data on same frequency, but at different times
- Needs time synchronization
- Pros
  - users can be given different amounts of bandwidth
  - mobiles can use idle times to determine best base station
  - can switch off power when not transmitting
- Cons
  - synchronization overhead
  - greater problems with multipath interference on wireless links

CDMA

- Users separated both by time and frequency
- Send at a different frequency at each time slot (*frequency hopping*)
- Or, convert a single bit to a code (*direct sequence*)
  - receiver can decipher bit by inverse process
- Pros
  - hard to spy
  - immune from narrowband noise
  - no need for all stations to synchronize
  - no hard limit on capacity of a cell
  - all cells can use all frequencies

CDMA

- Cons
  - implementation complexity
  - need for power control
    - to avoid capture
  - need for a large contiguous frequency band (for direct sequence)
  - problems installing in the field

FDD and TDD

- Two ways of converting a wireless medium to a duplex channel
- In Frequency Division Duplex, uplink and downlink use different frequencies
- In Time Division Duplex, uplink and downlink use different time slots
- Can combine with FDMA/TDMA
- Examples
  - TDD/FDMA in second-generation cordless phones
  - FDD/TDMA/FDMA in digital cellular phones

Outline

- Contexts for the problem
- Choices and constraints
- Performance metrics
- Base technologies
- Centralized schemes
- Distributed schemes

Centralized access schemes

- One station is master, and the other are slaves
  - slave can transmit only when master allows
- Natural fit in some situations
  - wireless LAN, where base station is the only station that can see everyone
  - cellular telephony, where base station is the only one capable of high transmit power

Centralized access schemes

- Pros
  - simple
  - master provides single point of coordination
- Cons
  - master is a single point of failure
    - need a re-election protocol
    - master is involved in every single transfer => added delay

Circuit mode

- When station wants to transmit, it sends a message to master using packet mode
- Master allocates transmission resources to slave
- Slave uses the resources until it is done
- No contention during data transfer
- Used primarily in cellular phone systems
  - EAMPS: FDMA
  - GSM/IS-54: TDMA
  - IS-95: CDMA

Polling and probing

- Centralized packet-mode multiple access schemes
- Polling
  - master asks each station in turn if it wants to send (roll-call polling)
  - inefficient if only a few stations are active, overhead for polling messages is high, or system has many terminals
- Probing
  - stations are numbered with consecutive logical addresses
  - assume station can listen both to its own address and to a set of multicast addresses
  - master does a binary search to locate next active station

Reservation-based schemes

- When ‘a’ is large, can’t use a distributed scheme for packet mode (too many collisions)
  - mainly for satellite links
- Instead master coordinates access to link using reservations
- Some time slots devoted to reservation messages
  - can be smaller than data slots => *minislots*
- Stations contend for a minislot (or own one)
- Master decides winners and grants them access to link
- Packet collisions are only for minislots, so overhead on contention is reduced

Outline

- Contexts for the problem
- Choices and constraints
- Performance metrics
- Base technologies
- Centralized schemes
- Distributed schemes

Distributed schemes

- Compared to a centralized scheme
  - more reliable
  - have lower message delays
  - often allow higher network utilization
  - but are more complicated
- Almost all distributed schemes are packet mode (why?)

Decentralized polling

- Just like centralized polling, except there is no master
- Each station is assigned a slot that it uses
  - if nothing to send, slot is wasted
- Also, all stations must share a time base

Decentralized probing

- Also called *tree based multiple access*
- All stations in left subtree of root place packet on medium
- If a collision, root <- root ->left_son, and try again
- On success, everyone in root->right_son places a packet etc.
- (If two nodes with successive logical addresses have a packet to send, how many collisions will it take for one of them to win access?)
- Works poorly with many active stations, or when all active stations are in the same subtree

Carrier Sense Multiple Access (CSMA)

- A fundamental advance: check whether the medium is active before sending a packet (i.e *carrier sensing*)
- Unlike polling/probing a node with something to send doesn't have to wait for a master, or for its turn in a schedule
- If medium idle, then can send
- If collision happens, detect and resolve
- Works when 'a' is small

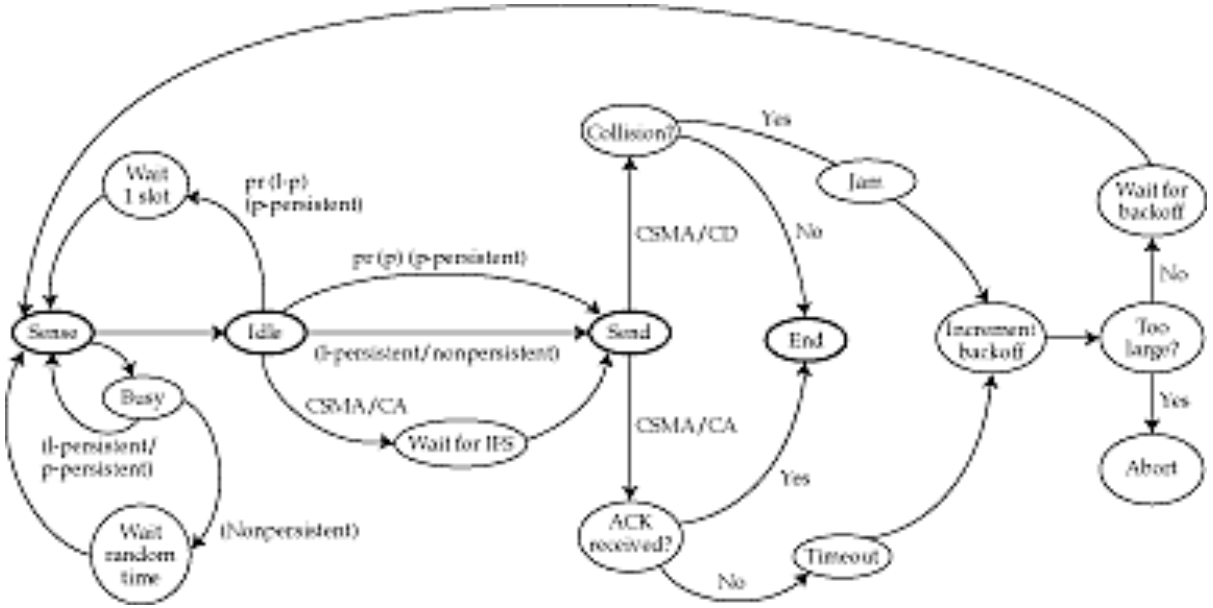
Simplest CSMA scheme

- Send a packet as soon as medium becomes idle
- If, on sensing busy, wait for idle -> *persistent*
- If, on sensing busy, set a timer and try later -> *non-persistent*
- Problem with persistent: two stations waiting to speak will collide

How to solve the collision problem

- Two solutions
- *p-persistent*: on idle, transmit with probability *p:*
  - hard to choose *p*
  - if *p* small, then wasted time
  - if *p* large, more collisions
- *exponential backoff*
  - on collision, choose timeout randomly from doubled range
  - backoff range adapts to number of contending stations
  - no need to choose *p*
  - need to detect collisions: *collision detect circuit* => CSMA/CD

Summary of CSMA schemes



Ethernet

- The most widely used LAN
- Standard is called IEEE 802.3
- Uses CSMA/CD with exponential backoff
- Also, on collision, place a *jam* signal on wire, so that all stations are aware of collision and can increment timeout range
- ‘a’ small =>time wasted in collision is around 50 microseconds
- Ethernet requires packet to be long enough that a collision is detected before packet transmission completes (a <= 1)
  - packet should be at least 64 bytes long for longest allowed segment
- Max packet size is 1500 bytes
  - prevents hogging by a single station

More on Ethernet

- First version ran at 3 Mbps and used ‘thick’ coax
- These days, runs at 10 Mbps, and uses ‘thin’ coax, or twisted pair (Category 3 and Category 5)
- Ethernet types are coded as <Speed><Baseband or broadband><physical medium>
  - Speed = 3, 10, 100 Mbps
  - Baseband = within building, broadband = on cable TV
  - Physical medium:
    - “2” is cheap 50 Ohm cable, upto 185 meters
    - “T” is unshielded twisted pair (also used for telephone wiring)
    - “36” is 75 Ohm cable TV cable, upto 3600 meters

Recent developments

- Switched Ethernet
  - each station is connected to switch by a separate UTP wire
  - line card of switch has a buffer to hold incoming packets
  - fast backplane switches packet from one line card to others
  - simultaneously arriving packets do not collide (until buffers overflow)
  - higher intrinsic capacity than 10BaseT (and more expensive)

Fast Ethernet variants

- Fast Ethernet (IEEE 802.3u)
  - same as 10BaseT, except that line speed is 100 Mbps
  - spans only 205 m
  - big winner
  - most current cards support both 10 and 100 Mbps cards (10/100 cards) for about $80
- 100VG Anylan (IEEE 802.12)
  - station makes explicit service requests to master
  - master schedules requests, eliminating collisions
  - not a success in the market
- Gigabit Ethernet
  - aims to continue the trend
  - still undefined, but first implementation will be based on fiber links

Evaluating Ethernet

- Pros
  - easy to setup
  - requires no configuration
  - robust to noise
- Problems
  - at heavy loads, users see large delays because of backoff
  - nondeterministic service
  - doesn't support priorities
  - big overhead on small packets
- But, very successful because
  - problems only at high load
  - can segment LANs to reduce load

CSMA/CA

- Used in wireless LANs
- Can't detect collision because transmitter overwhelms colocated receiver
- So, need explicit acks
- But this makes collisions more expensive
  - => try to reduce number of collisions

CSMA/CA algorithm

- First check if medium is busy
- If so, wait for medium to become idle
- Wait for interframe spacing
- Set a *contention timer* to an interval randomly chosen in the range [1, CW]
- On timeout, send packet and wait for ack
- If no ack, assume packet is lost
  - try again, after doubling CW
- If another station transmits while counting down, freeze CW and unfreeze when packet completes transmission
- (Why does this scheme reduce collisions compared to CSMA/CD?)

Dealing with hidden terminals

- CSMA/CA works when every station can receive transmissions from every other station
- Not always true
- Hidden terminal
  - some stations in an area cannot hear transmissions from others, though base can hear both
- Exposed terminal
  - some (but not all) stations can hear transmissions from stations not in the local area

Dealing with hidden and exposed terminals

- In both cases, CSMA/CA doesn't work
  - with hidden terminal, collision because carrier not detected
  - with exposed terminal, idle station because carrier incorrectly detected
- Two solutions
- Busy Tone Multiple Access (BTMA)
  - uses a separate "busy-tone" channel
  - when station is receiving a message, it places a tone on this channel
  - everyone who might want to talk to a station knows that it is busy
    - even if they cannot hear transmission that that station hears
  - this avoids both problems (why?)

Multiple Access Collision Avoidance

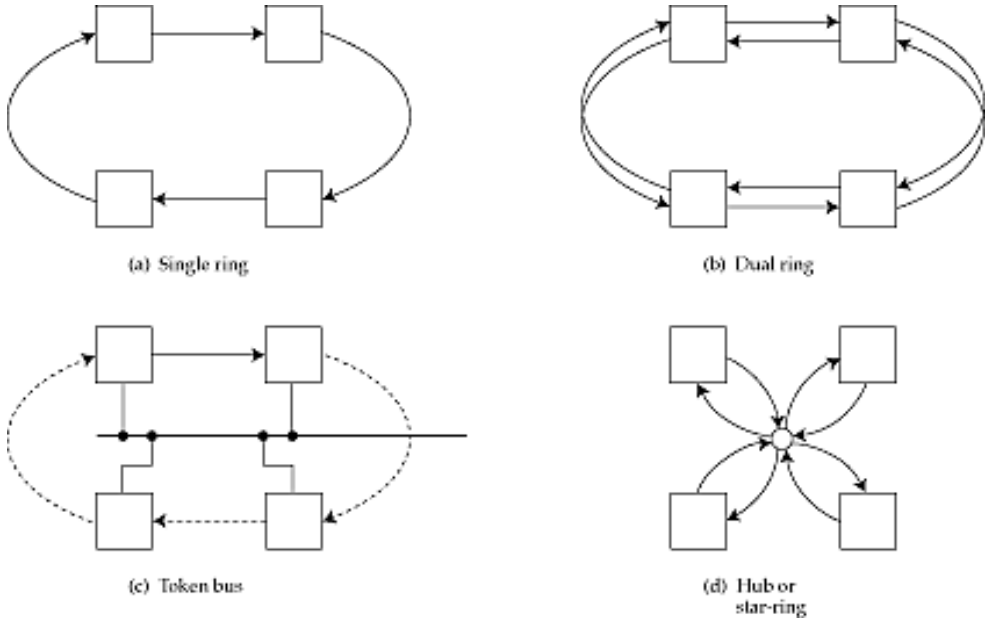
- BTMA requires us to split frequency band
  - more complex receivers (need two tuners)
- Separate bands may have different propagation characteristics
  - scheme fails!
- Instead, use a single frequency band, but use explicit messages to tell others that receiver is busy
- In MACA, before sending data, send a Request to Sent (RTS) to intended receiver
- Station, if idle, sends Clear to Send (CTS)
- Sender then sends data
- If station overhears RTS, it waits for other transmission to end
- (why does this work?)

Token passing

- In distributed polling, every station has to wait for its turn
- Time wasted because idle stations are still given a slot
- What if we can quickly skip past idle stations?
- This is the key idea of token ring
- Special packet called ‘token’ gives station the right to transmit data
- When done, it passes token to ‘next’ station
  - => stations form a logical ring
- No station will starve

Logical rings

- Can be on a non-ring physical topology



Ring operation

- During normal operation, copy packets from input buffer to output
- If packet is a token, check if packets ready to send
- If not, forward token
- If so, delete token, and send packets
- Receiver copies packet and sets ‘ack’ flag
- Sender removes packet and deletes it
- When done, reinserts token
- If ring idle and no token for a long time, regenerate token

Single and double rings

- With a single ring, a single failure of a link or station breaks the network => fragile
- With a double ring, on a failure, go into *wrap mode*
- Used in FDDI

Hub or star-ring

- Simplifies wiring
- Active hub is predecessor and successor to every station
  - can monitor ring for station and link failures
- Passive hub only serves as wiring concentrator
  - but provides a single test point
- Because of these benefits, hubs are practically the only form of wiring used in real networks
  - even for Ethernet

Evaluating token ring

- Pros
  - medium access protocol is simple and explicit
  - no need for carrier sensing, time synchronization or complex protocols to resolve contention
  - guarantees zero collisions
  - can give some stations priority over others
- Cons
  - token is a single point of failure
    - lost or corrupted token trashes network
    - need to carefully protect and, if necessary, regenerate token
  - all stations must cooperate
    - network must detect and cut off unresponsive stations
  - stations must actively monitor network
    - usually elect one station as monitor

Fiber Distributed Data Interface

- FDDI is the most popular token-ring base LAN
- Dual counterrotating rings, each at 100 Mbps
- Uses both copper and fiber links
- Supports both non-realtime and realtime traffic
  - token is guaranteed to rotate once every Target Token Rotation Time (TTRT)
  - station is guaranteed a *synchronous allocation* within every TTRT
- Supports both *single attached* and *dual attached* stations
  - single attached (cheaper) stations are connected to only one of the rings

ALOHA and its variants

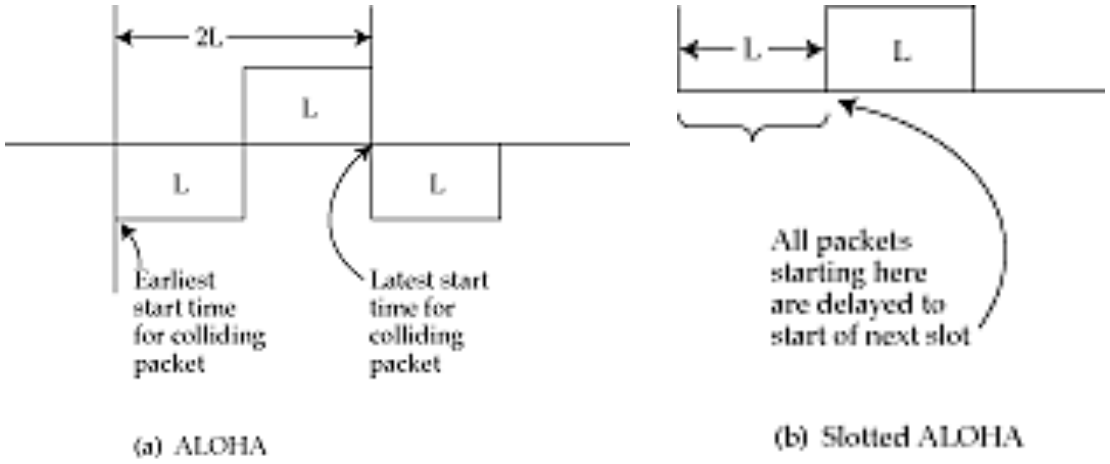
- ALOHA is one of the earliest multiple access schemes
- Just send it!
- Wait for an ack
- If no ack, try again after a random waiting time
  - no backoff

Evaluating ALOHA

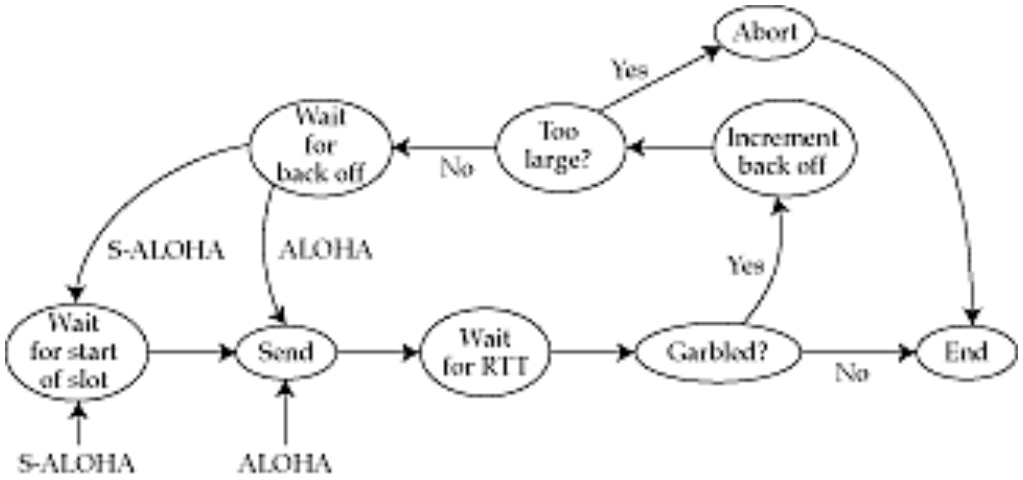
- Pros
  - useful when ‘a’ is large, so carrier sensing doesn’t help
    - satellite links
  - simple
    - no carrier sensing, no token, no timebase synchronization
  - independent of ‘a’
- Cons
  - under some mathematical assumptions, goodput is at most .18
  - at high loads, collisions are very frequent
  - sudden burst of traffic can lead to instability
    - unless backoff is exponential

Slotted ALOHA

- A simple way to double ALOHA's capacity
- Make sure transmissions start on a slot boundary
- Halves *window of vulnerability*
- Used in cellular phone uplink



ALOHA schemes summarized



Reservation ALOHA

- Combines slot reservation with slotted ALOHA
- Contend for reservation minislots using slotted ALOHA
- Stations independently examine reservation requests and come to consistent conclusions
- Simplest version
  - divide time into frames = fixed length set of slots
  - station that wins access to a reservation minislot using S-ALOHA can keep slot as long as it wants
  - station that loses keeps track of idle slots and contends for them in next frame

Evaluating R-ALOHA

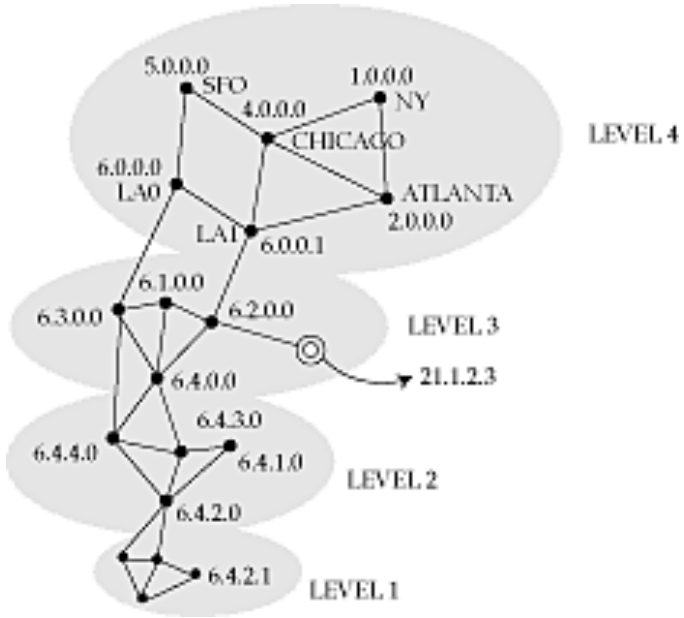
- Pros
  - supports both circuit and packet mode transfer
  - works with large ‘a’
  - simple
- Cons
  - arriving packet has to wait for entire frame before it has a chance to send
  - cannot preempt hogs
  - variants of R-ALOHA avoid these problems
- Used for cable-modem uplinks

Switching

An Engineering Approach to Computer Networking

What is it all about?

- How do we move traffic from one part of the network to another?
- Connect end-systems to switches, and switches to each other
- Data arriving to an input port of a switch have to be moved to one or more of the output ports



Types of switching elements

- Telephone switches
  - switch samples
- Datagram routers
  - switch datagrams
- ATM switches
  - switch ATM cells

Classification

- Packet vs. circuit switches
  - packets have headers and samples don't
- Connectionless vs. connection oriented
  - connection oriented switches need a call setup
  - setup is handled in *control plane* by *switch controller*
  - connectionless switches deal with *self-contained* datagrams

	Connectionless (router)	*Connection-oriented (switching system)*
Packet switch	Internet router	ATM switching system
Circuit switch		Telephone switching system

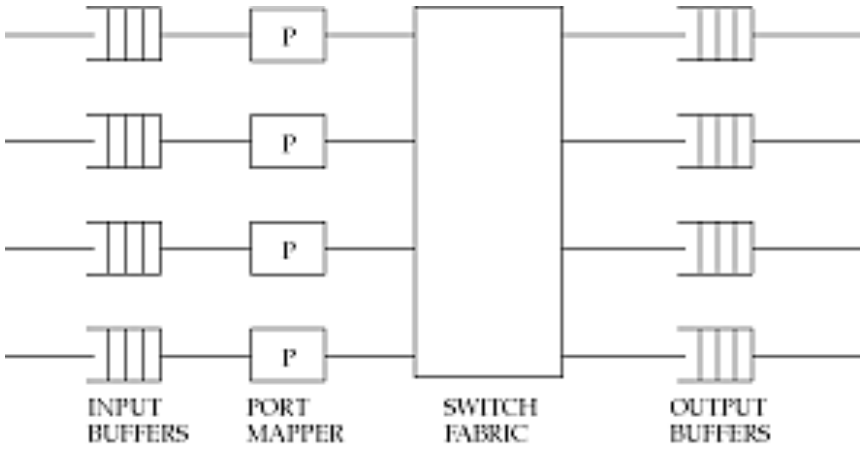
Other switching element functions

- Participate in routing algorithms
  - to build routing tables
- Resolve contention for output trunks
  - scheduling
- Admission control
  - to guarantee resources to certain streams
- We'll discuss these later
- Here we focus on pure data movement

Requirements

- Capacity of switch is the maximum rate at which it can move information, assuming all data paths are simultaneously active
- Primary goal: maximize capacity
  - subject to cost and reliability constraints
- Circuit switch must reject call if can't find a path for samples from input to output
  - goal: minimize call blocking
- Packet switch must reject a packet if it can't find a buffer to store it awaiting access to output trunk
  - goal: minimize packet loss
- Don't reorder packets

A generic switch



Outline

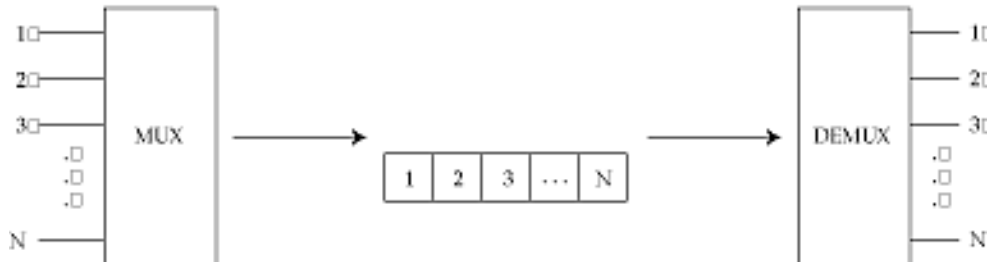
- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Circuit switching

- Moving 8-bit samples from an input port to an output port
- Recall that samples have no headers
- Destination of sample depends on *time* at which it arrives at the switch
  - actually, relative order within a *frame*
- We'll first study something simpler than a switch: a multiplexor

Multiplexors and demultiplexors

- Most trunks time division multiplex voice samples
- At a central office, trunk is demultiplexed and distributed to active circuits
- Synchronous multiplexor
  - N input lines
  - Output runs N times as fast as input



More on multiplexing

- Demultiplexor
  - one input line and N outputs that run N times slower
  - samples are placed in output buffer in round robin order
- Neither multiplexor nor demultiplexor needs addressing information (why?)
- Can cascade multiplexors
  - need a standard
  - example: DS hierarchy in the US and Japan

Inverse multiplexing

- Takes a high bit-rate stream and scatters it across multiple trunks
- At the other end, combines multiple streams
  - resequencing to accommodate variation in delays
- Allows high-speed virtual links using existing technology

A circuit switch

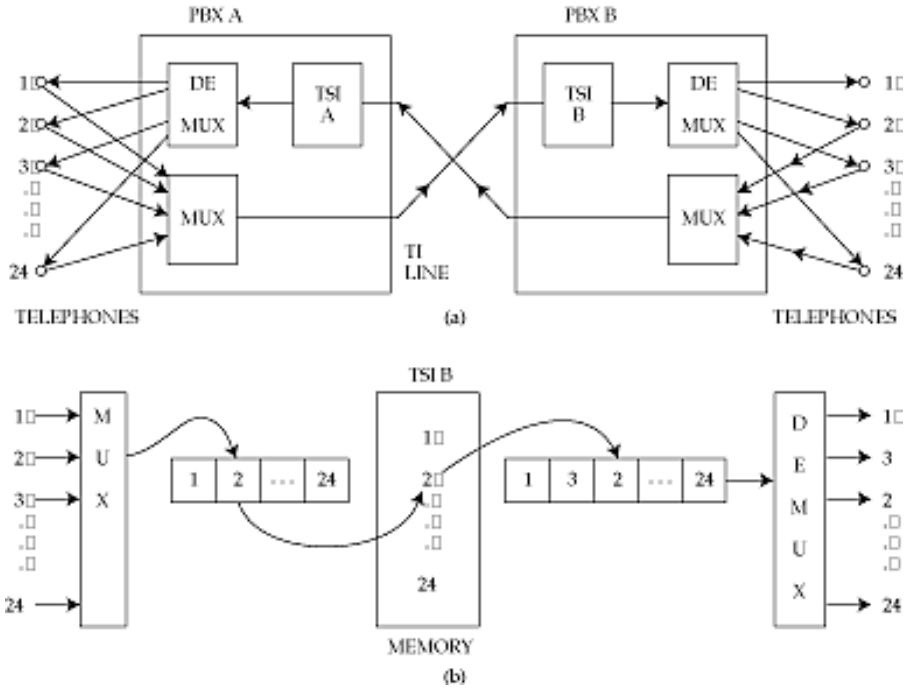
- A switch that can handle N calls has N logical inputs and N logical outputs
  - N up to 200,000
- In practice, input trunks are multiplexed
  - example: DS3 trunk carries 672 simultaneous calls
- Multiplexed trunks carry *frames* = set of samples
- Goal: extract samples from frame, and depending on position in frame, switch to output
  - each incoming sample has to get to the right output line and the right slot in the output frame
  - demultiplex, switch, multiplex

Call blocking

- Can’t find a path from input to output
- Internal blocking
  - slot in output frame exists, but no path
- Output blocking
  - no slot in output frame is available
- Output blocking is reduced in *transit* switches
  - need to put a sample in one of *several* slots going to the desired next hop

Time division switching

- Key idea: when demultiplexing, position in frame determines output trunk
- Time division switching interchanges sample position within a frame: time slot interchange (TSI)

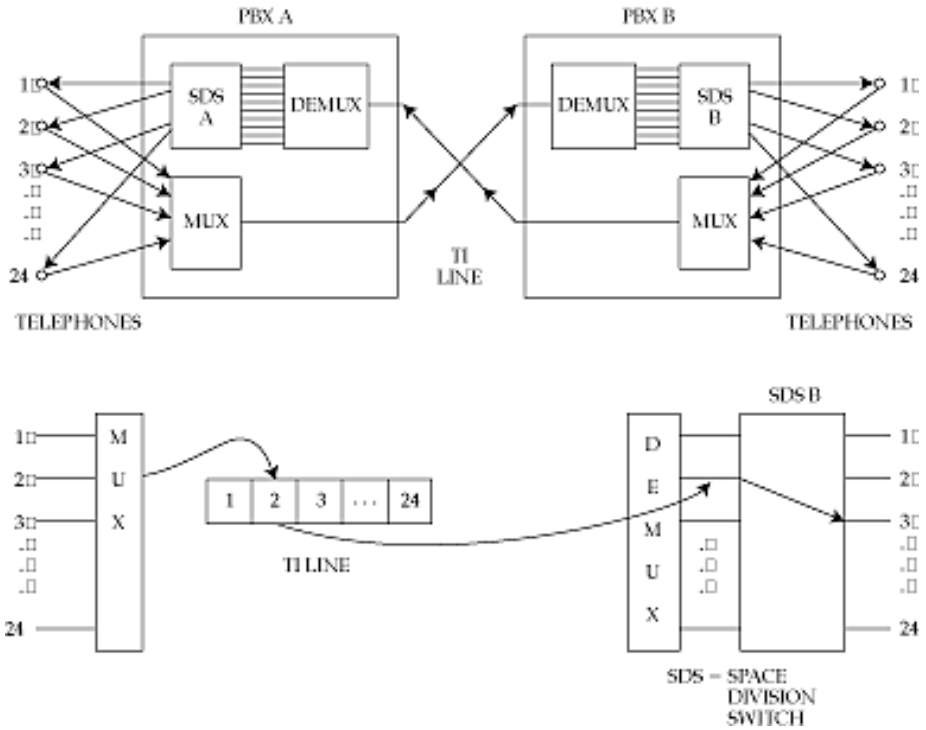


How large a TSI can we build?

- Limit is time taken to read and write to memory
- For 120,000 circuits
  - need to read and write memory once every 125 microseconds
  - each operation takes around 0.5 ns => impossible with current technology
- Need to look to other techniques

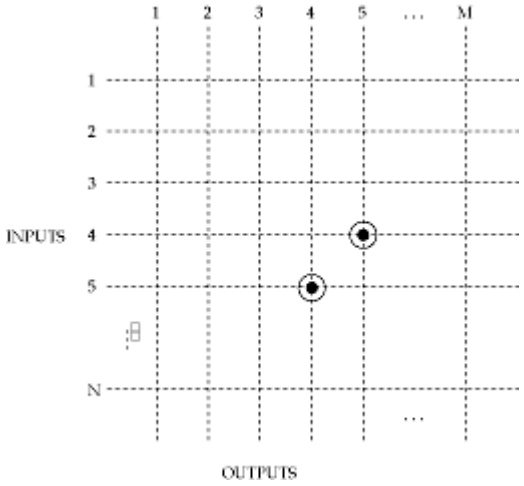
Space division switching

- Each sample takes a different path thoguh the swithc, depending on its destination



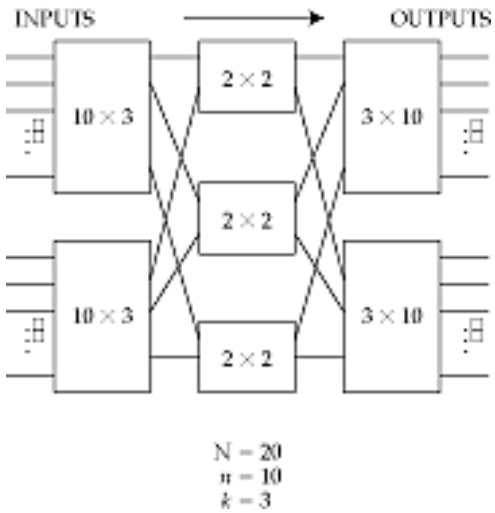
Crossbar

- Simplest possible space-division switch
- *Crosspoints* can be turned on or off
- For multiplexed inputs, need a switching *schedule* (why?)
- Internally nonblocking
  - but need N^2 crosspoints
  - time taken to set each crosspoint grows quadratically
  - vulnerable to single faults (why?)



Multistage crossbar

- In a crossbar during each switching time only one crosspoint per row or column is active
- Can save crosspoints if a crosspoint can attach to more than one input line (why?)
- This is done in a multistage crossbar
- Need to rearrange connections every switching time

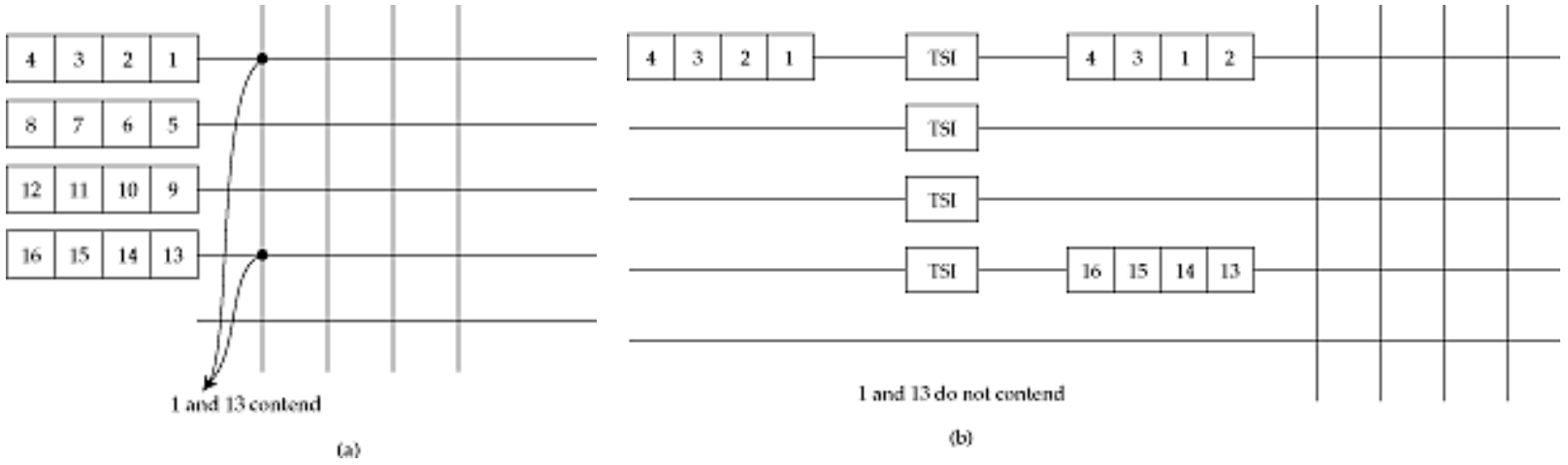


Multistage crossbar

- Can suffer internal blocking
  - unless sufficient number of second-level stages
- Number of crosspoints < N^2
- Finding a path from input to output requires a depth-first-search
- Scales better than crossbar, but still not too well
  - 120,000 call switch needs ~250 million crosspoints

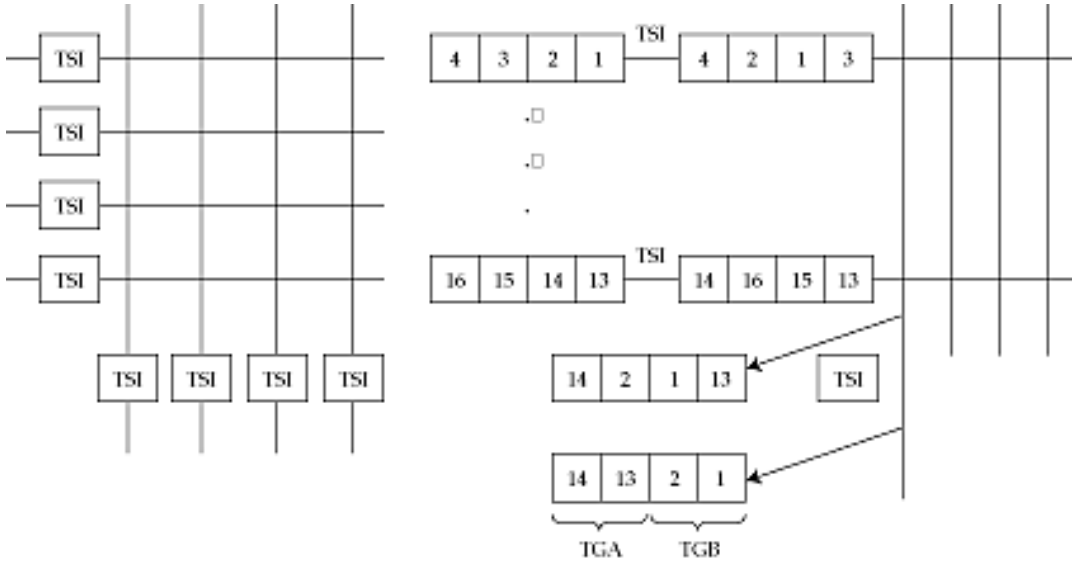
Time-space switching

- Precede each input trunk in a crossbar with a TSI
- Delay samples so that they arrive at the right time for the space division switch's schedule



Time-space-time (TST) switching

- Allowed to flip samples both on input and output trunk
- Gives more flexibility => lowers call blocking probability



Outline

- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Packet switching

- In a circuit switch, path of a sample is determined at time of connection establishment
- No need for a sample header--position in frame is enough
- In a packet switch, packets carry a destination field
- Need to look up destination port on-the-fly
- Datagram
  - lookup based on entire destination address
- Cell
  - lookup based on VCI
- Other than that, very similar

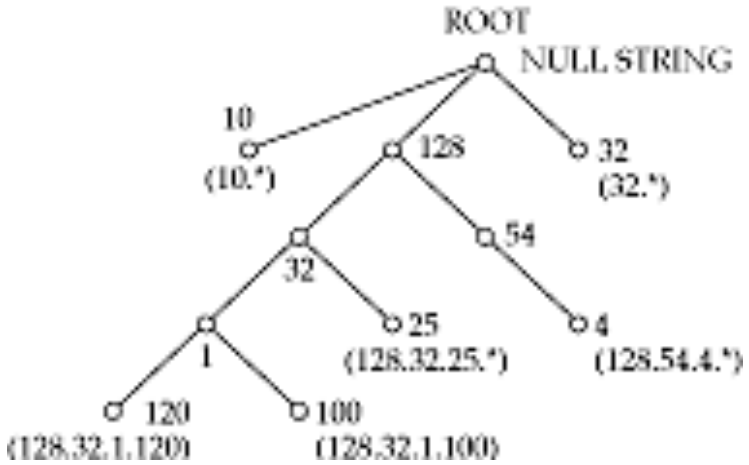
Repeaters, bridges, routers, and gateways

- Repeaters: at physical level
- Bridges: at datalink level (based on MAC addresses) (L2)
  - discover attached stations by listening
- Routers: at network level (L3)
  - participate in routing protocols
- Application level gateways: at application level (L7)
  - treat entire network as a single hop
  - e.g mail gateways and transcoders
- Gain functionality at the expense of forwarding speed
  - for best performance, push functionality as low as possible

Port mappers

- Look up output port based on destination address
- Easy for VCI: just use a table
- Harder for datagrams:
  - need to find *longest prefix match*
    - e.g. packet with address 128.32.1.20
    - entries: (128.32.*, 3), (128.32.1.*, 4), (128.32.1.20, 2)
- A standard solution: trie

Tries



- Two ways to improve performance
  - cache recently used addresses in a CAM
  - move common entries up to a higher level (match longer strings)

Blocking in packet switches

- Can have both internal and output blocking
- Internal
  - no path to output
- Output
  - trunk unavailable
- Unlike a circuit switch, cannot predict if packets will block (why?)
- If packet is blocked, must either buffer or drop it

Dealing with blocking

- Overprovisioning
  - internal links much faster than inputs
- Buffers
  - at input or output
- Backpressure
  - if switch fabric doesn't have buffers, prevent packet from entering until path is available
- Parallel switch fabrics
  - increases effective switching capacity

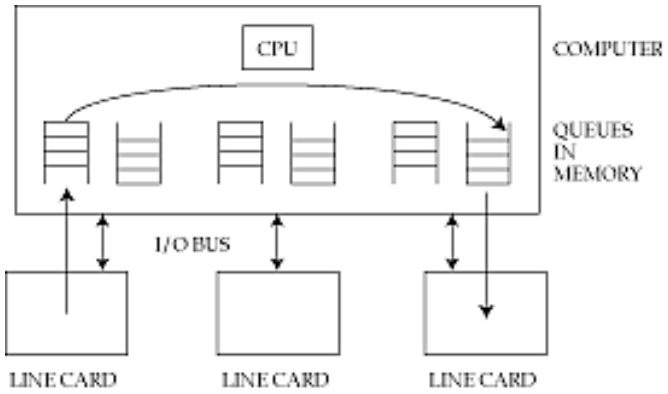
Outline

- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Three generations of packet switches

- Different trade-offs between cost and performance
- Represent evolution in switching capacity, rather than in technology
  - With same technology, a later generation switch achieves greater capacity, but at greater cost
- All three generations are represented in current products

First generation switch



- Most Ethernet switches and cheap packet routers
- Bottleneck can be CPU, host-adaptor or I/O bus, depending

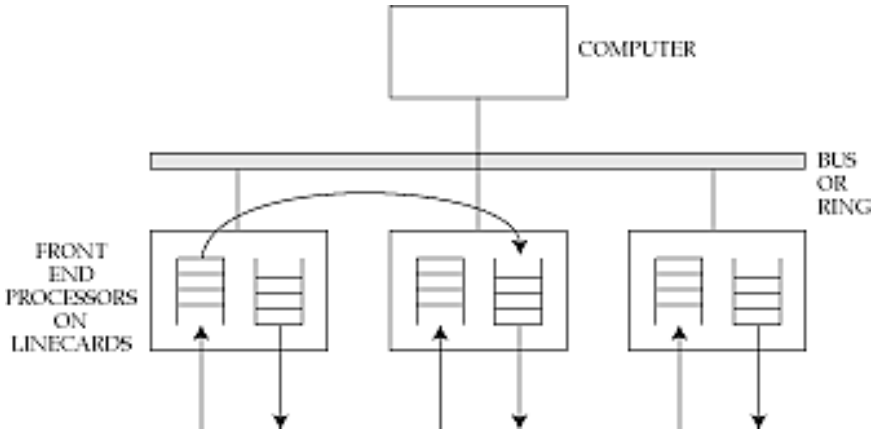
Example

- First generation router built with 133 MHz Pentium
  - Mean packet size 500 bytes
  - Interrupt takes 10 microseconds, word access take 50 ns
  - Per-packet processing time takes 200 instructions = 1.504 μs
- Copy loop

```
register <- memory[read_ptr]
memory [write_ptr] <- register
read_ptr <- read_ptr + 4
write_ptr <- write_ptr + 4
counter <- counter -1
if (counter not 0) branch to top of loop
```

- 4 instructions + 2 memory accesses = 130.08 ns
- Copying packet takes 500/4 *130.08 = 16.26 μs; interrupt 10 μs
- Total time = 27.764 μs => speed is 144.1 Mbps
- Amortized interrupt cost balanced by routing protocol cost

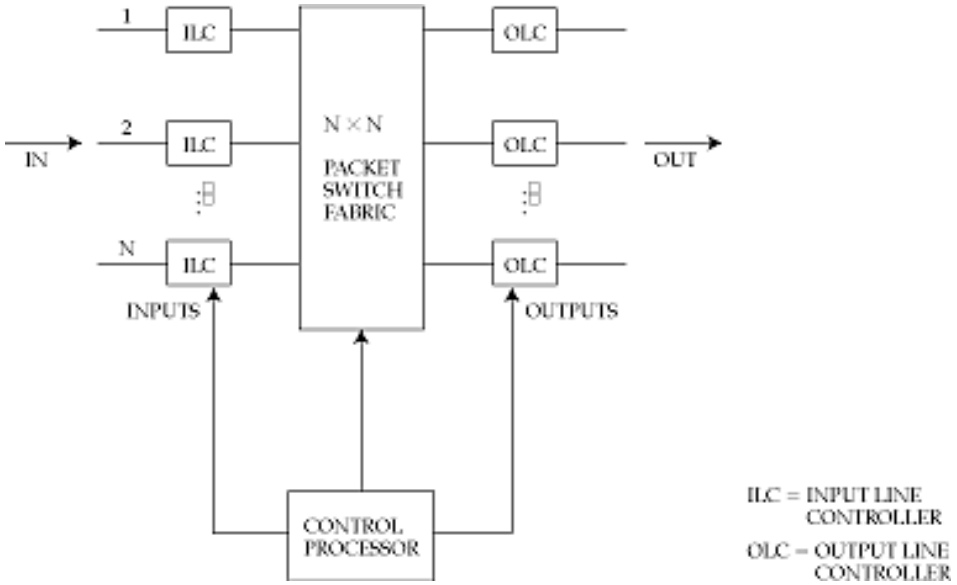
Second generation switch



- Port mapping intelligence in line cards
- ATM switch guarantees hit in lookup cache
- Ipsilon *IP switching*
  - assume underlying ATM network
  - by default, assemble packets
  - if detect a flow, ask upstream to send on a particular VCI, and install entry in port mapper => implicit signaling

Third generation switches

- Bottleneck in second generation switch is the bus (or ring)
- Third generation switch provides parallel paths (fabric)



Third generation (contd.)

- Features
  - self-routing fabric
  - output buffer is a point of contention
    - unless we *arbitrate* access to fabric
  - potential for unlimited scaling, as long as we can resolve contention for output buffer

Outline

- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Switch fabrics

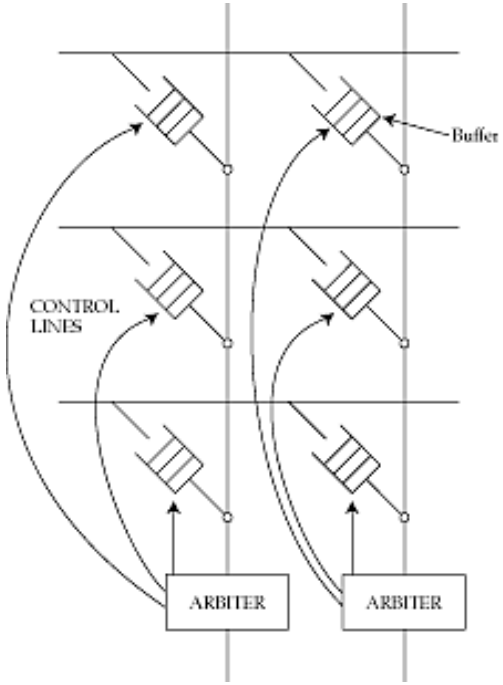
- Transfer data from input to output, ignoring scheduling and buffering
- Usually consist of links and *switching elements*

Crossbar

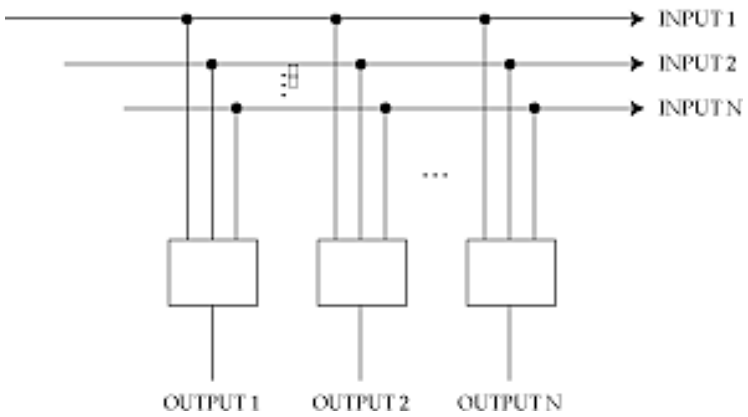
- Simplest switch fabric
  - think of it as 2N buses in parallel
- Used here for *packet* routing: crosspoint is left open long enough to transfer a packet from an input to an output
- For fixed-size packets and known arrival pattern, can compute schedule in advance
- Otherwise, need to compute a schedule on-the-fly (what does the schedule depend on?)

Buffered crossbar

- What happens if packets at two inputs both want to go to same output?
- Can defer one at an input buffer
- Or, buffer crosspoints



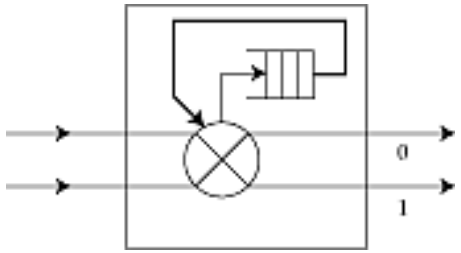
Broadcast



- Packets are tagged with output port #
- Each output matches tags
- Need to match N addresses in parallel at each output
- Useful only for small switches, or as a stage in a large switch

Switch fabric element

- Can build complicated fabrics from a simple element



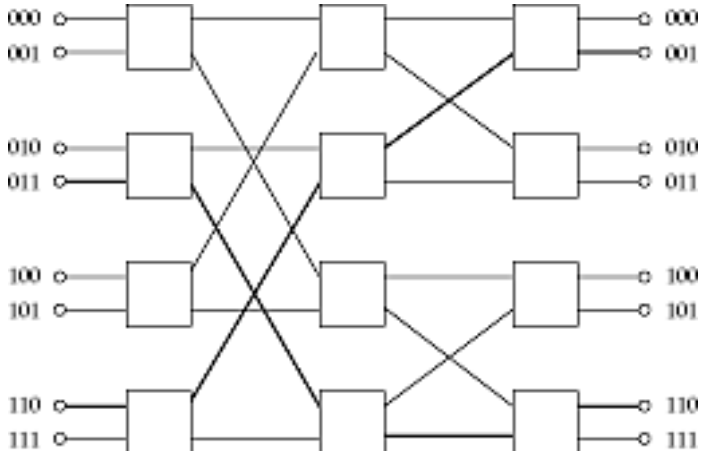
- Routing rule: if 0, send packet to upper output, else to lower output
- If both packets to same output, buffer or drop

Features of fabrics built with switching elements

- NxN switch with bxb elements has $\lceil \log_b N \rceil$ elements with $\lfloor N \backslash p \rfloor$ elements per stage
- Fabric is *self routing*
- Recursive
- Can be synchronous or asynchronous
- Regular and suitable for VLSI implementation

Banyan

- Simplest self-routing recursive fabric



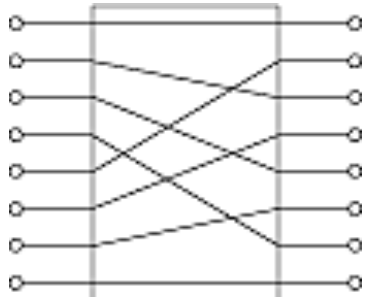
- (why does it work?)
- What if two packets both want to go to the same output?
  - output blocking

Blocking

- Can avoid with a buffered banyan switch
  - but this is too expensive
  - hard to achieve zero loss even with buffers
- Instead, can check if path is available before sending packet
  - three-phase scheme
  - send requests
  - inform winners
  - send packets
- Or, use several banyan fabrics in parallel
  - intentionally misroute and tag one of a colliding pair
  - divert tagged packets to a second banyan, and so on to k stages
  - expensive
  - can reorder packets
  - output buffers have to run k times faster than input

Sorting

- Can avoid blocking by choosing order in which packets appear at input ports
- If we can
  - present packets at inputs sorted by output
  - remove duplicates
  - remove gaps
  - precede banyan with a perfect shuffle stage
  - then no internal blocking

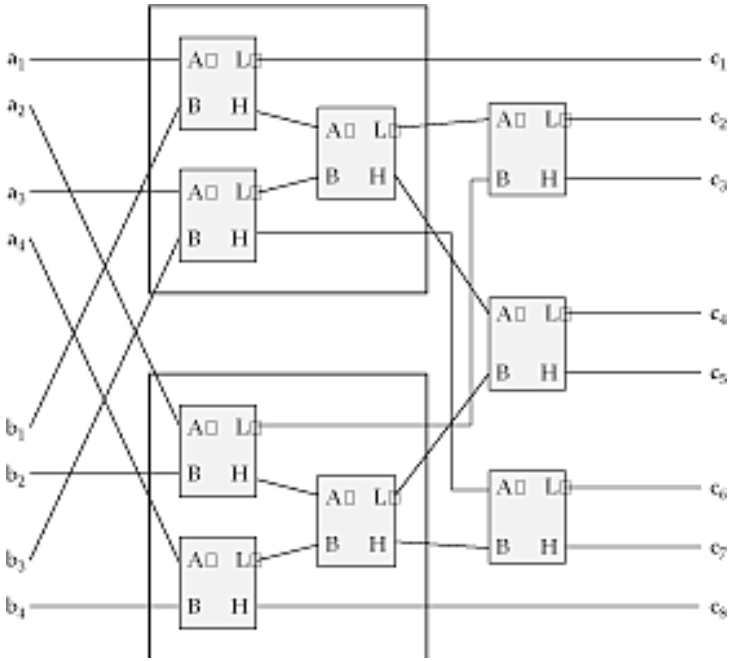


- For example, [X, 010, 010, X, 011, X, X, X] -(sort)->
  [010, 011, 011, X, X, X, X, X] -(remove dups)->
  [010, 011, X, X, X, X, X, X] -(shuffle)->
  [010, X, 011, X, X, X, X, X]
- Need sort, shuffle, and trap networks

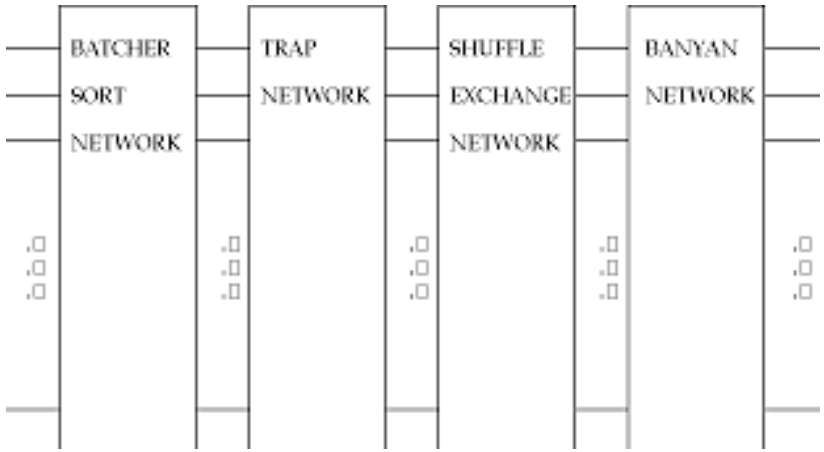
Sorting

- Build sorters from merge networks
- Assume we can merge two sorted lists
- Sort pairwise, merge, recurse

Merging



Putting it together- Batcher Banyan



- What about trapped duplicates?
  - recirculate to beginning
  - or run output of trap to multiple banyans (*dilation*)

Effect of packet size on switching fabrics

- A major motivation for small fixed packet size in ATM is ease of building large parallel fabrics
- In general, smaller size => more per-packet overhead, but more preemption points/sec
  - At high speeds, overhead dominates!
- Fixed size packets helps build synchronous switch
  - But we could fragment at entry and reassemble at exit
  - Or build an asynchronous fabric
  - Thus, variable size doesn't hurt too much
- Maybe Internet routers can be almost as cost-effective as ATM switches

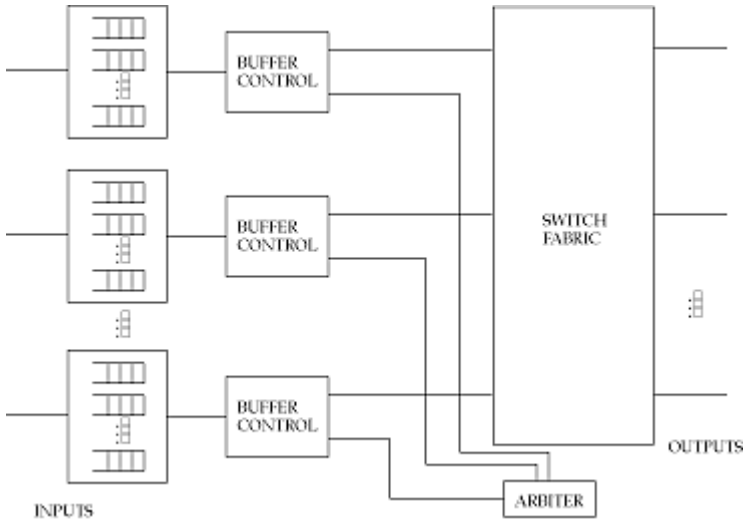
Outline

- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Buffering

- All packet switches need buffers to match input rate to service rate
  - or cause heavy packet loses
- Where should we place buffers?
  - input
  - in the fabric
  - output
  - shared

Input buffering (input queueing)

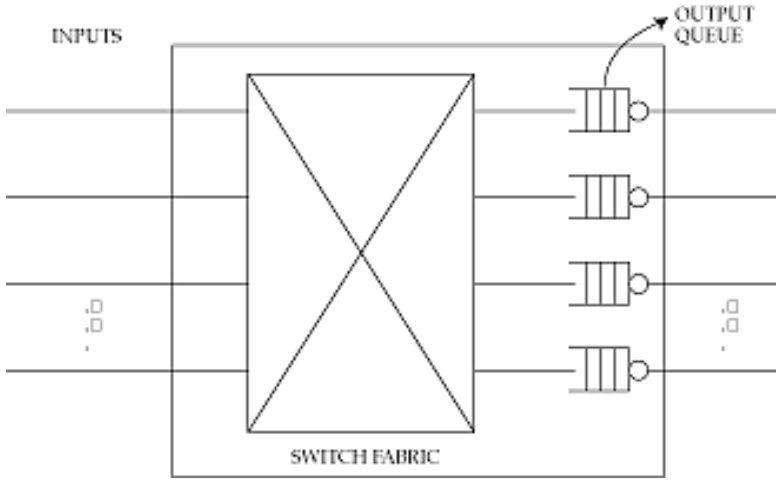


- No speedup in buffers or trunks (unlike output queued switch)
- Needs arbiter
- Problem: *head of line blocking*
  - with randomly distributed packets, utilization at most 58.6%
  - worse with *hot spots*

Dealing with HOL blocking

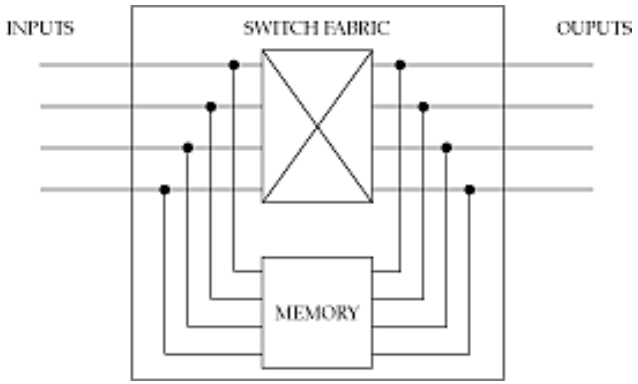
- Per-output queues at inputs
- Arbiter must choose one of the input ports for each output port
- How to select?
- Parallel Iterated Matching
  - inputs tell arbiter which outputs they are interested in
  - output selects one of the inputs
  - some inputs may get more than one *grant*, others may get none
  - if >1 grant, input picks one at random, and tells output
  - losing inputs and outputs try again
- Used in DEC Autonet 2 switch

Output queueing



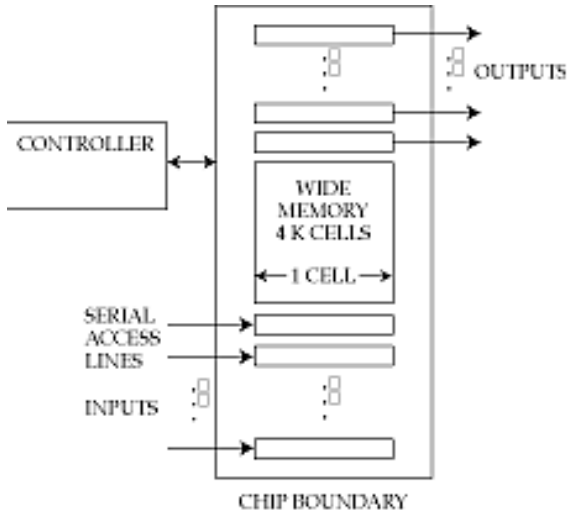
- Don't suffer from head-of-line blocking
- But output buffers need to run much faster than trunk speed (why?)
- Can reduce some of the cost by using the *knockout* principle
  - unlikely that all N inputs will have packets for the same output
  - drop extra packets, fairly distributing losses among inputs

Shared memory



- Route only the header to output port
- Bottleneck is time taken to read and write multiported memory
- Doesn’t scale to large switches
- But can form an element in a multistage switch

Datapath: clever shared memory design



- Reduces read/write cost by doing wide reads and writes
- 1.2 Gbps switch for $50 parts cost

Buffered fabric

- Buffers in each switch element
- Pros
  - Speed up is only as much as fan-in
  - Hardware backpressure reduces buffer requirements
- Cons
  - costly (unless using single-chip switches)
  - scheduling is hard

Hybrid solutions

- Buffers at more than one point
- Becomes hard to analyze and manage
- But common in practice

Outline

- Circuit switching
- Packet switching
  - Switch generations
  - Switch fabrics
  - Buffer placement
  - Multicast switches

Multicasting

- Useful to do this in hardware
- Assume portmapper knows list of outputs
- Incoming packet must be copied to these output ports
- Two subproblems
  - generating and distributing copes
  - VCI translation for the copies

Generating and distributing copies

- Either implicit or explicit
- Implicit
  - suitable for bus-based, ring-based, crossbar, or broadcast switches
  - multiple outputs enabled after placing packet on shared bus
  - used in Paris and Datapath switches
- Explicit
  - need to copy a packet at switch elements
  - use a *copy* network
  - place # of copies in tag
  - element copies to both outputs and decrements count on one of them
  - collect copies at outputs
- Both schemes increase blocking probability

Header translation

- Normally, in-VCI to out-VCI translation can be done either at input or output
- With multicasting, translation easier at output port (why?)
- Use separate port mapping and translation tables
- Input maps a VCI to a set of output ports
- Output port swaps VCI
- Need to do two lookups per packet

Scheduling and queue management

DigiComm II

Traditional queuing behaviour in routers

- Data transfer:
  - datagrams: individual packets
  - no recognition of **flows**
  - connectionless: no signalling
- Forwarding:
  - based on per-datagram, forwarding table look-ups
  - no examination of "type" of traffic – no **priority** traffic
- Traffic patterns

DigiComm II

Let us first examine the service that IP offers. IP offers a **connectionless** datagram service, giving no guarantees with respect to delivery of data: no assumptions can be made about the delay, jitter or loss that any individual IP datagrams may experience. As IP is a connectionless, datagram service, it does not have the notion of **flows** of datagrams, where many datagrams form a sequence that has some meaning to an applications. For example, an audio application may take 40ms "time-slices" of audio and send them in individual datagrams. The correct sequence and timeliness of datagrams has meaning to the application, but the IP network treats them as individual datagrams with no relationship between them. There is no **signalling** at the IP-level: there is no way to inform the network that it is about to receive traffic with particular handling requirements and no way for IP to tell signal users to back-off when there is congestion.

At IP routers, the forwarding of individual datagrams is based on forwarding tables using simple metrics and (network) destination addresses. There is no examination of the type of traffic that each datagram may contain – all data is treated with equal **priority**. There is no recognition of datagrams that may be carrying data that is sensitive to delay or loss, such as audio and video.

As IP is a totally connectionless datagram traffic, there is no protection of the packets of one flow, from the packets of another. So, the traffic patterns of one particular user's traffic affects traffic of other users that share some part or of, or all of, the network path (and perhaps even traffic that does not share the same network path!).

Questions

- How do we modify router scheduling behaviour to support QoS?
- What are the alternatives to FCFS?
- How do we deal with congestion?

DigiComm II

So we can ask ourselves several questions.

Firstly, can we provide a better service that that which IP currently provides– the so-called best-effort?

The answer to this is actually, “yes”, but we need to find out what it is we really want to provide! We have to establish which parameters of a real-time packet flow are important and how we might control them. Once we have established our requirements, we must look at new mechanisms to provide support for these needs in the network itself. here are many functional elements required in order to provide QoS in the network, some of which we will look at later. Here, we are essentially trying to establish alternatives to FCFS for providing better control of packet handling in the network.

We also need to consider how the applications gain access to such mechanisms, so we must consider any **application-level interface** issues, e.g. is there any interaction between the application and the network and if so, how will this be achieved.

In all our considerations, one of the key points is that of **scalability** – how would our proposals affect (and be affected by) use of IP on a global scale, across the Internet as a whole. Also, we must try to adhere, as much as possible, to the current service interface, i.e. a connectionless datagram delivery.

Scheduling mechanisms

Scheduling [1]

- Service request at server:
  - e.g. packet at router inputs
- Service order:
  - which service request (packet) to service first?
- Scheduler:
  - decides service order (based on policy/algorithm)
  - manages service (output) queues
- Router (network packet handling server):
  - **service:** packet forwarding
  - **scheduled resource:** output queues
  - **service requests:** packets arriving on input lines

DigiComm II

In general, the job a a scheduler is to control access to resources at some kind of server. The server offer a service and receives service requests. the way in which the scheduler determines which service request is dealt with next is subject to some well-defined policy or algorithm.

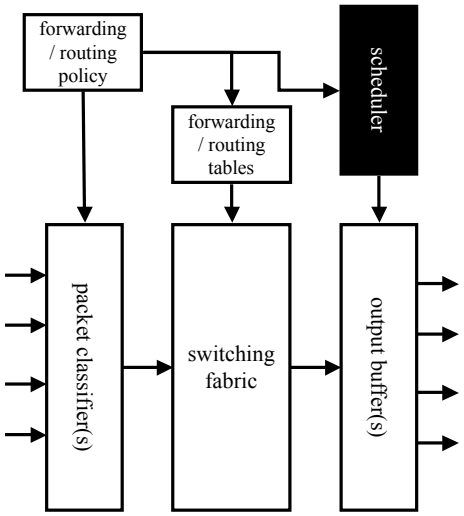
The main job of a scheduler is to make decisions regarding the order in which service request should be allowed access to resources. It has a secondary job which is to manage the service queues. This secondary roles reflects that fact that, in the face of excessive service requests, the finite resources of the server must be managed in a way that is consistent with the policy that is established at the server.

For a router (a network packet handling server), the service being offered is a packet forwarding service; the resource which is under the control of the scheduler is (are) the output queue(s) of the router; service requests are packets that need forwarding arriving on input line(s).

Scheduling [2]

Simple router schematic

- Input lines:
  - no input buffering
- Packet classifier:
  - policy-based classification
- Correct output queue:
  - forwarding/routing tables
  - switching fabric
  - output buffer (queue)
- Scheduler:
  - which output queue serviced next



DigiComm II

In a model of a simple router, we have the following functional elements.

forwarding/routing policy: this is part of the decision making process that is set-up by a network administrator. The policy includes information about how to classify packets as well as any constraints on routing an forwarding.

forwarding/routing tables: these are constructed by the normal operation of the routing protocols and algorithms that run in the router.

input lines: as packets arrive interrupts are generated to indicate that an input should be processed. We assume that for a QoS sensitive network, router do not perform input buffering. Although input buffering is possible, packets may have their QoS requirements violated as they wait in input buffers, so we assume that packets are processed at line speed.

packet classifier: this makes some policy-based decision in order to classify the packets. The packet classification will allow the scheduler to make decisions about the order in which packets are serviced.

switching fabric: this moves a packet in an input queue to the appropriate output queue.

output queue: hold packets awaiting transmission.

scheduler: makes decisions as to which output queue should be serviced next.

More information router design and scheduling can be found in [KS98] and [KLS98].

FCFS scheduling

- Null packet classifier
- Packets queued to outputs in order they arrive
- Do packet differentiation
- No notion of flows of packets
- Anytime a packet arrives, it is serviced as soon as possible:
  - FCFS is a **work-conserving** scheduler

DigiComm II

In first-come first-served (FCFS) scheduling, the packets are scheduled in the order they arrive at the router. No other packet differentiation is performed. The router has no notion of flows of packets. In fact, FCFS scheduling is a very function and the main concern of most FCFS scheduled routers is queue management – coping with excessive bursts of packets with finite buffer space. We look at key management techniques and congestion control later.

There is one very important aspect of FCFS which we will examine before we go on to discuss other scheduling disciplines, and that is that it is a **work-conserving** scheduling disciple. That is, the router is never idle when there are packets waiting to be serviced.

Conservation law [1]

- FCFS is work-conserving:
  - not idle if packets waiting
- Reduce delay of one flow, increase the delay of one or more others
- We can not give *all* flows a lower delay than they would get under FCFS

$$\sum_{n=1}^{N} \rho_n q_n = C$$

$\rho_n = \lambda_n \mu_n$

ρ_n : mean link utlisation

q_n : mean delay due to scheduler

C : constant [s]

λ_n : mean packet rate [p/s]

μ_n : mean per – packet service rate [s/p]

DigiComm II

An important theorem due to Kleinrock (p.117 in [Kle75]) helps us in analysing scheduling disciplines. This theorem is **The Conservation Law**. Consider N flows arriving at a scheduler, so that the traffic rate of connection n ($1 \le n \le N$) is λ_n. If we assume that flow n has a mean service rate of μ_n. So, the mean utilisation of a link by flow n, ρ_n, is $\lambda_n \mu_n$. Let the mean waiting time due to the scheduler for the packets of flow n be q_n. According to the Conservation Law, the scheduler is then work conserving if:

$\sum_n \rho_n q_n = C$ where C is a constant value

This expression is important as it makes no reference to any particular scheduling disciplines. This expression also says that for any work conserving scheduling discipline, if one of the flows is given a lower delay by the scheduler, then it must be by increasing the delay for one or more of the other flows. This is a fundamental result.

Conservation law [2]

Example

- μ_n : 0.1ms/p (fixed)
- Flow f1:
  - λ_1 : 10p/s
  - q_1 : 0.1ms
  - $\rho_1 q_1 = 10^{-7}$s
- Flow f2:
  - λ_2 : 10p/s
  - q_2 : 0.1ms
  - $\rho_2 q_2 = 10^{-7}$s
- $C = 2\times10^{-7}$s

- Change f1:
  - λ_1 : 15p/s
  - q_2 : 0.1s
  - $\rho_1 q_1 = 1.5\times10^{-7}$s
- For f2 this means:
  - decrease λ_2?
  - decrease q_2?
- Note the trade-off for f2:
  - **delay vs. throughput**
- Change service rate (μ_n):
  - change service **priority**

DigiComm II

Let us demonstrate this using an example. Assume we have two equal flows, initially, f1 and f2. We assume that packet service rate is a function of the hardware/software of the router (though it is possible it is also a function of the average packet size in the flow) and is therefore fixed and equal for all flows. For the initial situation shown ab0ve, we can use the conservation law to evaluate *C*. If we now change the attributes f flow 1, so that we would like it to have a higher data rate, we find that the Conservation Law tells us that we must either decrease the data rate of f2 to maintain its data rate, or decrease its data rate to maintain the delay.

It is possible to change the service rate of each flow by changing the **priority** of the flow. Here, we have assumed that both flows have the same, i.e. FCFS scheduling is being used.

Non-work-conserving schedulers

- Non-work conserving disciplines:
  - can be idle even if packets waiting
  - allows "smoothing" of packet flows
- Do not serve packet as soon as it arrives:
  - what until packet is **eligible** for transmission
- Eligibility:
  - fixed time per router, or
  - fixed time across network

✓ Less jitter

✓ Makes downstream traffic more predictable:
  - output flow is controlled
  - less bursty traffic

✓ Less buffer space:
  - router: output queues
  - end-system: de-jitter buffers

× Higher end-to-end delay

× Complex in practise
  - may require time synchronisation at routers

DigiComm II

We have defined a work-conserving scheduler as one that is never idle if there are packets waiting. So what is a non-work conserving scheduler and why might they be useful? In a non-work-conserving scheduler, the scheduler can be idle even though there are packets waiting for transmission. The reason it remains idle is that waits for packets to become eligible for transmission. It is a bit like a clever set of traffic lights at a junction – not only does it know where you are going, it also knows how fast you should be going and only lets you move on when the time is right, even if there is no other traffic going through at other junctions. (Continuing the analogy, a work-conserving discipline is more like a junction with stop signs – as soon as there is a chance for you to go through, then you do.)

The eligibility of a packet can be determined in different ways. One way is to ensure that it always spends no more than a fixed time at a router. In this way the end-to-end jitter is controlled. This is equivalent to control the data rate and the jitter, and so such mechanisms are called **delay-jitter** regulators. Another way to determine eligibility is establish the notion of a fixed end-to-end delay for a packet. Then, routers decide a packet is eligible based on the time budget that remains for each packet. This is called a **delay-jitter** regulator as both the end-to-end delay and the end-to-end jitter are controlled.

The advantages of non-work conserving disciplines is that the reduce jitter, making downstream traffic more predictable. This means that there are fewer and smaller traffic burst, which reduces the buffering requirements at router and end-systems. However, we have already discussed that end-systems can de-jitter streams as required.

However, the cost of using such mechanisms is higher end-to-end delay overall. Also, such systems can be complex to build in practise. For example, delay-jitter regulators require that the network operator know the propagation on each link in the network, and that all router maintain a synchronised clock with which they can establish the eligibility of packets. Consequently, non-work-conserving disciplines are not used are are still considered a research issue.

Scheduling: requirements

- Ease of implementation:
  - simple → fast
  - high-speed networks
  - low complexity/state
  - implementation in hardware
- Fairness and protection:
  - local fairness: **max-min**
  - local fairness → global fairness
  - protect any flow from the (mis)behaviour of any other
- Performance bounds:
  - per-flow bounds
  - deterministic (guaranteed)
  - statistical/probabilistic
  - data rate, delay, jitter, loss
- Admission control:
  - (if required)
  - should be easy to implement
  - should be efficient in use

DigiComm II

There are four key requirements that can be listed for any scheduling mechanism.

• **ease of implementation:** the mechanisms should be easy to implement. Keeping the mechanism as simple as possible makes it faster, it should be easier to implement. Also, if we can keep the amount of algorithmic complexity low, as well as minimise the amount of state information needed by the algorithm, then the mechanisms may lend itself more easily to implementation in hardware, making it suitable for high-speed networking.

• **fairness and protection:** flows should receive fair allocation of resource, all other things being equal, they should not receive any less than any other flow also asking to use the same resources. In fact, it is possible to be much more precise about how resources should be allocated, using the **max-min fairness** criteria. The max-min fairness criteria allows per-hop local fairness for each flow, which results in global fairness for all of the flows. The protection requirement states that no flow should suffer due to the (mis)behaviour or characteristics of any other flow. We can see from the Conservation Law that FCFS does not provide protection as if one flow was to increase its data rate, all the other flows would suffer.

• **performance bounds:** for supporting QoS in networks, it should be possible to specify performance bounds to describe so that we can be sure that our flow is handled appropriately by the network. The exact nature of these performance bounds depends on the kind of QoS guarantee that we need for our flow. Where a scheduler is being used to provide a guaranteed service, it would have to offer the ability to work with deterministic performance bounds. If relative priorities for flows were being specified, then the scheduler should allow the specification of performance bounds in a statistical or probabilistic way.

• **admission control:** where deterministic performance bounds are specified, the network may need to perform admission control before it can allow a flow to start transmission. In this case the admission control mechanism should be easy to implement and efficient to operate.

The max-min fair share criteria

- Flows are allocated resource in order of increasing demand
- Flows get no more than they need
- Flows which have not been allocated as they demand get an equal share of the available resource
- Weighted max-min fair share possible
- If max-min fair → provides protection

$$m_n = \min(x_n, M_n) \quad 1 \le n \le N$$

$$M_n = \frac{C - \sum_{i=1}^{n-1} m_i}{N - n + 1}$$

C : capacity of resource (maximum resource)

m_n : actual resource allocation to flow n

x_n : resource demand by flow $n, x_1 \le x_2 \cdots \le x_N$

M_n : resource available to flow n

Example:
C = 10, four flow with demands of 2, 2.6, 4, 5
actual resource allocations are 2, 2.6, 2.7, 2.7

DigiComm II

In **max-min fair share**, resources are allocated according to some simple rules. The demands of the resource flows are ordered in increasing demand so that flows with the lowest demands are allocated resources first. This means that resources with small demands are likely to be allocated all that they ask for (as the simple example above illustrates.)

It is possible to assign weights to the resource demands so that some demands receive a greater share of the resources than others. In this case th rules ofr max-min fair share are modified as follows:

• flows are allocated resource in order of increasing **weighted** demand (the demands are normalised by the weight)

• no flow gets more than it needs

• flows which have not been allocated as they demand get a **weighted** share of the available resource

Scheduling: dimensions

- Priority levels:
  - how many levels?
  - higher priority queues services first
  - can cause starvation lower priority queues
- Work-conserving or not:
  - must decide if delay/jitter control required
  - is cost of implementation of delay/jitter control in network acceptable?
- Degree of aggregation:
  - flow granularity
  - per application flow?
  - per user?
  - per end-system?
  - cost vs. control
- Servicing within a queue:
  - "FCFS" within queue?
  - check for other parameters?
  - added processing overhead
  - queue management

DigiComm II

When designing a scheduling mechanism, we have several **dimensions** or **degrees of freedom** in which we can work.

The first of these is the use of **priority levels**. Here we assign different priorities to different output queues. Queues with higher priorities receive service first. Queues with lower priorities are only serviced when the higher priorities queues are empty. We can decide to have an arbitrary number of priority levels to reflect our QoS policy.

We can also choose for our scheduling mechanism to be **work-conserving** or **non-work-conserving**. We have already discussed the differences between these two, and the decision boils down to two questions:

• do you need to have good delay and jitter control?

• if so, is the additional cost of implementing non-work-conserving schedulers acceptable?

When we design our scheduling policy, we must decide on the granularity of our flows, i.e. we must decide on the level of aggregation of traffic that we want. We can have large levels of **aggregation** and have relatively poor control over individual application-level flows, or we can have per-application flow state and have very fine grained control. The choice is a trade-off between the amount of control we have for our flows (with respect to individual packet sources) versus the amount of state we store for the scheduler and the processing overhead in involved with that state.

Finally, within an individual queue, we need to decide on how packets should be **serviced within that queue**. We can just decide to service the packets in the order in which they arrived within the queue (i.e. treat each queue as a single FCFS queue waiting fro transmission) or we can examine the contents of individual packets in the queue and make fine-grained policy decisions as to which should be serviced first. Again, the trade-off is between the level of control desired and the cost of the processing required. Queue servicing may be required for queue management, which we look at later.

Simple priority queuing

- K queues:
  - $1 \le k \le K$
  - queue $k + 1$ has greater priority than queue k
  - higher priority queues serviced first
- ✓ Very simple to implement
- ✓ Low processing overhead
- Relative priority:
  - no deterministic performance bounds
- ✗ Fairness and protection:
  - not max-min fair: starvation of low priority queues

DigiComm II

In priority queuing, K queues are established, numbered 1 … K. Higher numbered queues are serviced before lower numbered queues – they have higher priority. A lower priority queue will only be serviced if there are no packets awaiting service in a higher priority queue. This means that busy higher priority queues could prevent lower priority queues from being serviced– situation known as starvation. This system would need to be used with some other mechanism to police the traffic into the queues, e.g. a token bucket filter for each queue, plus admission control at the edge of the network.

Note that the queues assign relative priority, so this mechanism by itself would not be suitable for providing deterministic guarantees for performance bounds.

Generalised processor sharing (GPS)

- Work-conserving
- Provides max-min fair share
- Can provide weighted max-min fair share
- Not implementable:
  - used as a reference for comparing other schedulers
  - serves an infinitesimally small amount of data from flow *i*
- Visits flows round-robin

$\phi(n) \quad 1 \le n \le N$

$S(i,\tau,t) \quad 1 \le i \le N$

$\frac{S(i,\tau,t)}{S(j,\tau,t)} \ge \frac{\phi(i)}{\phi(j)}$

$\phi(n)$: weight given to flow *n*

$S(i,\tau,t)$: service to flow *i* in interval *[ôô,t*

flow *i* has a non – empty queue

DigiComm II

Generalised processor sharing (GPS) is a work-conserving scheduler that provide max-min fairness and protection. It can be used to provide probabilistic/statistical (relative) as well as deterministic performance bounds. Unfortunately, it is not implementable as the analysis requires that for each flow *i*, only an infinitesimally small amount of data is served. However, the analysis is useful in that it provides us with a reference with which we can assess other schedulers by comparison. All flows are serviced round-robin. GPS is described in [GP93] and [GP94].

GPS – relative and absolute fairness

- Use fairness bound to evaluate GPS emulations (GPS-like schedulers)
- Relative fairness bound:
  - fairness of scheduler with respect to other flows it is servicing
- Absolute fairness bound:
  - fairness of scheduler compared to GPS for the same flow

$$RFB = \left| \frac{S(i,\tau,t)}{g(i)} - \frac{S(j,\tau,t)}{g(j)} \right|$$

$$AFB = \left| \frac{S(i,\tau,t)}{g(i)} - \frac{G(i,\tau,t)}{g(i)} \right|$$

$S(i,\tau,t)$: actual service for flow i in $[\tau,t]$

$G(i,\tau,t)$: GPS service for flow i in $[\tau,t]$

$g(i) = \min\{g(i,1),\cdots,g(i,K)\}$

$$g(i,k) = \frac{\phi(i,k)r(k)}{\sum_{j=1}^{N}\phi(j,k)}$$

$\phi(i,k)$: weight given to flow i at router k

$r(k)$: service rate of router k

$1 \le i \le N$ flow number

$1 \le k \le K$ router number

DigiComm II

In [Gol94] is described two fairness bounds that can be used to assess the performance of the service, *S*, provides by an emulation of GPS (i.e. a scheduler that attempts to achieve a GPS-like service). The first fairness bound is the **relative fairness bound (RFB)**, which attempts to provide evaluate how fairly resources are allocated between flows being serviced by the GPS emulation. The second fairness bound is the **absolute fairness bound (AFB)**, which tries to make a comparison of the service provided by the GPS emulation to that which would be provided by GPS. The closer that the RFB and AFB are to zero the better. It is normally hard to obtain an evaluation of the AFB fro most emulations, so the RFB is used.

In the expressions above, it is assumed that we examine the *K* routers along a path that serve *N* flows. *S*(.) is the service given to a flow by the GPS emulation and *G*(.) is the service that would be given by GPS. g(*i*, *k*) is the relative service rate given to flow *i* at router *k* along the path. g(*i*) is, effectively, the bottleneck rate along the path.

Weighted round-robin (WRR)

- Simplest attempt at GPS
- Queues visited round-robin in proportion to weights assigned
- Different means packet sizes:
  - weight divided by mean packet size for each queue
- Mean packets size unpredictable:
  - may cause unfairness
- Service is fair over long timescales:
  - must have more than one visit to each flow/queue
  - short-lived flows?
  - small weights?
  - large number of flows?

DigiComm II

The simplest emulation of GPS is **weighted round-robin (WRR)**. Here, queues are serviced round-robin, in proportion to a weight that is assigned to each flow/queue, i.e. flows/queues with a greater weight have more service. In each round each queue is visited once. Normally, at least one packet is transmitted from each queue that is non-empty. The weight assigned to each flow/queue is normalised by dividing the by the average packet size for each flow/queue. If this is not know before hand, them WRR may not be able to offer the correct service rate for the flow. For many applications, it is not easy to evaluate the average packet size before hand. The service provided by WRR can be shown to be fair over long-lived flows but for short lived flows, flows with small weights or where there are a large number of flows, WRR may exhibit unfairness.

Deficit round-robin (DRR)

- DRR does not need to know mean packet size
- Each queue has deficit counter (dc): initially zero
- Scheduler attempts to serve one quantum of data from a non-empty queue:
  - packet at head served if size ≤ quantum + dc
    dc ← quantum + dc – size
  - else dc += quantum
- Queues not served during round build up "credits":
  - only non-empty queues
- Quantum normally set to max expected packet size:
  - ensures one packet per round, per non-empty queue
- RFB: 3T/r (T = max pkt service time, r = link rate)
- Works best for:
  - small packet size
  - small number of flows

DigiComm II

In **deficit round-robin (DRR)**, an attempt is made to by-pass the requirement to know the average packet size for each flow. Each non-empty queue initially has a deficit counter (dc) which starts of at the value zero. As the scheduler visits each non-empty queue, it compares the packet at the head each of these queues and tries to serve one quantum of data. If dc is zero for a queue, then the packet is served if the size of the packet at the head of the queue is less than or equal to the quantum size. If a packet can not be served, then the value of the quantum is added to the dc for that queue. If the scheduler visits a queue with a dc > 0, the a packet from the queue is served if quantum + dc is greater than or equal to the size of the packet at the head of the queue. If the scheduler sees a queue with a non-zero dc, yet it is empty, then the dc is reset to zero so that it can not keep acquiring deficit. The quantum size used is normally that largest expected packet size in the network. Example, three flows A, B, C wit packets of size 600, 1000, 1400 bytes. The quantum size is set to 1000 bytes. In the first round:

• queue A: packet is served, dc = 1000 – 600 = 400

• queue B: packet is served, dc = 1000 – 1000 = 0

• queue C: packet is not served, dc = 1000

In the second round:

• queue A: no packet, dc set to zero

• queue B: no packet, dc set to zero

• queue C: packet is served, dc = 1000 + 1000 – 1400 = 600

In third round:

• queue A: not served

• queue B: not served

• queue C: no packet, dc set to zero

If weights are assigned to the queues, then the quantum size applied for each queue is multiplied by the assigned weight.

DRR works best for small packets sizes and a small number of flows, suffering similar unfairness behaviour to that for WRR.

Weighted Fair Queuing (WFQ) [1]

- Based on GPS:
  - GPS emulation to produce **finish-numbers** for packets in queue
  - Simplification: GPS emulation serves packets bit-by-bit round-robin
- **Finish-number**:
  - the time packet would have completed service under (bit-by-bit) GPS
  - packets tagged with finish-number
  - smallest finish-number across queues served first
- **Round-number**:
  - execution of round by bit-by-bit round-robin server
  - finish-number calculated from round number
- If queue is empty:
  - finish-number is: *number of bits in packet* + *round-number*
- If queue non-empty:
  - finish-number is: *highest current finish number for queue* + *number of bits in packet*

DigiComm II

WFQ is an emulation of GPS that actually uses a GPS computations to control its behaviour. In fact it runs a GPS computations and uses these to tag packets in flows to indicate the time the would have finished being served had they been subject to GPS scheduling. WFQ itself approximates GPS by using a model of bit-by-bit round robin service, i.e. for a packet of size N bits at the head of a queue, it will have complete service after the scheduler has performed N rounds of visiting all the other queues. This is bit-by-bit fair. However, we can not actually transmit packets one bit at a time from multiple queues, so instead the packets are tagged with a **finish-number**, which indicates when they would have completed service if served bit-by-bit. So, we can see that packets with the smallest finish number are the ones that should be transmitted first. Let us take the simple case that the queue is empty and a packet of size 50 bits arrives in the queue. As it is served bit-by-bit round robin, it will be serviced in 50 rounds. So its finish number will simply be the current round-number plus its own size in bits. So if the round number is 20, the finish-number for our packet is 70. If the queue is non-empty, then packets already in the queue will be served first. So the finish number of this new packet, which is at the back of the queue, will be the finish number for the packet in front of it plus its own size in bits. When the round-number is equal to (or greater than) the finish-number of the packet, it should be served immediately.

The closest analogy (at least that I can think of at this time!) is probably that of using a number system when you go to the delicatessen counter at the supermarket. All the people (packets) are of unit size, so this simplifies things. They are served one at a time (unit-by-unit) and people can appear at the delicatessen counter from any of a number of aisles in the supermarket. A person takes a piece of paper with a number on it from the from the counter. This is your finish-number and is one more than the person in front of you. It does not matter where you are physically, but when your number comes up on the big counter display (the round-number), it is your turn to be served.

Weighted Fair Queuing (WFQ) [2]

$$F(i,k,t) = \max\{F(i,k-1,t), R(t)\} + P(i,k,t)$$

$F(i,k,t)$: finish - number for packet k on flow i arriving at time t

$P(i,k,t)$: size of packet k on flow i arriving at time t

$R(t)$: round - number at time t

$$F_\phi(i,k,t) = \max\{F_\phi(i,k-1,t), R(t)\} + \frac{P(i,k,t)}{\phi(i)}$$

$\phi(i)$: weight given to flow i

- Rate of change of $R(t)$ depends on number of active flows (and their weights)
- As $R(t)$ changes, so packets will be served at different rates

- Flow completes (empty queue):
  - one less flow in round, so
  - R increases more quickly
  - so, more flows complete
  - R increases more quickly
  - etc. …
  - **iterated deletion** problem
- WFQ needs to evaluate R each time packet arrives or leaves:
  - processing overhead

DigiComm II

More formally, the evaluation of the finish-number, F, is as shown above.

The 'W' in WFQ comes from applying weights to evaluation of the finish-number, as shown in the expression for F_ϕ.

We can see here that the finish time is depended on R. Also, F and R are both functions of absolute time t. As the number of flows change, so does the rate of change of R. As more flows become active, so the rate of change of R decreases. As the number flows decreases, so the rate of change of R increases. This latter property leads to a problem know as **iterated deletion**. A flow is deleted if it has completed, i.e. the queue becomes empty. When this happens, it means that R will increase at a faster rate. This means that the R will approach the finish-number of other packets, already queued, faster than originally evaluated. This may cause some of the other flows to complete also, leading to a further increase in the rate of change of R. In order to keep track of R, a WFQ system must evaluate the value of R every time a packet arrives or leaves.

Weighted Fair Queuing (WFQ) [3]

- Buffer drop policy:
  - packet arrives at full queue
  - drop packets already in queued, in order of decreasing finish-number
- Can be used for:
  - best-effort queuing
  - providing guaranteed data rate and deterministic end-to-end delay
- WFQ used in "real world"
- Alternatives also available:
  - self-clocked fair-queuing (SCFQ)
  - worst-case fair weighted fair queuing (WF^2Q)

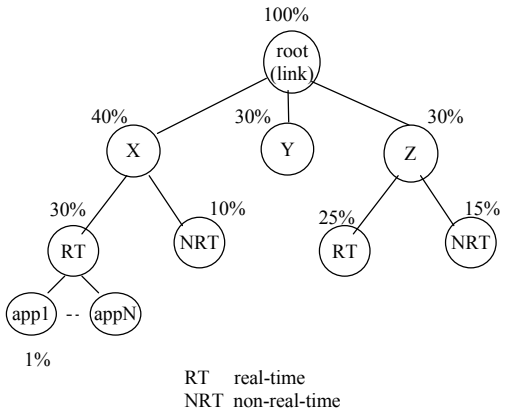
WFQ also proposes a a buffer drop policy that is max-min fair. If a packet arrives to a full queue, enough packets are dropped to allow the new packet to be queued. Packets are dropped in order of decreasing finish-number, i.e. most recent arrivals are dropped first.

WFQ can be used for scheduling multiple best effort flows, providing protection for each flow. With a few modifications – the weighted part of WFQ which we have not examined – it can also provide services for flows that require guaranteed data rate and an upper bound on the end-to-end delay.

Even though it is computationally expensive, and need to hold per-flow state, WFQ is used by many router manufacturers. Alternatives to WFQ, such as self-clocked fair-queuing (SCFQ) and worst-case fair weighted fair queuing (WF^2Q). SCFQ has similar properties to WFQ, except that it removes the expensive round-number computation but also has a higher end-to-end delay bound than WFQ. WF^2Q has an AFB that more closely approaches the AFB that is lower than WFQ (i.e. it has a a better AFB than WFQ), but requires an additional implementation complexity and has higher processing overhead in operation.

Class-Based Queuing

- Hierarchical link sharing:
  - link capacity is shared
  - class-based allocation
  - policy-based class selection
- Class hierarchy:
  - assign capacity/priority to each node
  - node can "borrow" any spare capacity from parent
  - fine-grained flows possible
- Note: this is a queuing mechanism: requires use of a scheduler



DigiComm II

CBQ was designed to allow sharing of link capacity within a class-based hierarchy [FJ95] [WGCJF95]. We see an example showing the link capacity as a root node in a tree at 100%. Organisations X, Y and Z that share the link are assigned 40%, 30% and 30% of the link capacity, respectively. Within their own allocations of capacity, the organisations can choose to partition the available capacity further by creating sub-classes within the tree. Organisation X decides to allocate 30% to real-time traffic and 10% to all non-real-time traffic. Within the real-time allocation, X decides to allocate capacity to individual applications. Organisation Z also divides its allocation into real-time and non-real-time, but with a different share of the available link capacity. Organisation Y decides not to further refine its allocation of link capacity. The percentages indicate the minimum share of the link capacity a node in the tree will receive. Child nodes effectively take capacity from their parent node allocation. If some sibling nodes are not using their full allocations, other siblings that might be overshooting their own allocation are allowed to "borrow" capacity by interacting with the parent node. With an appropriate scheduling mechanism, this allows support for QoS sensitive flows. Classifications could be made per application, per flow, per IP source address, etc., as dictated by the policy chosen by the individual organisations in conjunction with their Internet connectivity provider. Not shown above is that different priority levels can be assigned to each classification. For example, real-time traffic could be given priority 2 and non-real-time priority 1, where priority 2 is higher (receive better service) than priority 1.

Queue management and congestion control

DigiComm II

Queue management [1]

- Scheduling:
  - which output queue to visit
  - which packet to transmit from output queue
- Queue management:
  - ensuring buffers are available: memory management
  - organising packets within queue
  - **packet dropping when queue is full**
  - **congestion control**

DigiComm II

Schedulers are used to select which packet from which queue will receive service next. Some, such as WFQ may also suggest ways in which queues may be managed under overflow conditions, i.e when there is a scarcity of buffers. However, **queue management** is a topic in its own right, although it is often related closely to the operation of scheduling. Queue management may include several function, including memory management and organisation/ordering of packets within a queue. However the two functions of queue management that we will discuss are **packet dropping** and **congestion control**.

Queue management [2]

- Congestion:
  - misbehaving sources
  - source synchronisation
  - routing instability
  - network failure causing re-routing
  - congestion could hurt many flows: aggregation
- Drop packets:
  - drop “new” packets until queue clears?
  - admit new packets, drop existing packets in queue?

DigiComm II

Congestion can occur for a range of reasons. Sources may be mis-behaving, generating traffic above their normal rate. Even when sources are all behaving, correctly, pathological occurrences such as source synchronisation may cause congestion. Routing instability or network path changes due to network failure close to a node may cause a temporary burst of congestion. The result of congestion could hurt many flows, and this is more so as you move towards the core of the network and find a higher aggregation of traffic.

When congestion occurs, the router has run out of buffers and must drop packets. However, which policy should the router adopt in dropping packets, and what are the effects of using one particular dropping policy over another?

Packet dropping policies

- Drop-from-tail:
  - easy to implement
  - delayed packets at within queue may "expire"
- Drop-from-head:
  - old packets purged first
  - good for real time
  - better for TCP
- Random drop:
  - fair if all sources behaving
  - misbehaving sources more heavily penalised
- Flush queue:
  - drop all packets in queue
  - simple
  - flows should back-off
  - inefficient
- Intelligent drop:
  - based on level 4 information
  - may need a lot of state information
  - should be fairer

DigiComm II

The granularity of the drop algorithm has the same choices as for scheduling. We can decide to drop packets from individual application-level flows, or have a courser grain of dropping. With a fine granularity, the cost, as always is the additional state to be held at the router and the overhead of maintaining and processing that state information. With a courser granularity, we have reduce the amount of state information and lose some control over which flow's packets are dropped. If packets are queued per-flow, then we also offer some protection as a mis-behaving source will only fill and overflow its own queue and not hurt any other flows.

We can decide to drop form the tail of the queue. This is easiest to implement, as it means that there is no need to adjust any pointers in any other part of the queue. However, packets already in the queue and already delayed, and their usefulness may expire while they wait in the queue, i.e. they get to the receiver too late to be used by the application. Yet the receiver may not become aware of the congestion until it sees dropped packets, i.e. until the drop packets at the end of the queue are not received.

So an alternative is drop form the head of the queue. This means that older packets – packets that have been in the network longer – are purged. This is good for real-time data, as theses packets would no longer be as useful anyway. This is also good for protocols like TCP, as the loss is seen sooner and so TCP's congestion control/avoidance mechanisms can operate. However drop from head typically has a higher processing overhead than drop from tail.

We can also drop a packet from a random place in the queue. This should be fair if all sources are behaving. If there are any mis-behaving sources, they are likely to be occupying more buffers and so are more likely have packets dropped than other flows, which is also beneficial.

Another very simple policy to implement is dropping a whole queue. This has the advantage that frees up buffers immediately, and this could be useful if the router is experiencing extreme congestion. However, this policy could be inefficient, as it may lead to many more re-transmission than are necessary, and may compound the burstiness of traffic. The exact behaviour of sources when such an event occurs may be application specific. For real-time applications, to have such a large contiguous hole in the flow may not be acceptable. This may be useful for "reprimanding" mis-behaving flows, when per-flow queuing is used.

Another alternative is intelligent dropping, where packets are dropped by knowledge of the traffic flow, for example knowledge of the level 4 protocol operation. This may require a lot of state information to be held at the router, but appears to have the potential of offer very fine-grained congestion control.

End system reaction to packet drops

- Non-real-time – TCP:
  - packet drop → congestion → slow down transmission
  - slow start → congestion avoidance
  - network is happy!
- Real-time – UDP:
  - packet drop → fill-in at receiver → ??
  - application-level congestion control required
  - flow data rate adaptation not be suited to audio/video?
  - real-time flows may not adapt → hurts adaptive flows
- Queue management could protect adaptive flows:
  - smart queue management required

DigiComm II

We are considering packet dropping to control congestion. However, what do applications do when they see packet loss? TCP uses loss as an indicator of congestion, and slows down its transmission rate. TCP has well-defined behaviour in the form slow-start and congestion avoidance, and generally does a reasonable job of being kind to the network.

Real-time applications, however, use UDP, which has now congestion control mechanisms. The packet loss may be seen at the receiver and he receiver may perform some application-specific action, but there is unlikely to be any feedback to the sender. Some form of application-level congestion control is required. This may require, for example, adapting the flow rate by hanging audio or video coding. However, the application may not e suited to such adaptation. yet, without such adaptations, congestion events may continue to occur. Indeed, adaptive flows (such as TCP) that do back down in the face of congestion my be forced down to low data rates as unconstrained, non-adaptive real-time flows continue unabated.

However, smart queue management (coupled with an appropriate scheduling mechanism) could be used to protect adaptive flows from is-behaving flows whilst at the same time controlling congestion.

RED [1]

- Random Early Detection:
  - spot congestion before it happens
  - drop packet → pre-emptive congestion signal
  - source slows down
  - prevents real congestion
- Which packets to drop?
  - monitor flows
  - cost in state and processing overhead vs. overall performance of the network

DigiComm II

One widely implemented intelligent drop mechanism is Random Early Detection (RED) [#FJ93#]. This drops packets before congestion actually occurs by monitoring queue lengths at a router and increasing the probability of a packet drop as the queue length builds up. The drop of a packet from a flow will act as an indication of congestion and cause it to back of is transmission rate. The packet drop occurs before real congestion starts and is intended to prevent real congestion occurring.

The cost of this mechanism is the requirement to monitor flows and queue lengths, and the processing overhead of dealing with the state information being held. This increase in performance cost is seen as acceptable as the overall performance of the network is improved by avoiding real congestion.

RED [2]

- Probability of packet drop ∝ queue length
- Queue length value – exponential average:
  - smooths reaction to small bursts
  - punishes sustained heavy traffic
- Packets can be dropped or marked as "offending":
  - RED-aware routers more likely to drop offending packets
- Source must be adaptive:
  - OK for TCP
  - real-time traffic → UDP ?

DigiComm II

RED drops a packet from a flow with probability, *p*, which increase linearly as the queue length increases. However, the actual value of queue length used to evaluate *p* is an exponentially-weighted moving average (EWMA) of the actual queue length. This is to allow small bursts of traffic to be allowed through the network, but to spot sustained, heavy traffic. The dampening effect of the EWMA also helps to stabilise the dropping mechanism, so it does not "over-react" to short-lived bursty traffic. As well as dropping packets, RED can mark IP packets as "offending", i.e. in violation of some traffic profile, and not drop them when traffic load is light. However, offending packets can be spotted by RED-aware routers downstream and can sill be dropped if congestion appears.

While RED can control queue length regardless of end-system co-operation, as we have noted already, if the end-system does reduce its transmission rate in response to such congestion signals, this would be helpful to the health of the network as a whole. We have also noted that while dynamically adaptive behaviour does not harm non-real-time applications, real-time applications, such as voice and video applications, normally do not find such behaviour acceptable.

TCP-like adaptation for real-time flows

- Mechanisms like RED require adaptive sources
- How to indicate congestion?
  - packet drop – OK for TCP
  - packet drop – hurts real-time flows
  - use ECN?
- Adaptation mechanisms:
  - layered audio/video codecs
  - TCP is unicast: real-time can be multicast

DigiComm II

In recognition of the usefulness of mechanisms like RED, and the requirement for congestion control in real-time end-systems, there are several proposals to deal with congestions.

The first one we shall examine how to indicate congestion to end-systems. The lost packet is an effective indicator of congestion to TCP, but real-time flows may not be monitoring for packet loss in the same way. There is a proposal to provide **explicit congestion notification (ECN)** [RFC2481] at the IP level. This would be a single-bit flag which would normally be clear, but which would be set by a congested router and so act as an explicit signal to a receiver that this packet has passed through a part of the network that is experiencing congestion.

Other, ore complex mechanisms concentrate on building congestion control into the application-layer protocol. We have already stated that real-time flows such as video do not normally adapt well. However, by using **layered-codecs** we can adjust the transmission rate dynamically. A layered video codec encodes video as several separate flows, each flowing carrying different frequency (spatial) components. There is a single base-layer which must be received, but several additional layers just adder finer detail to the base layer, and can be received optionally depending on the availability of network resources. In [VRC98] is described an experiment to provide TCP-like adaptation for video.

Scheduling and queue management: Discussion

- Fairness and protection:
  - queue overflow
  - congestion feedback from router: packet drop?
- Scalability:
  - granularity of flow
  - speed of operation
- Flow adaptation:
  - non-real time: TCP
  - real-time?
- Aggregation:
  - granularity of control
  - granularity of service
  - amount of router state
  - lack of protection
- Signalling:
  - set-up of router state
  - inform router about a flow
  - explicit congestion notification?

A good analysis of requirements for the future Internet is given in [She95].

Summary

- Scheduling mechanisms
  - work-conserving vs. non-work-conserving
- Scheduling requirements
- Scheduling dimensions
- Queue management
- Congestion control

Integrated services

- Reading:
  - S. Keshav, “An Engineering Approach to Computer Networking”, chapters 6, 9 and 14

Module objectives

Learn and understand about:

- Support for real-time applications:
  - network-layer and transport-layer
- **Quality of service (QoS)**:
  - the needs of real-time applications
  - the provision of QoS support in the network
- Many-to-many communication - **multicast**
- **Integrated Services Network (ISN)**

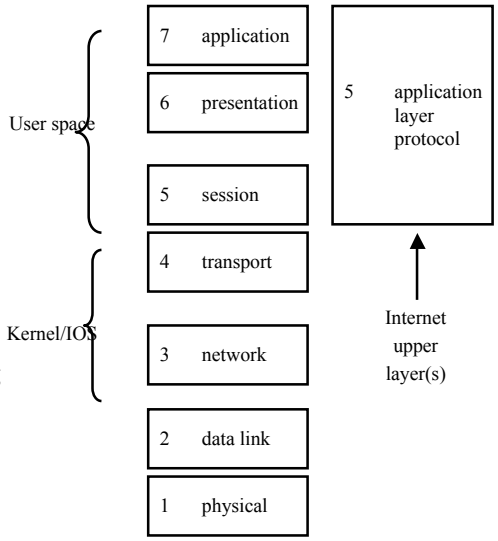
DigiComm II

During the 1990's, applications have become increasingly reliant on the use of the Internet protocols to provide data communications facilities. The use of the Internet protocols seems likely to increase at an extremely rapid rate and the Internet Protocol (IP) will be the dominant data communications protocol in the next decade. IP is being used for a huge variety of "traditional" applications, including e-mail, file transfer and other general non-real-time communication. However, IP is now being used for real-time applications that have **quality of service (QoS)** sensitive data **flows**. A **flow** is a stream of semantically related packets which may have special QoS requirements, e.g. an audio stream or a video stream. Applications such as conferencing (many-to-many communication based on **IP multicast**), telephony – **voice-over-IP (VoIP)** – as well as streaming audio and video are being developed using Internet protocols. The Internet and IP was never designed to handle such traffic and so the Internet community must evolve the network and enhance the Internet protocols in order to cater for the needs of these new and demanding applications. Users wish to have access to a whole plethora of telecommunication and data communication services via the Internet; users wish to access an **Integrated Services Network (ISN)**.

In this set of lectures, we try to understand about QoS for IP-based applications and how the network must be changed to support these new applications.

Support for real-time applications

- Support in the network:
  - routers, routing
- Support at the end-systems:
  - transport protocols
- Support at the application level:
  - user-network signalling
  - application-level signalling and control
- (Link & physical layers?)



DigiComm II

To provide support for real-time applications, we need to introduce mechanisms at many different parts of the communication stack.

At the network layer, we need to modify router behaviour so that packets belonging to QoS sensitive flows receive some kind of preferential treatment, compared to "normal" data packets. We also need to modify the behaviour of routing protocols in order to support multicast communication and QoS-based routing metrics.

At the transport layer, recall that we only have two general protocols: TCP for traditional applications that require an ordered by-stream delivery, and UDP for applications that build in specific control mechanisms at the application layer. For real-time flows, we can identify some general requirements, which we will see can be implemented by extending UDP as in the **Real-time Transport Protocol (RTP)**.

At the application layer, we may identify other mechanisms that are required for specific real-time applications: **floor control** for conference applications; **transcoding** for audio and video flows; **security mechanisms** such as authentication. Although it is possible to identify some general requirements, such higher-layer mechanisms tend to specific to particular applications.

In this set of lectures, we consider the support that we have in the network and at the transport layer, as well as some general issues concerning the interface between the application and the network.

Why do we not consider the link layer and physical layer? Surely these have a fairly vital role in QoS as they provide the transmission capability? Remember that IP tries to hide the lower layers, so although we will se there are important issues concerning the lower layers, we concentrate on the network layer and transport layer.

Real-time flows and the current Internet protocols

DigiComm II

The "problem" with IP [1]

- Data transfer:
  - datagrams: individual packets
  - no recognition of **flows**
  - connectionless: no signalling
- Forwarding:
  - based on per-datagram forwarding table look-ups
  - no examination of "type" of traffic – no **priority** traffic
- Routing:
  - **dynamic routing** changes
  - no "fixed-paths" → no fixed QoS
- Traffic patterns

DigiComm II

Let us first examine the service that IP offers. IP offers a **connectionless** datagram service, giving no guarantees with respect to delivery of data: no assumptions can be made about the delay, jitter or loss that any individual IP datagrams may experience. As IP is a connectionless, datagram service, it does not have the notion of **flows** of datagrams, where many datagrams form a sequence that has some meaning to an applications. For example, an audio application may take 40ms "time-slices" of audio and send them in individual datagrams. The correct sequence and timeliness of datagrams has meaning to the application, but the IP network treats them as individual datagrams with no relationship between them. There is no **signalling** at the IP-level: there is no way to inform the network that it is about to receive traffic with particular handling requirements and no way for IP to tell signal users to back-off when there is congestion.

At IP routers, the forwarding of individual datagrams is based on forwarding tables using simple metrics and (network) destination addresses. There is no examination of the type of traffic that each datagram may contain – all data is treated with equal **priority**. There is no recognition of datagrams that may be carrying data that is sensitive to delay or loss, such as audio and video.

One of the goals of IP was to be robust to network failure. That is why it is a datagram-based system that uses dynamic routing to change network paths in event of router overloads or router failures. This means that there are no fixed paths through the network. It is possible that during a communication session, the IP packets for that session may traverse different network paths. The absence of a fixed path for traffic means that, in practice, it can not be guaranteed that the QoS offered through the network will remain consistent during a communication session.

Even if the path does remain stable, because IP is a totally connectionless datagram traffic, there is no protection of the packets of one flow, from the packets of another. So, the traffic patterns of a particular user's traffic affects traffic of other users that share some or all of the same network path (and perhaps even traffic that does not share the same network path!).

The "problem" with IP [2]

- **Scheduling** in the routers:
  - first come, first serve (FCFS)
  - no examination of "type" of traffic
- No priority traffic:
  - how to mark packets to indicate priority
  - IPv4 ToS not widely used across Internet
- Traffic aggregation:
  - destination address
- (QoS: pricing?)

DigiComm II

At the individual routers, the process of forwarding a packet involves, taking an incoming packet, evaluating its forwarding path, and then sending it to the correct output queue. Packets in output queues are serviced in a simple first-come first-serve (FCFS) order, i.e. the packet at the front of the queue is transmitted first. The ordering of packets for transmission takes the general term on **scheduling**, and we can see FCFS is a very simple scheduling mechanism.

FCFS assumes that all packets have equal priority. However, there is a strong case to instruct the router to give some traffic higher priority than other traffic. For example, it would be useful to give priority to traffic carrying real-time video or voice. How do we distinguish such **priority** traffic from non-priority traffic, such as, say e-mail traffic. The IPv4 **type of service (ToS)** do offer a very rudimentary form of marking traffic, but the semantics of the ToS markings are not very well defined. Subsequently, the ToS field is not widely used across the Internet. However, it can be used effectively across corporate Intranets.

Also, in dealing with traffic that has been marked, we need to be wary of the extra processing a marking scheme may introduce in the core of the network where there is very high **aggregation** of network traffic. With FCFGS, effectively, aggregation is by use of the destination IP address.

Even if we could offer some sort of QoS control mechanism, with **prioritisation** or **traffic differentiation**, there is then the issue of **pricing**. How do we charge for use of network resources for a particular treatment of traffic for a particular customer?

Questions

- Can we do better than **best-effort**?
- What support do real-time flows need in the network?
- What support can we provide in the network?
- Alternatives to FCFS?
- Many-to-many communication?
- Application-level interfaces?
- **Scalability?**

DigiComm II

So we can ask ourselves several questions.

Firstly, can we provide a better service that that which IP currently provides– the so-called best-effort?

The answer to this is actually, "yes", but we need to find out what it is we really want to provide! We have to establish which parameters of a real-time packet flow are important and how we might control them. Once we have established our requirements, we must look at new mechanisms to provide support for these needs in the network itself. We are essentially asking trying to establish alternatives of FCFS for providing better control of packet handling in the network as well as trying to support **multi-party (many-to-many) communication**.

We also need to consider how the applications gain access to such mechanisms, so we must consider any **application-level interface** issues, e.g. is there any interaction between the application and the network and if so, how will this be achieved.

In all our considerations, one of the key points is that of **scalability** – how would our proposals affect (and be affected by) use on a global scale across the Internet as a whole.

Requirements for an ISN [1]

Today's Internet

- IPv4: QoS not specified
- TCP: elastic applications
- Many network technologies:
  - different capabilities
  - no common layer 2
- No support for QoS:
  - ToS in IPv4 – limited use
- QoS requirements:
  - not well understood

Integrated Services Packet Network (ISPN)

- QoS service-level:
  - service type descriptions
- Service interface:
  - signalling
- Admission control:
  - access to resources
- Scheduling:
  - prioritisation and differentiation of traffic

DigiComm II

The Internet was never designed to cope with such a sophisticated demand for services [Cla88] [RFC1958]. Today's Internet is built upon many different underlying network technologies, of different age, capability and complexity. Most of these technologies are unable to cope with such QoS demands. Also, the Internet protocols themselves are not designed to support the wide range of QoS profiles required by the huge plethora of current (and future) applications.

In [CSZ92], the authors speak of the Internet evolving to an **integrated services packet network (ISPN)**, and identify four key components for an Integrated Services architecture for the Internet:

• **service-level:** the nature of the commitment made, e.g. the INTSERV WG has defined **guaranteed** and **controlled-load** service-levels (these are discussed later) and a set of control parameters to describe traffic patterns, which we examine later

• **service interface:** a set of parameters passed between the application and the network in order to invoke a particular QoS service-level, i.e. some sort of **signalling protocol** plus a set of parameter definitions

• **admission control:** for establishing whether or not a service commitment can be honoured before allowing the flow to proceed

• **scheduling mechanisms within the network:** the network must be able to handle packets in accordance with the QoS service requested

A key component that is required here is signalling – talking to the network. Signalling is essential in connection-oriented networks (used for connection control), but datagram network typically need no signalling. No signalling mechanism exists in the IP world – it is not possible to talk to the network, one simply uses the service it provides. The signalling part of a CO network communication offers a natural point at which information about the particular requirements of a connection can be transmitted to the network. As IP is connectionless, any signalling mechanism should that is compatible with current operation of the Internet and should not constrain or change the operation of existing applications and services.

Requirements for an ISN [2]

- QoS service-level:
  - packet handling
  - traffic description
  - policing
  - application flow description
- Service interface:
  - common data structures and parameters
  - signalling protocol
- Admission control:
  - check request can be honoured
- Scheduling:
  - packet classification
  - prioritisation of traffic
  - queue management

DigiComm II

The simple description of the interactions between these components is as follows:

• a **service-level** is defined (e.g. within an administrative domain or, with global scope, by the Internet community). The definition of the service-level includes all the service semantics; descriptions of how packets should be treated within the network, how the application should inject traffic into the network as well as how the service should be policed. Knowledge of the service semantics must be available within routers and within applications

• an application makes a request for service invocation using the **service interface** and a **signalling protocol**. The invocation information includes specific information about the traffic characteristics required for the flow, e.g. data rate. The network will indicate if the service invocation was successful or not, and may also inform the application if there is a service violation, either by the application's use of the service, or if there is a network failure

• before the service invocation can succeed, the network must determine if it has enough resources to accept the service invocation. This is the job of **admission control** that uses the information in the service invocation, plus knowledge about the other service requests it is currently supporting, and determines if it can accept the new request. The admission control function will also be responsible for policing the use of the service, making sure that applications do not use more resources than they have requested. This will typically be implemented within the routers

• once a service invocation has been accepted, the network must employ mechanisms that ensure that the packets within the flow receive the service that has been requested for that flow. This requires the use of **scheduling mechanisms** and **queue management** for flows within the routers

Imagine a video application. The application or user would select a service level and note the traffic characteristics required for the video flow. Information such as required data rate would be encapsulated in a data structure (service interface) that is passed to the network (signalling). The network would make an assessment of the request made by the application and consider if the requirements of the flow can be met (admission control). If they can be met routers would ensure that the flow receives the correct handling in the network (scheduling and queue management).

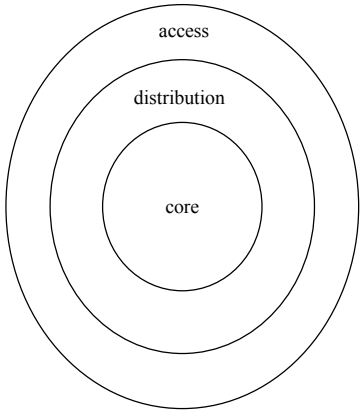
Traffic and QoS parameters

DigiComm II

Network structure [1]

Network hierarchy

- Access network:
  - low multiplexing
  - low volume of traffic
- Distribution network:
  - interconnectivity at local level
  - medium volume of traffic
  - low multiplexing
- Core network – backbone:
  - high volume of traffic
  - high multiplexing



DigiComm II

Let us first examine the problem of traffic. The network consists of several layers of hierarchy with respect to traffic volume and traffic multiplexing.

The outer layer is the **access** network, and can also be considered the edge of the network – closest to the users (applications) which generate the traffic. Here, there is typically a dedicated link to the end-system (for example Ethernet over UTP or residential dial-up). On these links, the level of multiplexing is low and the volume of traffic is comparatively low. If we consider a corporate network, with respect to the Internet, the site network and the corporate's ISP is seen as the access network. For residential users, the ISP network is seen as the access network.

The access network passes on traffic to a **distribution** network. The distribution network may also be called a **transit** network. This network has the job of connecting the access network to the main **core** or **backbone** networks. Typically, end users do no transmit traffic directly onto the distribution network. The distribution network has higher levels of multiplexing and higher volumes of traffic than the access network. However, the level of multiplexing may not be much more than the access network. As an example, several ISPs may use the same transit network (to access the same backbone provider, perhaps). Distribution networks may have a scale that covers a large geographical region, may be as much as a whole country.

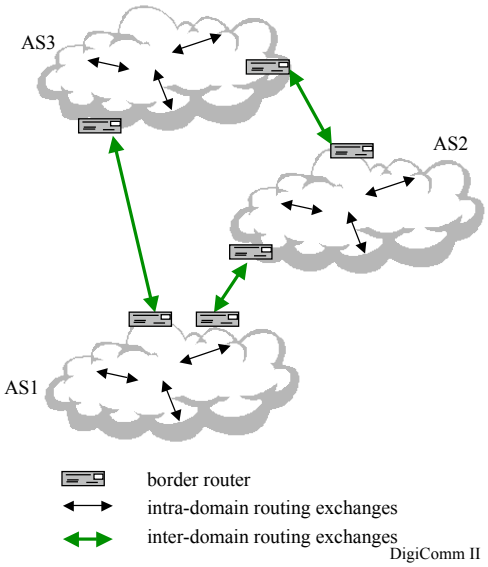
The **core** or **backbone** network is the part of the network that provides international/global interconnectivity. There are in fact many backbone network providers, and the have **peering agreements** to interconnect their networks. Here, there are very high volumes of traffic and very high levels of multiplexing – traffic from millions of users.

This diversity in the volume of traffic and its level of multiplexing has serious implications for any kind of traffic control mechanisms we may wish to apply. We have to remember that if we choose mechanisms that act on a per-packet basis to control traffic, these mechanisms must be scalable in order to maintain performance.

Network structure [2]

Administrative boundaries

- Autonomous system (AS):
  - **intra-domain routing**
  - internal policy
  - routing metric?
  - protocols: RIPv2, OSPFv2
- Interconnection of ASs:
  - **inter-domain routing**
  - interconnectivity information
  - protocols: BGP



This is not the only way of viewing the network. The Internet was formed from the interconnection of many other networks. Today it consists of many network technologies and many network providers all using a standard set of protocols and mechanisms for internetworking. However, each network operator wishes to have control of the construction, operation, management and maintenance of their own network. Indeed the way that routing is organised for the Internet acknowledges these **administrative boundaries** between networks that run as **autonomous systems**.

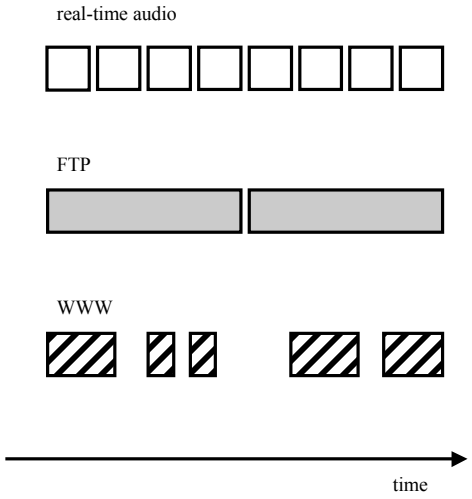
Within an AS, an operator can choose to run their network how they like. They will use an**intra-domain routing** protocol such as RIPv2 or OSPFv2. They will have their own policy for administering the day to day operation of the network, and this policy may well be confidential. The routing metrics that they use may not make sense outside the AS boundary.

Between ASs, different routing protocols – **inter-domain routing** protocols - are used, such as BGP. In fact, no routing metrics are passed between ASs using BGP. rather, **interconnectivity** information is passed,using (effectively) discovery messages to determine connectivity paths with respect toASs.

Given this autonomy of network operators, and the usage of diverse routing information and routing protocols, it is not realistic to agree on common routing metrics and polices for handling traffic on a global basis.

Mixed traffic in the network [1]

- Different applications:
  - traffic (generation) profiles
  - traffic timing constraints
- Routers use FCFS queues:
  - no knowledge of application
  - no knowledge of traffic patterns
- Different traffic types share same network path
- Consider three different applications …



DigiComm II

Different applications generate different kinds of traffic. The traffic has different requirements with respect to a series or **flow** of chunks of data; the size of data chunks that are created, the inter-chunk spacing with respect to time, and the timeliness of delivery of the chunks across the network. These different traffic types are mixed together when transmitted across the network. Remember that routers traditionally use **first-come-first-served (FCFS)** queuing mechanisms. That is, the packets are forwarded by the router in an order determined only by their arrival at router's input lines, and no consideration is made about the type of traffic or the requirements of the application. Remember that datagram forwarding considers each datagram (IPv4 packet) as a separate entity and there is no notion of a **flow** of packets in IPv4. This means that where an application does have a flow of packets, they will not be treated in any special way by the network. We will have a look at some simple (fictional) example applications to demonstrate what happens when the traffic traverses a network.

Firstly, consider a real-time audio application. This may produce a stream of evenly spaced packets, each of which represents, say, a 40ms time slice of audio. The spacing of the audio packets should be maintained at the receiver so that the playout of the audio stream is not "jerky". Also, we would prefer packets not to be reordered or lost. The whole of the IP packet may be no more than about 100bytes in size.

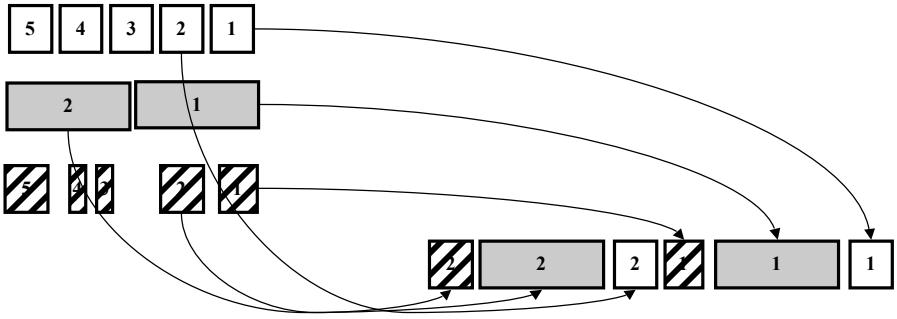
Secondly, let us consider a large file transfer. This will typically create large packets, up to the size of the path MTU (e.g. 1500bytes),in order to make most efficient use of the available network capacity. The packets will be generated as a steady stream by the sender of the file. Although loss of packets could be a problem to the application, reordering and disruption of the packet spacing will not have nay adverse affects to the application.

Thirdly, let us consider someone browsing the web. The traffic generated by the server will depend on the actions of the users of the clients (browsers). There may be relatively large silent periods (no transmissions) while the user is reading a page and the bursts of activities during browsing and download of pages from server to client. The chunks of data are of very variable size, from a few 10's of bytes to several hundred bytes.

A representation of the traffic profiles of these three types of applications is given above. We see that, pictorially, they look very different. Let use consider what happens as these flows pass along parts of the same network path.

Mixed traffic in the network [2]

- Router:
  - 3 input lines: serviced round-robin at router
  - 1 output line (1 output buffer)



Let us examine the processing of our example flows in a very simple router. Let us assume that these three flows arrive at the router on three separate interfaces and will all be forwarded by the route onto the same outgoing link. We assume that the router performs a very simple round-robin servicing of the input queues, taking on packet from each input queue and sending it to the output. We can see how the traffic patterns of our flows are disrupted when the flows are aggregated. This picture may be slightly misleading in that we do not show the relative speeds of the incoming lines and outgoing lines, but if we assume that they all run at the same speed, we see that there disruption to the traffic flow.

Mixed traffic in the network [3]

- Different traffic patterns:
  - different applications
  - many uses of an application
  - different requirements
- Traffic aggregation:
  - core: higher aggregation
  - many different sources
  - hard to model
- Routing/forwarding:
  - destination-based
  - single metric for all traffic
  - queuing effects
- Large packet size:
  - good for general data
  - "router friendly"
  - "slows" real-time traffic
- Small packet size:
  - good for real-time data
  - less end-to-end delay
  - better tolerance to loss
  - (less jitter?)
  - less efficient (overhead)
  - "not router-friendly"

DigiComm II

The mix of traffic is inevitable. There are many different types of applications, all with different requirements for how they might like their packets to be treated within the network. Indeed, a single application may be used differently by different users, so different instances of the same application may produce different traffic flows.

As traffic moves towards the core of the network, there is much higher multiplexing and so a much higher mix and aggregation of traffic. The huge number of sources and the extremely varied patterns of traffic, not only due to the sources but also due to the accumulated queuing effects due to upstream routers, makes it very hard to model traffic on the Internet. (See [PF97] and [WP98].) Remember in the traditional IP-based network, all traffic is treated in the same way, destination-based forwarding, all packets for the same destination pretty much sharing the same forwarding path and all routing information using a single metric.

One thing that does seem evident from the previous slide is that the differences in packet size of the different traffic types play an important role in how mixing of traffic affects individual flows. Large packets are efficient for non-real-time applications. For example, file transfer applications are happy with larger packet sizes, e.g. 1500bytes. However, large packets add delay to smaller packets from other sources (such as real-time audio flows) that they share queues with inside the network.

Smaller packets are much more suited to real-time applications, e.g. a real-time audio application may generate a packet that is of size 100bytes or less. This means that the the packet should have smaller transmission delay and so that flow as a whole has a smaller end-to-end delay. Also, losing a small packet – a small amount of data – is generally better than losing a large packet for a real-time application.

Also recall that the main "cost" of forwarding within an IP-based network is the per-packet cost of making a forwarding decision and then queuing a packet to the output at each router hop, so large packets that transfer lots of data per packet can, in this context, be considered more"router-friendly" than many small packets.

Delay [1]

End-to-end delay

- Propagation:
  - speed-of-light
- Transmission:
  - data rate
- Network elements:
  - buffering (queuing)
  - processing
- End-system processing:
  - application specific

Delay bounds?

- Internet paths:
  - "unknown" paths
  - dynamic routing
- Other traffic:
  - traffic patterns
  - localised traffic
  - "time-of-day" effects
- Deterministic delay:
  - impractical but not impossible

DigiComm II

On factor of great importance to interactive applications is end-to-end delay. This is certainly true for real-time applications using voice or video streams. Across a network such as the Internet, the end-to-end delay is made up of many components.

• **propagation delay:** this is also called "speed-of-light" delay, and is a physical constraint that is linked to the propagation of a physical signal. In general, the speed of light, c, is taken to be approximately 3.0×10^8m/s, and this is often used in calculations. However, it should be noted that in copper the propagation speed of an electrical signal is nearer 2.5×10^8m/s, whilst in fibre the propagation speed of an optical signal is nearer 2.0×10^8m/s.

•**transmission delay:** this is the delay in getting bits onto the wire due to the speed of the link. Do not confuse this with propagation delay. A 1000byte packet is transmitted "faster" on a 10Mb/s line than on a 64Kb/s link, i.e. the bits are placed onto the wire quicker, but the electrical signal is subject to the same propagation delay in both cases.

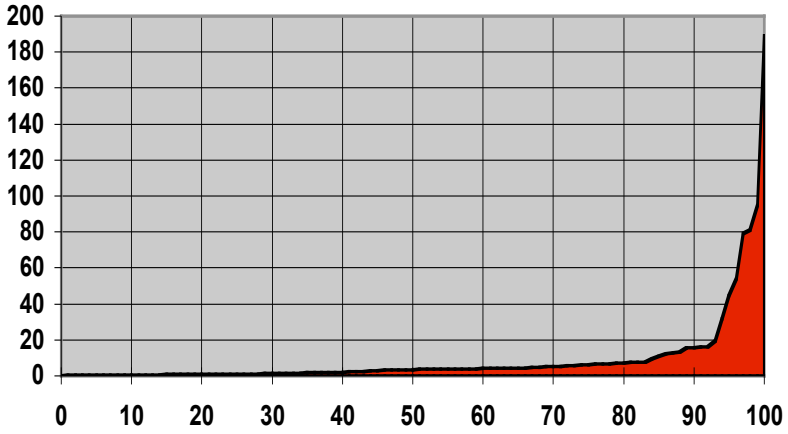
• **network element processing delay:** a packet arriving at a network element may be queued a an input buffer, then it will be read and processed (e.g. forwarding decisions made at routers), and finally queued to an output buffer while it waits to be transmitted.

• **end-system delay:** delay may be introduced at the sender or receiver for various reasons. The input may not be processed immediately or transmitted immediately at the sender, for example due to end-system load. At the receiver, delay may be introduced in order to compensate for network affects, e.g. the use of de-jitter buffers in real-time audio tools.

The end-to-end path that traffic follows across the Internet is never fixed for any application. In general, the application neither knows or cares about the actual end-to-end path. Changes to the the path occur due to the dynamic nature of the IP routing protocols, and do not forget that paths may not be symmetric delay may be asymmetric. Traffic patterns may be observed to have effects that are localised, e.g. localised congestion, as well as "time-of-day" effects.

(See [PF97a] and [PF97b] for a discussion of observed effects of routing and packet dynamics.)

Delay [2] #picture#



DigiComm II

Jitter (delay jitter) [1]

End-to-end jitter

- Variation in delay:
  - per-packet delay changes
- Effects at receiver:
  - variable packet arrival rate
  - variable data rate for flow
- Non-real-time:
  - no problem
- Real-time:
  - need jitter compensation

Causes of jitter

- Media access (LAN)
- FIFO queuing:
  - no notion of a flow
  - (non-FIFO queuing)
- Traffic aggregation:
  - different applications
- Load on routers:
  - busy routers
  - localised load/congestion
- Routing:
  - dynamic path changes

DigiComm II

Jitter is the **delay variation** observed at the receiver. Packets do not arrive with constant delay so the timing of packet generation at the sender is perturbed and timing needs to be re-constructed at the receiver – this is called synchronisation. The effects at the receiver are application dependent, but what is visible is a variable packet arrival rate, and therefore a variable data rate for the flow. This is not suitable for application such as audio which produce flows that may have a constant data rate. For non-real-time applications, jitter is typically not an issue. For real-time applications, jitter compensation needs to be applied at the receiver.

Jitter is caused by a number of factors. At the sender, use of LAN technology like Ethernet may lead to packet transmissions being unevenly spaced. In routers, the use of FIFO queuing and traffic aggregation may lead to packet spacing being perturbed. Some routers may also use non-FIFO techniques, perhaps prioritising certain traffic (e.g. policy-based, using source address), and so disrupting the normal spacing of other flows. Traffic aggregation, with many different size packets sharing the same buffers/output-link may also cause jitter.

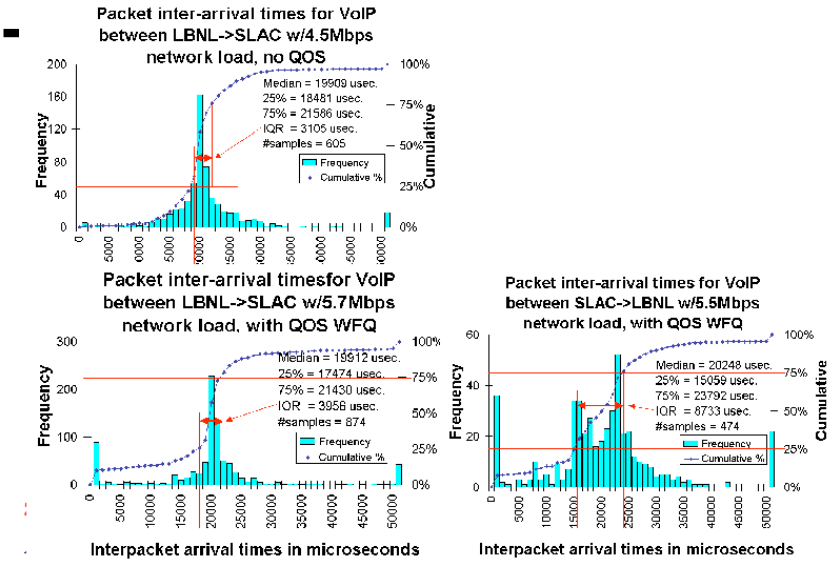
As router load increases, buffers/queues in routers may start to fill up, adding queuing delay. Very busy routers may lead to (localised) congestion, and this may lead to further delay. Where congestion or router failure leads to dynamic routing changes, packets may find themselves traversing different network paths between the same source and destination. This causes delay variation. Congestion in the network can lead to routing instability, and **route flapping** – dynamic changes routes from A → B → A – may occur.

Jitter (delay jitter) [2] #picture#

- Easiest measure is the variance in inter-packet delay
- Can use lots of other metrics (e.g. other moments of the inter-arrival time distribution
- Not critical to protocols like TCP unless jitter is 1st order (I.e. not 2nd order) effect
- Critical to voice
- (e.g. playout buffer to make sure D2A device or display is not starved…)
- VOIP, Video over IP
- Critical for interactive

DigiComm II

VoIP jitter LBNL<=>SLAC with load



DigiComm II

Loss [1]

End-to-end loss

- Non-real-time:
  - re-transmission, e.g.: TCP
- Real-time:
  - forward error correction and redundant encoding
  - media specific "fill-in" at receiver
- Adaptive applications:
  - adjust flow construction

Causes of loss

- Packet-drop at routers:
  - congestion
- Traffic violations:
  - mis-behaving sources
  - source synchronisation
- Excessive load due to:
  - failure in another part of the network
  - abnormal traffic patterns, e.g. "new download"
- Packet re-ordering may be seen as loss

DigiComm II

Loss generally occurs because of congestion somewhere in the network. Congestion at an individual router occurs when it does not have enough resources to deal with the number if incoming packets and so has to discard packets. For non-real-time applications, loss is normally not a problem. For example, applications using TCP rely on TCP's re-transmission mechanisms and congestion control to ensure that data does get through. However, retransmission schemes are generally unsuitable for real-time traffic, and receiver-side mechanisms (such as forward error correction, redundant encoding and "fill-in" schemes) are used.

Loss is caused by packet drop ate routers, Router do not have enough input-buffer space or output-buffer space to deal with the number of packets they have received and some must discard some packets. Such congestion can occur at at traffic aggregation points (perhaps due to insufficient provisioning) or at the ingress to "busy" network/server sites. Traffic violations from sources may cause congestion, if traffic ingress is not policed. In some cases, even with well behaving, policed sources, there may still be congestion if many sources go to "peak" load at the same time – source synchronisation.

Excessive load at a point in the network maybe due to under-provisioning, traffic being re-routed because of a failure elsewhere in the network or sometimes due to abnormal traffic patterns, for example due to many people downloading the latest piece of software from a popular server.

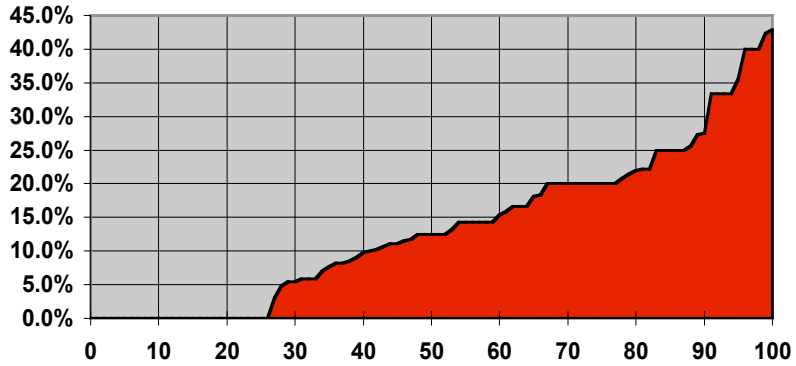
Another important effect to note is that, as viewed from the receiver, certain modes of packet re-ordering may seem like loss to the user. For example, consider an application that is waiting to receive packets A, B and C. A and C arrive, and B has not been lost but just re-ordered and will arrive shortly. However, the application can not wait and so perhaps will "fill-in" or correct fro the absence of B in an application-specific manner.

Loss [2] #picture#

- Varies with route
- Depends on link quality
- Depends on router quality
- Varies with time
- Depends on other load
- And interference

DigiComm II

Error rates


45.0%
40.0%
35.0%
30.0%
25.0%
20.0%
15.0%
10.0%
5.0%
0.0%
0
10
20
30
40
50
60
70
80
90
100

Data rate [1]

End-to-end data rate

- Short-term changes:
  - during the life-time of a flow, seconds
- Long-term changes:
  - during the course of a day, hours
- Protocol behaviour:
  - e.g. TCP congestion control (and flow control)

Data-rate changes

- Network path:
  - different connectivity
- Routing:
  - dynamic routing
- Congestion:
  - network load – loss
  - correlation with loss and/or delay?
- Traffic aggregation:
  - other users
  - (time of day)

DigiComm II

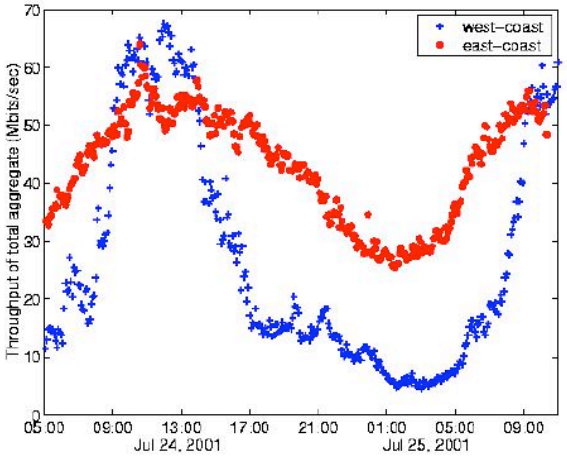
The end-to-end data rate across the Internet varies tremendously. Changes are observable at just about every timescale, from second to hours. There are absolutely no guarantees with respect to data rate, and fluctuations can occur at any time. These fluctuations are typically as the result of the complex interaction of network elements and traffic. At the application-level they may also be seen due to the action of transport protocols (or application-specific control), for example TCP congestion control. (note that in TCP, although flow control also affects the end-to-end data rate, this is due to the action of the receiver and not the network, so we do not consider it here.)

Data rate changes are due to similar reasons as for delay and loss: network connectivity, dynamic changes in routing resulting in a change of the end-to-end path, changes network traffic leading to generally higher network load and congestion. The change in value of the end-to-end data rate may often be closely correlated with the changes in end-to-end delay and/or observed loss, but this is not a general rule.

Data rate changes can be highly visible as time-of-day effects due the the network usage of other users. For example, from the UK try browsing a WWW server on the west coast of the US early on a Sunday morning and you should see a good data rate. Try the same server again at 3.00pm on a Monday afternoon and you will notice a huge difference in the end-to-end data rate.

Data rate [2] #picture#

- Link is multiplexed so rate is not just link speed
- Varies with overall load depending on the link sharing rules
- Note latency (RTT) contributed to throughput (e.g. if you have to wait for acks…)
- Lots of different possibile basic link speeds depending on media, modulation, and protocol stack overheads
- "Goodput" is often used for the residual throughput after you allow for all overheads (including retransmissions)


west-coast
east-coast
Throughput of total aggregate (Mbits/sec)
0
10
20
30
40
50
60
70
05:00
09:00
13:00
17:00
21:00
01:00
05:00
09:00
Jul 24, 2001
Jul 25, 2001

Network probing: a quick note

- Can use probes to detect:
  - delay
  - jitter
  - loss
  - data rate
- Use of network probes:
  - ping
  - traceroute
  - pathchar
- Probes load the network, i.e the affect the system being measured
- Measurement is tricky!
- See:
  - www.caida.org
  - www.nlanr.net

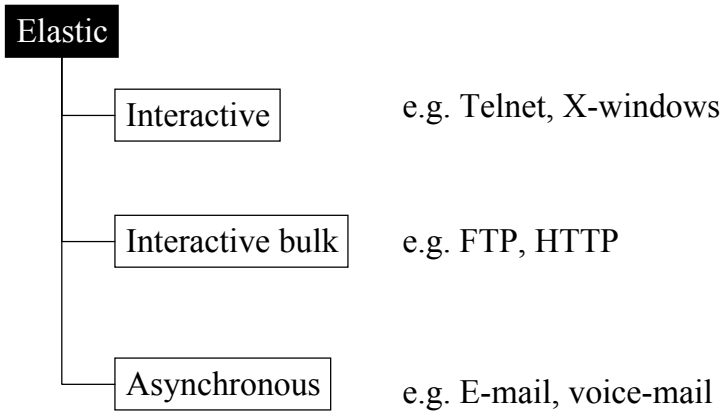
DigiComm II

It is possible to using probe techniques to determine the value of certain parameters on an end-to-end path. Probing techniques work by an application generating, transmitting and receiving specially formatted packets, e.g. ICMP packets as used by ping, traceroute and pathchar. These specially formatted packets are sometimes called probes. They travel along the network path and the application can determine the values of certain parameters by looking the the response to the probe from the network. Note that in order to do this, the application is actually loading the network– it introduces additional traffic onto the network – in order to determine what is happening. Measuring performance and QoS in IP-based networks and on the Internet as a whole is still a research issue. Many other tools and techniques are available, and you can find information about some of them at:

• Cooperative Association for Internet Data Analysis: http://www.caida.org/

• National Laboratory for Applied Network Research: http://www.nlanr.net/

Elastic applications



DigiComm II

Elastic applications are those applications that can tolerate relatively large delay variance– essentially the traditional data applications. They work best with low delays but they do not become unusable when delays increase or vary somewhat. Throughput may or may not be an issue. The basic requirement for these applications generally is that that they receive a reliable, ordered end-to-end data delivery service.

Interactive applications require near real-time, human interaction and ideally would like to have delays of 200ms or less, but could possibly tolerate slightly more.

Interactive bulk applications will also have similar requirements for delay, but they may also have high throughput requirements. Generally, the user is willing to tolerate a delay that is roughly proportional to the volume of data being transferred.

Asynchronous applications may or may not involve bulk data (e.g. e-mail), but they are prepared to tolerate much greater delay as they are generally store-and-forward type applications and the user does not expect anything close to real-time interaction.

Examples of elastic applications

- E-mail:
  - asynchronous
  - message is not real-time
  - delivery in several minutes is acceptable
- File transfer:
  - interactive service
  - require "quick" transfer
  - "slow" transfer acceptable
- Network file service:
  - interactive service
  - similar to file transfer
  - fast response required
  - (usually over LAN)
- WWW:
  - interactive
  - file access mechanism(!)
  - fast response required
  - QoS sensitive content on WWW pages

DigiComm II

Examples of "traditional" datacomms services are e-mail, file transfer, network file services and WWW.

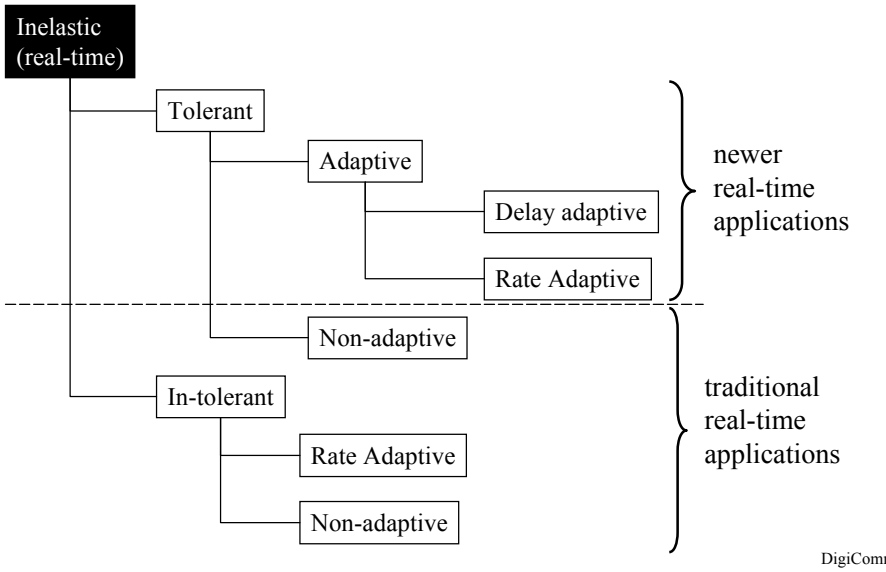
E-mail is the least demanding of these services in terms of timeliness and interaction. E-mail messages can be read and prepared off-line (without a network connection). The network connection is required only when sending or receiving messages. When sending messages, it is sufficient that the e-mail service manages to delver your mail "sometime soon". Typically this is the order of several minutes (or less). However, in many cases, even a delivery time of several hours is sufficient. Asynchronous means (literally) that there is no synchronisation between the sender and receiver of an e-mail message.

With file transfer, what is required is that you can retrieve or send files to a remote server. Generally, the user must somehow interact with the server so some timeliness of response to commands is required. When a file is transferred, the user would like to have the transfer take place "as quickly as possible" but the user's value in use of the service is not severely depreciated if the transfer happens to be slightly "slower".

Network file service can be though of as an enhanced version of file transfer. Here, files that are on a remote server are manipulated as if they were local to the end-system at the which the user is physically situated. Of course, a fast response to commands and fast file transfer are vital to this service. This is typically not a problem as network file services are normally run within an office environment where there is a high data rate available. Again, "slower" transfers may be acceptable but the service would still perform a useful function.

The WWW is essentially an interactive, non-real-time service. Web pages are documents that can be access from a remote server. In some ways, there are elements similar to file transfer and network file service here. However, the interaction is very important here – web pages should appear quickly and hyperlinks that are selected ("clicked") should be responded to quickly also. Web pages often now contain links to other content such as streaming audio and video. This is not really interactive but may have fairly strict requirements with respect to QoS (e.g. loss, available data rates, etc.).

Inelastic applications



Inelastic applications are comparatively intolerant to delay, delay variance, throughput variance and errors. If QoS is not sufficiently well controlled, the applications become unusable.

Tolerant applications can be adaptive or non-adaptive. Tolerant adaptive applications may be able to:

• adapt to delay variation: a voice application might adapt to a decrease in the average delay by dropping a few packets to allow the transmitter to catch up with the receiver.

• adapt to data rate variation: a video application may be able to adjust the encoding and trade quality against throughput.

Tolerant non-adaptive applications cannot adapt – but can still tolerate – some QoS variation. A voice application may still be usable with a measure of packet loss.

Intolerant applications cannot tolerate QoS variation – for example real-time control of a robot arm. Some may be rate-adaptive being able to adjust to detected changes in throughput, but others are totally non-adaptive.

Examples of inelastic applications

- Streaming voice:
  - not interactive
  - end-to-end delay not important
  - end-to-end jitter not important
  - data rate and loss very important
- Real-time voice:
  - person-to-person
  - interactive
- Important to control:
  - end-to-end delay
  - end-to-end jitter
  - end-to-end loss
  - end-to-end data rate

DigiComm II

Inelastic applications are generally those that involve some sort of QoS sensitive media, such as voice.

If we have a look first at non-real-time voice, e.g. streaming audio. Here, large end-to-end delay and delay variation – **jitter** – does not matter greatly as the media flow is not interactive. There are well-known mechanisms that can adjust for jitter at the application-level. What is important is that a certain data rate is maintained that there is relatively low packet loss.

For real-time audio, e.g. person-to-person, the data rate and packet loss remain important but the end-to-end delay and end-to-end jitter are now quite important. There should be a constant and flowing dialogue in order for human voice interaction to work. If the delay is large (e.g. over half a second), conversation becomes difficult. (You may have experienced this sometimes on very-long distance phone calls, e.g. to Asia or Australasia from the UK.)

In fact, there are similar requirements for streaming and real-time video, respectively, as the are for streaming and real-time voice. However, humans tend to be much less tolerant to packet loss and variations in data rate in video traffic than in voice traffic.

Notice also that the requirements for **real-time** voice and video are different from those of **streaming** voice/video. For real-time voice and video, we assume that the media streams are being created in real-time and are being used in an interactive manner, e.g. a **conversation**. Applications that are providing conversational services have more stringent QoS constraints than applications that are processing streaming media (e.g. voice/video playback). The latter can cope (to some extent) with variations in data rate, packet loss rates and delay.

QoS parameters for the Internet [1]

Delay

- Not possible to request maximum delay value
- No control over end-to-end network path
- Possible to find actual values for:
  - maximum end-to-end delay, D_{MAX}
  - minimum end-to-end delay, D_{MIN}

Jitter

- Not possible to request end-to-end jitter value
- Approximate maximum jitter:
  - $D_{MAX} - D_{MIN}$
  - evaluate D_{MIN} dynamically
  - D_{MAX}? 99^{th} percentile?
- Jitter value:
  - transport-level info
  - application-level info

DigiComm II

In general, requests for particular delay and jitter constraints by an application are not parameters that can be honoured across the Internet. That is not to say they could not be honoured by an IP-based network, e.g. it might be possible to arrange for such mechanisms in an IP-based corporate VPN. As we have noted, there are many autonomous systems that are linked together to form the Internet and it is not possible implement a homogenous environment across them. So, typically, it may be possible for an application to request "low delay" and "low jitter" and it may be possible for the network to give an indication of what the delay/jitter is along a particular network path, e.g. a value for maximum end-to-end delay can be found with the use of RSVP/INTSERV as we will see later.

So, it may be possible to find the maximum possible delay bound by querying network elements along a certain end-to-end path, and it should be possible to evaluate current (and mean) delay by using network probes (a "ping"-type of scheme). However, jitter is very much harder to evaluate. This, once again, is because the end-to-end path may consist of concatenation of various network technologies and routers with different behaviour. However, with an upper bound for the delay, D_{MAX}, and a lower bound for delay, D_{MIN}, it could be argued that a reasonable estimate for the jitter is $D_{MAX} - D_{MIN}$. D_{MIN} can be evaluated dynamically during the operation of the flow, perhaps from application-level header information or from protocol headers as in RTP/RTCP (we look at RTP/RTCP later).

D_{MAX} may also be evaluated dynamically, form the same measurements used to determine D_{MIN}, however, it may be that such an estimate for D_{MAX} results in an estimate that is too conservative. Other summaries may be used, e.g. the 99^{th} percentile for the measured delay values. Evaluations of both D_{MIN} and D_{MAX} are likely to be application specific.

QoS parameters for the Internet [2]

Loss

- Not really a QoS parameter for IP networks
- How does router honour request?
- Linked to data rate:
  - hard guarantee?
  - probabilistic?
  - best effort?
- (Traffic management and congestion control)

Packet size

- Restriction: path MTU
- May be used by routers:
  - buffer allocation
  - delay evaluation

DigiComm II

It should be possible to specify loss characteristics as QoS parameters. However, it may not be practical to ask for a particular numerical value for loss, e.g. asking for 10^{-4} packet loss rate. Note that packet loss decreases the end-to-end data rate. So, specification of loss may be linked to the specification of how a data rate quest if honoured. For example, one could ask for a data rate to be guaranteed (no loss), have "low" loss (low probability of packet loss) or best effort. One reason for not using loss rates is that monitoring per-flow loss rates is taken to be a computationally expensive exercise for routers to perform in practise. So, in IP-based networks such as the Internet, packet loss rate is not usually specified numerically. (In ATM networks, or Frame Relay networks, numerical values for cell loss rates and frame loss rates may be specified, respectively.)

Although packet size may not be considered specifically as a QoS parameter, the maximum (and perhaps the minimum) packet size that an application will generate may be specified. Of course the maximum packet size will be restricted by the path MTU, and real-time applications generally do not exceed the path MTU as IP-packet fragmentation is normally considered harmful for real-time applications. Routers may find the specification of maximum and minimum packet size useful, e.g. for the Guaranteed Service in INTSERV as we will see later.

QoS parameters for the Internet [3]

- Data rate:
  - how to specify?
- Data applications are bursty:

$$\frac{\text{peak data rate}}{\text{mean data rate}} >> 1$$

- Specify mean data rate?
  - peak traffic?
- Specify peak data rate?
  - waste resources?
- Real-time flows:
  - may be constant bit rate
  - can be variable bit rate
- Application-level flow:
  - **application data unit (ADU)**
- Data rate specification:
  - application-friendly
  - technology neutral

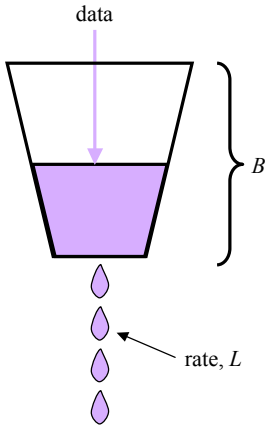
DigiComm II

For many IP-based application, the most precious resource is data rate. So, specification of data rate as a QoS parameter is essential. However, how is data rate to be specified? remember that traditional data applications may produce a very varied traffic profile. Also, while we have generally considered real-time applications as producing constant bit-rate flows, more sophisticated audio and video coding techniques produce variable bit rates. So do we specify mean rates, and then lose data when there are peaks above the mean? Or do we specify peak rates, and then make inefficient use of the network resources when we operate below the peak rate?

Multimedia applications generate a lump of data that can be considered as an **application data unit (ADU)**. We need to have a specification mechanism can reflect the way that the application may generate data. It should also be something that is technology neutral, as IP itself is. That is, the specification should not rely on any particular mechanisms or functions in levels 1 (physical) and 2 (data link).

Leaky bucket

- Two parameters:
  - *B*: bucket size [Bytes]
  - *L*: leak rate [B/s or b/s]
- Data pours into the bucket and is leaked out
- *B*/*L* is maximum latency at transmission
- Traffic always constrained to rate *L*



DigiComm II

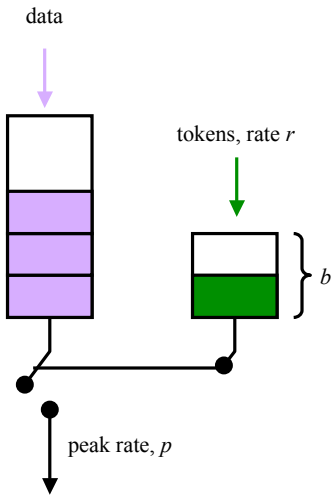
A traffic control mechanism used by ATM is leaky bucket. Here, there is a fixed data rate, effectively the peak rate of transmission, which is *L*. However, to cope with bursts of data, a bucket size, *B*, is also defined. The model is that data arrives in the bucket and is drained from the bucket at the rate *L*. Bursts of traffic that result in an overflow of the bucket are discarded.

Note that the bucket may be filled in ADU sized lumps, so it is probably sensible to arrange for the *B* to be a a multiple of the transmitted ADU size. The maximum latency introduced by the bucket is *B*/*L*. Note that the traffic is always constrained to rate *L* by the leaky bucket, even if greater capacity is available, i.e. a higher peak rate may be possible at a given time. Leaky bucket is used in ATM, where *B* and *L* be specified in units of cells and cells per second, respectively.

Token bucket

Token bucket

- Three parameters:
  - *b*: bucket size [B]
  - *r*: bucket rate [B/s or b/s]
  - *p*: peak rate [B/s or b/s]
- Bucket fills with tokens at rate *r*, starts full
- Presence of tokens allow data transmission
- Burst allowed at rate *p*
- data sent < *r*t + *b*



DigiComm II

There is a subtle, and very important, difference between a leaky bucket and a token bucket. The token bucket also has a bucket size, *b*, and a bucket rate, *r*, but it allows traffic bursts to be transmitted at peak rate, *p*, under certain conditions. In this case, the bucket does not "fill with data" as it does in the leaky bucket, but it fills with tokens, that that allow data to be transmitted. Data can only be transmitted when there are enough token to allow transmission to take place. Transmission can then take place at a peak rate of *p*. Nominally, data is transmitted at a rate *r*, the same rate at which the bucket is filled with tokens. However, it can be seen that bursts of traffic, up to the bucket size, can be transmitted at the peak rate, *p*.

Real-time media flows

DigiComm II

Interactive, real-time media flows

- Audio/video flows:
  - streaming audio/video
  - use buffering at receiver
- Interactive real-time:
  - only limited receiver buffering
  - delay <200ms
  - jitter <200ms
  - keep loss low
- Effects of loss:
  - depend on application, media, and user
- Audio:
  - humans tolerant of "bad" audio for speech
  - humans like "good" audio for entertainment
- Video:
  - humans tolerant of "low" quality video for business
  - humans like "high" quality video for entertainment
- Audio – video sync:
  - separate flows?

DigiComm II

People often think of real-time flows as "audio and video". We must make a distinction between **streamed** audio and video flows and **real-time**, interactive audio and video flows. For streamed audio and video, while low delay, low jitter, low loss and high data rate are desirable for playback of the flow, under normal circumstances, only the latter (data rate) is significantly important that it might be considered a QoS issue. We do require large amounts of capacity to transmit high quality streamed audio and video. However, end-to-end delay is not an issue, even if this may be in the order of a few seconds but for one –way transmission - playback - it should not matter. Jitter and loss can be compensated having a large buffer at the receiver for delaying playback at the receiver to allow time for smoothing unevenly spaced packet arrival as well as dealing with re-transmissions of lost data as required. This scenario may be complicated in a multicast scenario (reliable multicast is a tricky issue as we will see later).

Real-time audio and video normally involves humans as the subjects of audio/video flows. Humans use applications that generate audio and video flows to interact across a network. Small"timeslices" of audio/video are placed into packets and then sent across a network. Generally, for human interaction to continue in a "conversational" manner, delay and jitter must be kept to around 200ms each, i.e. the maximum end delay should be no more than around 400ms. The rules for loss are less easy to specify so easily, as loss effects tend to be application-specific, media-specific and user-specific. For example, let us consider a fictitious video-telephone (VT) application being used by a human on a laptop machine, ad the user is travelling away from home. That user may find it appropriate to use only low quality audio when talking to someone back at the office to check on deliveries, then use high quality audio (and perhaps some video) when giving a report to his/her boss, and use high quality and use high quality video and audio when talking to his family. Also,hu,mans are generally tolerant of low quality audio and video for "normal" (business or domestic) usage, but for entertainment, they generally want high quality audio and video. Some other applications, e.g. medical applications, may require high quality audio and video at all times.

Also, if audio and video flows are transmitted separately, then the receiver may need to re-synchronise the flows for playback.

Audio

QoS requirements

- Delay < 400ms:
  - including jitter
- Low loss preferable:
  - loss tolerant encodings exist
- Data rates:
  - speech ≤ 64Kb/s
  - "good" music ≥ 128Kb/s

- Time domain sampling
- Example – packet voice:
  - 64Kb/s PCM encoding
  - 8-bit samples
  - 8000 samples per second
  - 40ms time slices of audio
  - 320 bytes audio per packet
  - 48 bytes overhead
    (20 bytes IP header)
    (8 bytes UDP header)
    (20 bytes RTP header)
  - 73.6Kb/s

DigiComm II

We discuss briefly the QoS requirements for real-time audio. For audio, we normally require that delay is less than 400ms end-to-end for any packet, and this includes any jitter. It is preferable to have low loss, but there are mechanisms to compensate for loss, either by use of special audio encoding techniques, or by "fill-in" techniques at the receiver. Telephone quality speech is achievable in audio encodings of 64Kb/s or less. Reasonable quality music (for entertainment) may require at least twice this rate.

Audio is taken to be a single dimensioned variable, and is normally sampled in the time domain.

When it is transmitted as packet voice over IP, a typical dat rate required for the application may be approximately 74Kb/s.

Example audio encoding techniques

G.711
- PCM (non-linear)
- 4KHz bandwidth
- 64Kb/s

G.722
- SB-ADPCM
- 48/56/64Kb/s
- 4-8KHz bandwidth

G.728
- LD-CELP
- 4KHz bandwidth
- 16Kb/s

G.729
- CS-ACELP
- 4KHz bandwidth
- 8Kb/s

G.723.1
- MP-MLQ
- 5.3/6.3Kb/s
- 4KHz bandwidth

GSM
- RPE/LTP
- 4KHz
- 13Kb/s

DigiComm II

LD-CELP:	low delay code excited linear prediction
CS-ACELP:	conjugate structure algebraic excited linear prediction
ML-MLQ:	multi-pulse maximum likelihood quantisation
PCM:	pulse code modulation
RPE/LTP:	residual pulse excitation/long term prediction
SB-ADPCM:	sub-band adaptive differential pulse code modulation

Video

QoS requirements

- Delay < 400ms:
  - including jitter
  - same as audio
  - inter-flow sync
- Loss must be low
- Data rate – depends on:
  - frame size
  - colour depth
  - frame rate
  - encoding
- Frequency domain:
  - discrete cosine transform (DCT)
- Example - packet video:
  - ###

DigiComm II

We discuss briefly the QoS requirements for video. Loss and delay on video has a much more significant affect than on audio, especially if there is significant compression. Remember that compression removes redundancy, the very thing that we need for robustness in the face of loss. Typically, a whole screen of data may not fit into a single packet and so multiple packets may be required to make up a single frame of video. This means that it is possible for part of a picture to go missing. However, burst errors (loss of several packets close in sequence) may mean that several parts of the same picture could go missing.

Example video encoding techniques

MPEG1

- upto 1.5Mb/s

MPEG2

- upto 10Mb/s (HDTV quality)

MPEG4

- 5-64Kb/s (mobile, PSTN)
- 2Mb/s (TV quality)
- MPEG7, MPEG21

H.261 and H.263

- $n \times 64$Kb/s, $1 \leq n \leq 30$

DigiComm II

MPEG moving pictures expert group

Commonly used picture sizes are given below.

Picture format	**Image size**	
sub-QCIF	128x96	
QCIF	176x144	
CIF	352x288	
4CIF	702x576	(full screen)
16CIF	1408x1152	

CIF common intermediate format
QCIF quarter CIF

Summary

- IPv4 and current Internet:
  - not designed for QoS support
- Need to add support for ISN:
  - service definitions
  - signalling
  - update routers
- Need to describe traffic:
  - QoS parameters
- Audio and video have different requirements

QoS services and application-level service interfaces

IP “service”

- IP datagram service:
  - datagrams are subject to loss, delay, jitter, mis-ordering
- Performance: no guarantees
- **Integrated Services:**
  - new QoS service-levels
- **Differentiated Services:**
  - class of service (CoS)
- User/application may need to signal network
- User/application may need to signal other parts of application

DigiComm II-2

Internet users have increasing demands to use a range of multimedia applications with QoS sensitive data flows. All these applications may require different QoS guarantees to be provided by the underlying network. An e-mail application can make do with a best-effort network service. Interactive or real-time voice and video applications require (some or all of) delay, jitter, loss and throughput guarantees in order to function. Web access can make do with a best-effort service, but typically requires low delay, and may require high throughput depending on the content being accessed. The Internet was never designed to cope with such a sophisticated demand for services. Today’s Internet is built upon many different underlying network technologies, of different age, capability and complexity. Most of these technologies are unable to cope with such QoS demands. Also, the Internet protocols themselves are not designed to support the wide range of QoS profiles required by the huge plethora of current (and future) applications.

To deal with such issues, the IETF have two working groups looking at QoS issues directly. The INTSERV WG is looking at how to extend the IP network to become an **Integrated Services Network (ISN)**. INTSERV have been looking at the definition and support of new **service-levels** (other than best-effort) within an IP network. The DIFFSERV WG is looking at the provision of **differentiated services** within IP networks, allowing service providers to treat packets from different sources with different QoS. The source of packets is defined administratively, e.g. could be a single host or a whole organisation.

In some cases, the user or the application (or both) may need to indicate their requirements to the network by use of some sort of signalling. The notion of user-to-network signalling is not inherent in the IP networking model. At the application-level, there may be a requirement for user-to-user (application-to-application) supported as a network service.

Questions

- Can we do better than **best-effort**?
- What support do real-time flows need in the network?
- What support can we provide in the network?
- QoS for many-to-many communication?
- Application-level interfaces?
- Signalling

DigiComm II-3

So we can ask ourselves several questions.

Firstly, can we provide a better service that that which IP currently provides– the so-called best-effort?

The answer to this is actually, "yes", but we need to find out what it is we really want to provide! We have to establish which parameters of a real-time packet flow are important and how we might control them. Once we have established our requirements, we must look at new mechanisms to provide support for these needs in the network itself. We are essentially asking trying to establish alternatives to FCFS for providing better control of packet handling in the network as well as trying to support QoS for **multi-party (many-to-many) communication**.

We also need to consider how the applications gain access to such mechanisms, so we must consider any **application-level interface** issues, e.g. is there any interaction between the application and the network and if so, how will this be achieved.

IP, as connectionless network protocol, involves no signalling. However, in order to allow the use of real-time applications, we need to establish a richer set of function in order to allow information about a communication session to sent into (and across) the network. So, we need signalling capability, both user-to-network and user-to-user.

INTSERV

Questions

- What support do we need form the network to give QoS capability to the Transport layer?
- How can we control congestion in the network?
- How can we support legacy network protocols over the Internet?

DigiComm II-5

In the last section we produced a taxonomy of applications with respect to their QoS requirements. Real-time applications need a better service that standard IP unreliable datagram delivery. We will see that there is some support available at the transport layer for real-time applications (e.g. by use of RTP), but this can not give us QoS guarantees. We need direct support form the network so we must modify the operation of the routers somehow so that they can give priority to QoS sensitive traffic. Finally, we take a brief look at how legacy applications might be operated across an IP-based network.

Integrated services

- Need:
  1. service-levels
  2. service interface – signalling protocol
  3. admission control
  4. scheduling and queue management in routers
- Scenario:
  - application defines service-level
  - tells network using signalling
  - network applies admission control, checks if reservation is possible
  - routers allocate and control resource in order to honour request

DigiComm II-6

To provide Integrated Services for IP applications, we can envisage the following scenario:

• a **service-level** is defined (e.g. within an administrative domain or, with global scope, by the Internet community). The definition of the service-level includes all the service semantics; descriptions of how packets should be treated within the network, how the application should inject traffic into the network as well as how the service should be policed. Knowledge of the service semantics must be available within routers and within applications

• an application makes a request for service invocation using the **service interface** and a **signalling protocol**. The invocation information includes specific information about the traffic characteristics required for the flow, e.g. data rate. The network will indicate if the service invocation was successful or not, and may also inform the application if there is a service violation, either by the application's use of the service, or if there is a network failure

• before the service invocation can succeed, the network must determine if it has enough resources to accept the service invocation. This is the job of **admission control** that uses the information in the service invocation, plus knowledge about the other service requests it is currently supporting, and determines if it can accept the new request. The admission control function will also be responsible for policing the use of the service, making sure that applications do not use more resources than they have requested. This will typically be implemented within the routers

• once a service invocation has been accepted, the network must employ mechanisms that ensure that the packets within the flow receive the service that has been requested for that flow. This requires the use of **scheduling mechanisms** and **queue management** for flows within the routers

We examine each of the highlighted components.

The Internet Integrated Services architecture is described in [RFC1633].

INTSERV

- http://www.ietf.org/html.charters/intserv-charter.html
- Requirements for Integrated Services based on IP
- QoS service-levels:
  - current service: **best-effort**
  - **controlled-load service** (RFC2211)
  - **guaranteed service** (RFC2212)
  - other services possible (RFC2215, RFC2216)
- Signalling protocol:
  - RSVP (RFC2205, RFC2210)

DigiComm II-7

It is possible to identify four specific technical issues that need to be addressed in the provision of Integrated Services for IP-based networks:

• **service-level:** the nature of the commitment made, e.g. the INTSERV WG has defined **guaranteed** and **controlled-load** service-levels and a set of control parameters to describe traffic patterns

• **service interface:** a set of parameters passed between the application and the network in order to invoke a particular QoS service-level, i.e. some sort of signalling protocol plus a set of parameter definitions

• **admission control:** for establishing whether or not a service commitment can be honoured before allowing the flow to proceed

• **scheduling mechanisms within the network:** the network must be able to handle packets in accordance with the QoS service requested

The INTSERV WG addresses only the first two of these issues. However, the INTSERV work does specify the requirements for any mechanisms used to address the last two issues, with some implementation hints. With the present IP service enumerated as **best-effort**, currently, two service-level specifications are defined:

• **controlled-load** service [RFC2211]: the behaviour for a network element required to offer a service that approximates the QoS received from an unloaded, best-effort network

• **guaranteed** service [RFC2212]: the behaviour for a network element required to deliver guaranteed throughput and delay for a flow

The INTSERV signalling protocol is called RSVP (Resource Reservation Set-up Protocol, [RFC2205] [RFC2210]).

INTSERV service templates

- Describe service semantics
- Specifies how packets with a given service should be treated by network elements along the path
- General set of parameters
  - <service_name>.<parameter_name>
  - both in the range [1, 254]
- TSpec: allowed traffic pattern
- RSpec: service request specification

INTSERV have produced a set of specifications for specific QoS service-levels based on a general network service specification template [RFC2216] and some general QoS parameters [RFC2215]. The template allows the definition of how network elements should treat traffic flows, i.e. the QoS granularity here is that of a single logical **flow** (or **session** in RSVP parlance).

The service template specifies describes the semantics of the services and specifies how packets should be treated as they pass through network elements that would like to implement the service, i.e. packet handling rules.

The genral parameters are identifed using two bytes, one identifying the service name (e.g. controlled-load) and one identifying the parameter.

The use of a service requires a **TSpec** (Traffic Specification) to specify the allowed traffic characteristics for a session. A **RSpec** (Resourse Specification) may also be used during reservation establishment for service specific parameters, but its use is service specific. The service definition includes information on how admission control is applied for the service and how the service is policed within the network (how non-conformant packets should be handled).

The controlled-load service requires a TSpec but no RSpec. For the guaranteed service,, as well as a TSpec, a RSpec should be specified (which will not be discussed here).

Note that this architecture requires that all the network elements along the path, as well as the applications, and applications have semantic knowledge about the service-levels for the application flows, as specified in the service templates.

Some INTSERV definitions

- Token bucket (rate, bucket-size):
  - token bucket filter: total data sent ≤ (rt + b)
- Admission control:
  - check before allowing a new reservation
- Policing:
  - check TSpec is adhered to
  - packet handling may change if TSpec violated (e.g. degrade service-level, drop, mark, etc.)
- Characterisation parameters: local and composed

DigiComm II-9

A key element of the flow description is the TSpec that describes the (expected) traffic characteristics of he flow/session. The traffic characteristic is defined in terms of a **token-bucket** filter which, in general has the following elements:

• **r**: the data rate (bytes/s)

• **b**: the bucket size (bytes)

This specifies that the flow shall have sent no more than (rt + b) bytes of data at any time t.

This information (along with other, service specific information) may be used by the network for:

• **admission control**: to check if the requested traffic profile and service can be currently be honoured along the intended network path

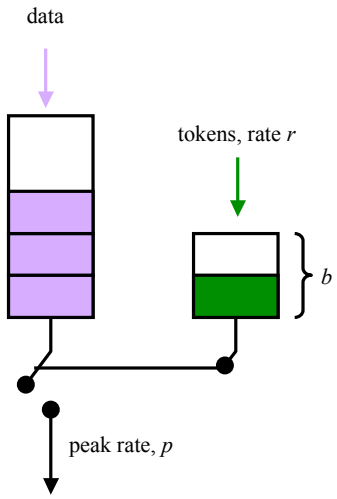
• **policing**: to ensure that the application/user does not exceed the requested traffic specification and to take action (mark/drop packets, degrade the service-level for some packets) as appropriate

The INTSERV specification allows network elements to have **local** value for INTSERV parameters. When a path for a flow/session is selected the **composed** values for a parameter are the combination of the local values for the network elemnts along the path. For example, network elements A, B and C form a path for a flow or session. They have path latency values of 10ms, 12ms and 8ms respectively. Th composed value for the (minimum) path latency will be (10+12+8)ms = 30ms. Different parameters have different composition rules which should be defined when the parameter for a service is defined.

Token bucket (recap)

Token bucket

- Three parameters:
  - *b*: bucket size [B]
  - *r*: bucket rate [B/s or b/s]
  - *p*: peak rate [B/s or b/s]
- Bucket fills with tokens at rate *r*, starts full
- Tokens allow transmission
- Burst allowed at rate *p:*
  - data sent < $rt + b$
- (Also *m* and *M*)



DigiComm II-10

The **token bucket** has a bucket size, *b*, and a bucket rate, *r*, and allows traffic bursts to be transmitted at peak rate, *p*, under certain conditions. The bucket does not "fill with data" as it does in the leaky bucket, but it fills with tokens, that that allow data to be transmitted. Data can only be transmitted when there are enough tokens to allow transmission to take place. Transmission can then take place at a peak rate of *p*. Nominally, data is transmitted at a rate *r*, the same rate at which the bucket is filled with tokens. However, it can be seen that bursts of traffic, up to the bucket size, can be transmitted at the peak rate, *p*. In fact, we may also need to specify *m*, the minimum packet size, and *M*, the maximum packet size.

General INTSERV parameters

- NON_IS_HOP (flag): no QoS support
- NUMBER_OF_IS_HOPS: QoS-aware hop count
- AVAILABLE_PATH_BANDWIDTH
- MINIMUM_PATH_LATENCY
- PATH_MTU
- TOKEN_BUCKET_TSPEC:
  - r (rate), b (bucket size), p (peak rate)
    m (minimum policed unit), M (maximum packet size)

DigiComm II-11

The service template specifies and describes the semantics of the services and specifies how packets should be treated as they pass through network elements that would like to implement the service, i.e. packet handling rules. There is a general set of parameters specified and their values can be defined for each service level, as required. The general parameters include:

• a flag to indicate that a network element is not INTSERV-aware

• hop-count of INTSERV-aware network elements

• available path bandwidth

• minimum path latency

• path MTU

• traffic characteristic in terms of a token bucket specification (data rate, bucket size, peak rate), minimum packet size to be policed and maximum packet size allowed

The last of these parameters is used in the **TSpec**. A **RSpec** (Resource Specification) may also be used during reservation establishment for service specific parameters.

Other, service-specific parameters may be defined. Some of the paramters (for example AVAILABLE_PATH_BANDWIDTH and MINMUM_PATH_LATENCY) are carried in a special **AdSpec** data structure (see later) and are filled in by the routers along the network path to form **composed values**, which represent the values of those parameters for the path as a whole.

Controlled-load service

- Best-effort **under unloaded conditions**:
  - probabilistic guarantee
- Invocation parameters:
  - TSpec: TOKEN_BUCKET_TSPEC
  - RSpec: none
- Admission control:
  - Class-Based Queuing (CBQ), priority and best-effort
- Policing:
  - not defined (e.g. treat as best-effort)

DigiComm II-12

The **controlled-load** service definition [RFC2211] specifies that network elements supporting this service should provide a service that is no worse than a best-effort service that would be seen at that network element under unloaded (lightly-loaded) conditions.

To invoke the service, the TSpec must be specified and a RSpec is not required.

It is suggested that Class-Based Queuing (CBQ) could be used to implement controlled-load service in network elements, e.g. with two classes of service, **priority** for the controlled-load packets and a separate class for normal best-effort packets.

Policing mechanisms are not specified/defined, but it is suggested that non-conformant packets should be degraded to best-effort.

Guaranteed service [1]

- **Assured data rate with bounded-delay**
  - deterministic guarantee
  - no guarantees on jitter
- Invocation parameters:
  - TSpec: TOKEN_BUCKET_TSPEC
  - RSpec: R (rate), S (delay slack term, μs)
- Admission control:
  - Weighted Fair Queuing (WFQ)
- Policing:
  - drop, degrade to best-effort, reshape (delay)

The **guaranteed** service definition [RFC2212] specifies the network elements should treat packets so that there is an assured data rate and all packets have a bounded overall delay. However, the service makes no commitment on jitter (packet inter-arrival delay).

The invocation of the service requires a TSpec and a RSpec. The RSpec contains two parameters, R a required service rate, and S a slack-term for the delay bound. R must be greater than or equal to r (the rate defined in the TSpec token-bucket). S must be non-negative. Defining a bigger value for R helps to decrease the overall path delay. Defining a bigger value for S makes it more likely that the reservation request will succeed, but may result in a larger end-to-end delay. R is used as a suggestion and larger values of S may require routers to use a value lower than R for the reservation. The exact use of R and S are given in [RFC2212].

The suggested admission control mechanism for this service is Weighted Fair Queuing (WFQ) where the weight assignments will be a function of the required rate, R.

The suggested policing function takes one of two forms. A simple policing function (the suggested default) is for non-conformant packets to be dropped or degraded to best-effort. A more complex policing function take the form of reshaping the flow/session by delaying packets so that they conform to the requested parameters.

Guaranteed Service [2]

- End-to-end delay bound:
  - maximum delay
  - based on fluid flow model
  - fluid flow model needs error terms for IP packets
- Error terms:
  - each router holds C and D
- C [B]: packet serialisation
- D [μs]: transmission *through* node
- Composed values:
  - C_{SUM} and D_{SUM}

$$delay = \frac{(b-M)(p-R)}{R(p-r)} + \frac{(M+C_{SUM})}{R} + D_{SUM} \qquad p > R \geq r$$

$$delay = \frac{(M+C_{SUM})}{R} + D_{SUM} \qquad R \geq p \geq r$$

Wit the Guaranteed Service, it is possible to evaluate the exact end-to-end delay bound from the TSpec and RSpec parameters. The equitions above show how to evaluate the delay, using the TSpec and RSpec parameters discussed earlier. There are also two as well as two new values, C_{SUM} and D_{SUM}. The equations above are based on a fluid flow model, and C_{SUM} and D_{SUM} provide the error terms required to correct for the effect of IP packets being of a size that deviates from the fluid flow model. C_{SUM} and D_{SUM} are discovered using RSVP, and constructed from individual values of C and D held at routers. Full details are given in [RFC2212].

RSVP

INTSERV: RSVP [1]

- Provides signalling:
  - user-to-network
  - network-to-network
- Traffic information – *FlowSpec*:
  - *TSpec*
  - sent through network
  - *AdSpec* (optional)
- Receiver confirms reservation:
  - uni-directional reservation

DigiComm II-16

RSVP is a signalling protocol that provides the service invocation interface for applications. The messages are sent between applications, but are acted upon and modified by the network elements en-route, so RSVP provides both user-to-network and network-to-network signalling. Special RSVP message carry *TSpec* and *RSpec* messages that are seen by (INTSERV aware) network elements along the network path as well as by the flow recipients.

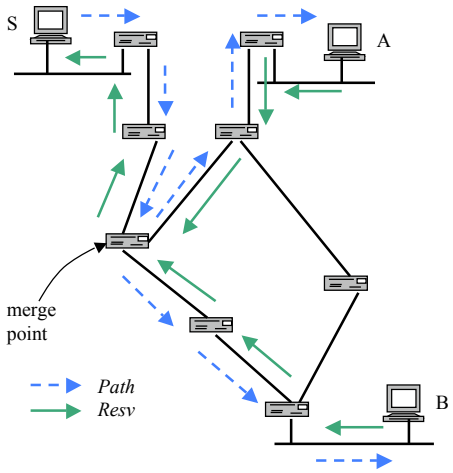
The reservation request consists of a *FlowSpec* identifying the traffic characteristics and service-level required. One part of the *FlowSpec* is a *TSpec*, a description of the traffic characteristic required for the reservation. So it is possible for the same traffic characteristic to be used with different service levels. This difference in QoS service-level could, for example, act as a way for offering cost differentials on the use of a particular application or service.

RSVP can be used to set-up resource reservations for multicast as well as unicast flows. The reservations are unidirectional and in fact it is the receiving application that actually confirms the reservation, i.e. this is a receiver-oriented reservation protocol. The receiver may also be made aware of composed parameter values along the route if an (optional) *AdSpec* is present within the *FlowSpec* transmitted from the sender.

Note that RSVP is a **general** QoS signalling protocol specified in [RFC2205]. For use in a particular QoS architecture additional specification is required. In the case of INTSERV, the additional specification is provided in [RFC2210].

INTSERV: RSVP [2]

- Two-pass, with soft-state:
  - sender: *Path* message
    - NEs hold **soft-state** until *Resv, PathTear* or time-out
  - receiver(s): *Resv* message - TSpec (+RSpec)
  - sender: *PathTear*
  - receiver(s): *ResvTear*
  - soft-state refreshed using *Path* and *Resv*
- Composed QoS params:
  - *AdSpec* for path



DigiComm II-17

To make a resource reservation, an appropriate *FlowSpec* is used along with session IP destination address, the protocol number in the IP packet and – optionally – the destination port number in the service invocation. The reservation procedure is as follows. The sender transmits a *Path* message advertising the session QoS requirements towards the destination IP address. All RSVP routers forwarding the *Path* message hold **soft-state** – information about the resource reservation required – until one of the following happens: a *PathTear* is sent from the sender cancelling the reservation, a *Resv* message is transmitted from a receiver effectively confirming the reservation, or the soft-state times-out. A *Resv* message from a receiver is sent back along the same route as the *Path* message, establishing the reservation and then the application starts sending data packets. *Path* and *Resv* messages are sent by the sender and receiver, respectively, during the lifetime of the session to refresh the soft-state and maintain the reservation. A *PathTear* or *ResvTear* message explicitly tears down the reservation and allows resources to be freed. It is possible for the reservation to be changed dynamically during the lifetime of the session. RSVP can be used for unicast or multicast sessions. (It is assumed that routes are symmetrical and relatively stable, but this is not always true in the wide area.)

As part of the *Path* message, an *AdSpec* data structure may also be sent in a **one pass with advertising (OPWA)** that allow network elements along the path to indicate to the receiver the **composed** (combined) QoS parameter values along the path based on **local** QoS capabilities at each network element. The local and composed capabilities are reported as QoS parameters for each service definition.

Where multicast communication is involved for the same flow, it is possible for a router to effectively merge two reservations instead of making two separate reservations.

Reservation types and merging

- *FilterSpec*: style of reservation
- Fixed-filter (FF):
  - *FilterSpec* required
  - distinct sender reservation
  - explicit sender selection
- Wildcard-filter (WF):
  - *FilterSpec* not required
  - shared sender reservation
  - wildcard sender selection
- Shared-explicit (SE):
  - *FilterSpec* required
  - shared sender reservation
  - explicit sender selection
- Merging reservation info:
  - merging allows aggregation of reservation information
  - merging not possible across styles
  - merging possible for reservations of the same style – use maximum

There are three reservation styles that are permitted with RSVP/INTSERV.

• **fixed-filter (FF)** style: this style sets up a distinct reservation per sender that requires and specifies explicitly the set of sender who can make use of this reservation specification.

• **wildcard-filter (WF)** style: allows a shared reservation for senders, but the senders are not explicitly specified.

• **shared explicit (SE)** style: allows the reservation to be shared amongst an explicitly specified list of senders.

Information about the list of senders for FF and SE is carried in a *FilterSpec* data structure that forms part of the *Resv* message provided by the sender.

Its is possible for the RSVP routers to merge reservations of the same style. This is effectively to allow router to pass upstream a single reservation that is a maximum of the incoming reservations. This is specifically for multicast, where many flows for the same group are merged.

FF would typically be used for unicast communication only

WF would be used for an open conference, where the number of senders and who they will be is not known *a priori*. It would be expected that only one person would be speaking at a time, and perhaps the reservation would be enough for two speakers just in case two people did start to speak at once.

SE would be for a similar situation to WF but the conference would be closed, with the senders known before hand and listed in the FilterSpec.

Reservations about reservations

- Two-pass – one reservation may "block" another:
  - *PathErr* and *ResvErr*
- Need to hold a lot of soft-state for each receiver
- Extra traffic due to soft-state refreshes
- Heterogeneity limitations:
  - same service-level
- Router failure:
  - QoS degrades to best-effort, need to re-negotiate QoS
- Applications and routers need to be RSVP aware:
  - legacy applications
- Charging

DigiComm II-19

We summarise the main problems with RSVP below:

1. Intuitively, we can see that in a network with limited resources, which are heavily utilised (e.g. the Internet), it is likely larger reservations are probably less likely to succeed that smaller reservations. During reservation establishment if the first pass of each of two separate reservation requests are sent through the same network element, where one request is a "super-set" of the other, the lesser one may be rejected (depending on the resources available), even if the greater one eventually fails to complete (of course it is possible to re-try).

2. If the first pass does succeed, the router must then hold a considerable amount of state for each receiver that wants to join the flow (e.g. in a multicast conference)

3. The routers must communicate with receivers to refresh soft-state, generating extra traffic, otherwise the reservation will time out

4. Complete heterogeneity is not supported, i.e. in a conference everyone must share the same service-level (e.g. guaranteed or controlled-load), though heterogeneity within the service-level is supported

5. If there are router failures along the path of the reservation, this results in IP route changes, so the RSVP reservation fails and the communication carries on at best-effort service, with the other routers still holding the original reservation until an explicit tear-down or the reservation times out or the reservation can be re-established along the new path

6. The applications must be made RSVP aware, which is a non-trivial goal to realise for the many current and legacy applications that already exist, including multimedia applications with QoS sensitive flows

Resource reservation could be expensive on router resources and adaptation capability is still required within the application to cope with reservation failures or lack of end-to-end resource reservation capability. Indeed, RSVP is now recommended for use only in restricted network environments [RFC2208].

Additionally, there is as yet no agreement as to how to charge for end-to-end QoS guarantees that span the networks of multiple administrations, e.g. across multiple ISPs.

DIFFSERV

DIFFSERV

- http://www.ietf.org/html.charters/diffserv-charter.html
- Differentiated services:
  - tiered service-levels
  - service model (RFC2475)
  - simple packet markings (RFC2474)
- Packets marked by network, not by application:
  - will support legacy applications
- Simpler to implement than INTSERV:
  - can be introduced onto current networks

DigiComm II-21

Concerns about resource reservation have directed the Internet community to consider alternatives; specifically **differentiated services**. In fact the IETF DIFFSERV WG was spawned directly from the INTSERV WG.

This is a relatively new IETF WG and most of the work within this group is currently at the stage of discussion and the formulation of a framework and architecture for the DIFFSERV work.

The IETF charter for the workgroup is:

http://www.ietf.org/html.charters/diffserv-charter.html

Two RFC documents have been produced. RFC2474 describes special values to be used for the IPv4 ToS field or IPv6 traffic-class field when DIFFSERV is in use. RFC2475 describes the DIFFSERV architecture.

DIFFSERV hopes to offer a relatively simple, coarse-grained QoS mechanism that can be deployed in networks without needing to change the operation of the end-system applications. The QoS mechanism is based around marking packets with a small-fixed bit-pattern, which maps to certain handling and forwarding criteria at each hop. The WG seeks to identify a common set of such per-hop handling behaviours as well as packet markings to identify these behaviours.

This is a much coarser granularity of service, but reflects a well understood service model used in other commercial areas. The DIFFSERV model is different to INTSERV. A key distinction of the DIFFSERV model is that it is geared to a business model of operation, based on administrative bounds, with services allocated to users or user groups.

The DIFFSERV mechanisms should be simpler to implement than INTSERV mechanisms and will allow some QoS control for legacy applications that are not QoS aware.

Service Level Agreements

- Not (necessarily) per-flow:
  - aggregate treatment of packets from a "source"
- Service classes:
  - Premium (low delay) - EF (RFC2598)
  - Assured (high data rate, low loss) - AF (RFC2597)
- **Service level agreement (SLA)**:
  - **service level specification (SLS)**
  - policy between user and provider - policing at ingress
  - service provided by network (end-system unaware)

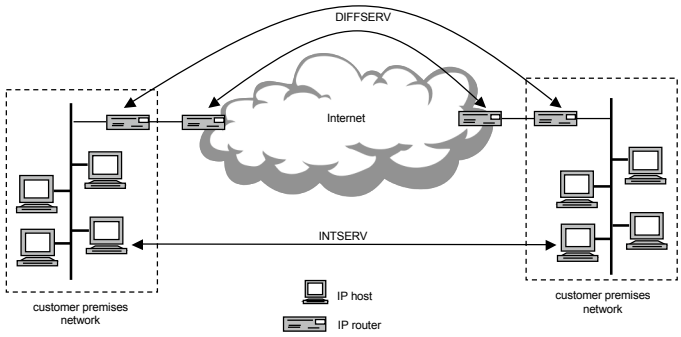
DigiComm II-22

Whereas RSVP can act on a per-flow basis, the DIFFSERV classes may be used by many flows. Any packets within the same class must share resources with all other packets in that class, e.g. a particular organisation could request a Premium (low delay service provided using **Expedited Forwarding**) quality with an Assured (low loss, using **Assured Forwarding** with different drop precedence assignments) service-level for all their packets at a given data rate from their provider.

The exact nature of the packet handling will be based on a policy and **Service Level Specification (SLS)** that forms part of a **Service Level Agreement (SLA)** between user and provider. The policy could be applied to all the traffic from a single user (or user group), and could be set up when subscription to the service is requested, or on a configurable profile basis. The policy implemented by the SLA may include issues other than QoS that must be met, e.g. security, time-of-day constraints, etc.

The DIFFSERV mechanisms would typically be implemented within the network itself, without requiring runtime interaction from the end-system or the user, so are particularly attractive as a means of setting up tiered services, each with a different price to the customer.

Scope of DIFFSERV



DigiComm II-23

The DIFFSERV-capable routers could be at the edge of the customer network or part of the provider's network. If the DIFFSERV-marking is performed within the customer network, then policing is required at the ingress router at the provider network in order to ensure that customer does not try to use more resources than allowed by the SLA.

The INTSERV mechanism seeks to introduce well-defined, end-to-end, per-flow QoS guarantees by use of a sophisticated signalling procedure. The DIFFSERV work seeks to provide a "virtual pipe" with given properties in which the user may require adaptation capability or further traffic control if there are multiple flows competing for the same "virtual pipe" capacity.

Additionally, the DIFFSERV architecture means that different instances of the same application throughout the Internet could receive different QoS, as different users may have different SLAs with their subscriber. So the application needs to be dynamically adaptable.

DIFFSERV classification [1]

- Packet marking:
  - IPv4 ToS byte or IPv6 traffic-class byte
  - **DS byte**
- Traffic classifiers:
  - **multi-field (MF)**: DS byte + other header fields
  - **behaviour aggregate (BA)**: DS field only
  - **DS codepoint**: values for the DS byte
- Aggregate per-hop behaviour (PHB):
  - aggregate treatment within network

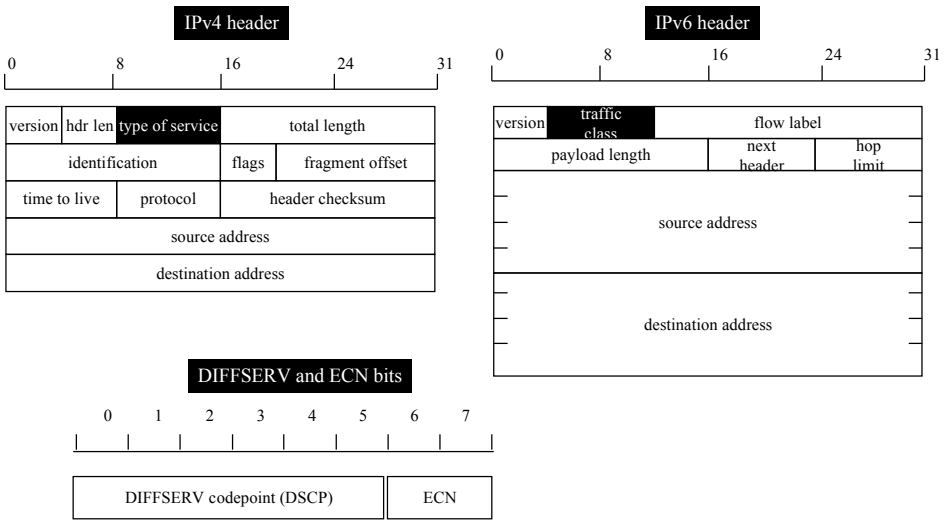
DigiComm II-24

The DIFFSERV work is aimed at providing a way of setting up QoS using policy statements that form part of a service level agreement between service user and service provider. The policy may use several packet header fields to classify the packet, but the classification marking can also be a simple identifier – currently a single byte, the **DS (differentiated services) byte** – within the packet header. The **DS (differentiated services) byte** will be used in place of the ToS (Type of Service) field in IPv4 packets or the traffic-class field in IPv6 packets. The DS byte will have the same syntax and semantics in both IPv4 and IPv6. There are likely to be some global values – **DS codepoints** – agreed for the DS field within the IETF but the intention is that the exact policy governing the interpretation of the DS codepoints and the handling of the packets is subject to some locally agreed SLA. SLAs could exist between customer and Internet Service Provider (ISP) as well as between ISPs. The DS codepoints are used to identify packets that should have the same aggregate **per-hop behaviour (PHB)** with respect to how they are treated by individual network elements within the network. The PHB definitions and the DS codepoints used may differ between ISPs, so there will be need for translation mechanisms between ISPs.

A traffic classifier selects packets based either on the on DS codepoint or on some (policy-based) combination header fields from the packet header and directs them to an appropriate traffic conditioner. When the DS codepoint is used to classify traffic, the classifier is called a **Behaviour Aggregate (BA)** classifier. When other packet header fields are used we have a **Multi-Field (MF)** classifier. And MF classifier may use information such as the port numbers, IP addresses protocol types, as well as the DS byte to make classification decisions.

Although there will be scope for changes to the SLA by agreement between customer and provider, the kind of dynamic, flexible, host-to-host resource reservation that is possible with the INTSERV model using RSVP is not envisaged for DIFFSERV.

DIFFSERV classification [2]



DigiComm II-25

This is the usage proposed by RFC2474 for the ToS (IPv4) and traffic class (IPv6) byte. For bits 6 and 7, marked “currently unused”, RFC2481 proposes they be used to provide explicit congestion notification (ECN) at the IP-level. This would allow DIFFSERV and ECN to be used together, the former to provide coarse-grained (class-based) QoS and the latter to provide congestion control signalling, by simply re-using an existing field in the IP header.

DIFFSERV PHBs

- Specify rate/delay in SLS
- **Expedited Forwarding (EF)** (RFC2598):
  - virtual leased line (VLL) service
  - data rate specified in SLS
  - low delay, low jitter, low loss
- **Assured Forwarding (AF)** (RFC2597):
  - 4 classes (1-4)
  - 3 levels of drop precedence per class (1-3)
  - AF11 - “best”, AF43 - “worst”

DigiComm II-26

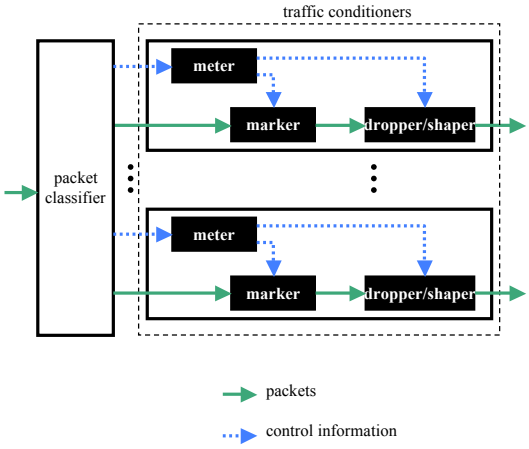
Recently (June 1999), the DIFFSERV WG have defined two PHBs, both of which are proposed standards. Both require that the SLS contain information such as delay, data rates (e.g. token bucket filters), and the scope over which the SLS applies (e.g. between ingress and all end-points, between ingress and specific end-points, etc.), as well as actions to take if the traffic is found to be violating the SLS.

The **Expedited Forwarding (EF)** PHB is used to provide a low loss, low delay, low jitter end-to-end service across DS domains. The service if provides is likened to that of a virtual leased line (VLL). Suggested implementation mechanisms include weighted round robin scheduling and class based queuing (CBQ). If simple priority queuing is used (the EF queue is always serviced before any other traffic) then the implementation must ensure that other traffic is not locked out (e.g. by using rate limiting via a token bucket filter). Violating traffic can be dropped. A single DS codepoint is defined for EF.

The **Assured Forwarding (AF)** PHB will allow a DS domain to provide different levels of assurance for forwarding of IP packets. Currently, 4 AF classes are defined with 3 drop precedence levels in each. An AF class mark is indicated by the lexeme *AF**cd*** where ***c*** is the AF class and ***d*** is the drop precedence within that class. An example usage is that each class represents a higher level of service (e.g. 1= platinum, 2=gold, 3=silver, 4=bronze), with low, medium and high (1, 2, 3 respectively) drop precedence levels in each class. So, AF11 would be the “best” AF mark and AF43 the “worst”. Implementation might be using weighted queuing/scheduling with violating traffic being dropped or re-marked to lower classes, higher drop precedence or best effort.

DIFFSERV traffic conditioning

- Traffic conditioners:
  - meter
  - marker
  - shaper/dropper
- Metering of traffic:
  - in-profile
  - out-of profile
- Re-marking:
  - new DS codepoint
- Shape/drop packets



DigiComm II-27

A **DS domain** contains **DS boundary nodes** at its edge and **DS interior nodes** within the domain. DS boundary nodes act as **traffic conditioners**. Traffic conditioners implement the **Traffic Conditioning Agreement (TCA)** part of a SLA. A schematic diagram showing how streams are treated is shown in .

Part of the SLA is the definition of a **traffic profile** for a packet stream. This may, for example, be specified as a token bucket, limiting the way that packets are transmitted into the DS domain. When packets in a stream from a user exceed the negotiated traffic profile, they are said to be **out-of-profile**, else packets are **in-profile**.

After passing through a classifier, information about the packet is passed to a meter that provides control information to other parts of the conditioner. This information includes whether or not the packet is in-profile or out-of-profile. Within the conditioner, the packets follow a path through a marking function and a policing function:

• **marker:** may change the DS codepoint of the packet – **re-mark** the packet

• **shaper:** delays out-of-profile packets in order to enforce the traffic profile for a stream

• **dropper:** drops out-of-profile packets in order to enforce the traffic profile for a stream

Note that a dropper can be implemented by using a shaper with the buffering reduced to zero packets (or a few packets).

DS interior nodes may perform limited BA traffic conditioning, but the intention is that the main traffic conditioning function is performed at the edges of DS domain (as the DS boundary nodes), close to where the packets enter the domain.

DIFFSERV service invocation

- At subscription:
  - per user/user-group/site/customer
  - multi-field, policy-based
- Within organisation:
  - per application/user/user-group
  - use ad hoc tools or network management system
  - behaviour aggregate or multi-field possible
- Dynamically using RSVP: IETF work in progress

DigiComm II-28

It is intended that the DIFFSERV work will offer a subscription-time mechanism for defining coarse-grained QoS requirements for an organisation. The exact nature of the service level agreement will be left as a matter of negotiation between user and provider. However, DIFFSERV offers an architecture and definitions that will allow an SLA to be defined. The policy for controlling traffic could be based on applications, individual users or user-groups.

It may even be possible to have control of traffic within an organisation, providing the network elements can be persuaded to be DIFFSERV aware. The network elements could be configured, for example, to control the amount of traffic form a particular application appearing on certain segments of the network, e.g. off-site WWW traffic.

It may even be possible to control or invoke SLAs more dynamically using RSVP (with suitable additional specifications) but this is currently work in progress.

Problems with DIFFSERV

- No standard for SLAs:
  - same DS codepoints could be used for different services by different providers
  - different providers using the same PHBs may have different behaviour
  - need end-to-end/edge-to-edge semantics
- Lack of symmetry:
  - protocols such as TCP (ideally) require symmetric QoS
- Multicast:
  - support for multi-party, symmetric communication?

DigiComm II-29

DIFFSERV is not without its own problems, however.

Firstly, there is a problem with service definitions. Only DS-codepoints have been defined, and not end-to-end semantics (though two standard track PHB documents do exit). This means that it will be possible for service providers to implement different services using the same DS codepoints. So, provider must co-operate and ensure that mappings between DS codepoints at network boundaries results in semantically correct service translation as packets go from one network to another.

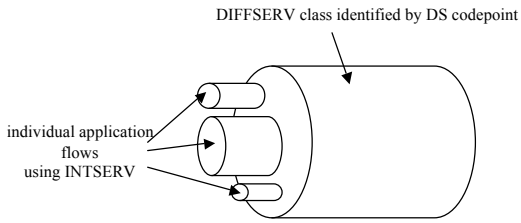
For the current standard-track PHB documents, it is possible that different provider may implement different behaviour across their networks for the same DS codepoints, though this is likely to be more so with AF than with EF. Edge-to-edge semantics are required, so that network boundaries can still be honoured but handling of packets is consistent.

Secondly, note that the SLA/SLS is between a user and their service provider. If that user accesses a server which is connected using a link that has a different, perhaps "worse", SLA/SLS with its provider then our user would not see the service they expect when paying for the "better" service. This is because the return traffic from the server is treated differently – worse – than the initial request to the server. So, for whizzy web-browsing you need to ensure that the server site has a "good" SLA/SLS as well as getting a "good" SLA/SLS yourself. This lack of symmetry in DIFFSERV connectivity would affect protocols such as TCP which rely on a a two way exchange for reliability.

There is also the issue of support for multicast. DIFFSERV is not as dynamic invocation of services as is INTSERV/RSVP. DIFFSERV, at least currently, I based on the notion of a subscription. Work is in progress to allow dynamic establishment of SLAs/SLS using RSVP. However, many end-to-edn signalling issues remain unresolved.

INTSERV and DIFFSERV [1]

- Complimentary:
  - DIFFSERV: aggregate, per customer/user/user-group/application
  - INTSERV: per flow
- For example:
  - INTSERV reservations within DIFFSERV flows (work in progress)



DigiComm II-30

The big gain with DIFFSERV is that the end-to-end signalling and the maintenance of per-flow soft-state within the routers that is required with RSVP is no longer required. This makes DIFFSERV easier to deploy and more scaleable than using RSVP and INTSERV services. However, this does not mean that INTSERV and DIFFSERV services are mutually exclusive. Indeed, it is likely that DIFFSERV SLAs will be set-up between customer and provider for general use, and then RSVP-based per-flow reservations may be used for certain applications as required, e.g. for instance an important video conference within an organisation.

INTSERV and DIFFSERV [2]

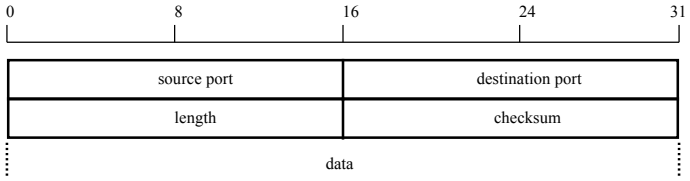
	INTSERV	DIFFSERV
signalling	from application	network management, application
granularity	flow	flow, source, site (aggregate flows)
mechanism	destination address, protocol and port number	packet class (other mechanisms possible)
scope	end-to-end	between networks, end-to-end

RTP

UDP

- Connectionless, unreliable, unordered, datagram service
- No error control
- No flow control
- No congestion control
- Port numbers

- Must be used for real-time data:
  - TCP automatic congestion control and flow control behaviour is unsuitable



DigiComm II-33

UDP provides an unreliable, connectionless datagram service. It does not guarantee delivery or ordering, and individual packets may be duplicated within the network. Exactly how "unreliable" the service is depends very much on the network environment. In a lightly loaded LAN, it is unlikely that you will observe much packet loss. Across a wide area backbone, however, there may be significant packet loss, especially over paths involving large numbers of routers or heavily loaded routes.

UDP is very simple to implement, and this is reflected in the packet header for UDP. The port number work in a similar way to those for TCP, identifying a local UDP end-point.

RTP

- RFC1889: general message format
  - specific formats for media types in other RFCs
- Carried in UDP packets:
  - application must implement reliability (if required)
  - supports multicast and point-to-point
- RTCP - Real Time Control Protocol:
  - application-level information (simple signalling)
- **RTP and RTCP provide no QoS guarantees:**
  - QoS mechanisms are separate

DigiComm II-34

The **Real time Transport Protocol (RTP)** is an Internet Proposed Standard and is widely used for multimedia applications (including voice and video) within the Internet community. Its use as the underlying transport mechanism for packetised voice and video is specified in H.323.

RTP carries "time-slices" of audio and video flows in UDP packets, with synchronisation information and application-specific identifiers, QoS parameter information and user information. RTP itself is a general mechanism and the are specific RTP usage **profiles** available for different media types, each described in their own RFC document, e.g.

• RFC2032 for H.261

• RFC2038 for MPEG1 and MPEG2

• RFC2190 for H.263

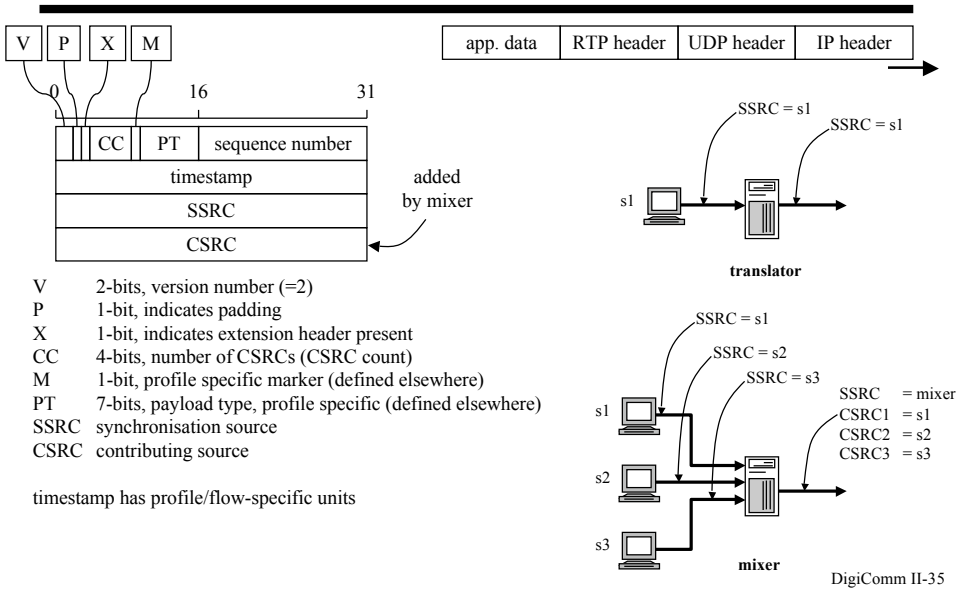
• RFC2198 for redundant (fault tolerant) audio

and many others. RTP is designed to support multicast and unicast communication.

RTP has an associated with it a simple application-level signalling protocol, the **Real Time Control Protocol (RTCP)** that allows application using RTP to pass resource usage information, flow QoS parameters and other information between senders and receivers.

RTP and RTCP themselves do not provide QoS control or resources reservation - they are protocols that enable the transport of real-time media flows.

RTP header information



RFC1889, the RTP specification, defines some general header information that is used by all RTP applications.

All RTP packets carry a sequence number to allow detection of loss and misordering at the receiver.

There is an application-specific timestamp indicates where in the flow this packet should be with respect to the rest of the flow. This allows synchronisation of the flow playback at the receiver, and also allows packets to be disregarded if they are delayed beyond the point that they have far exceeded their playout time.

RTP uses unique identifiers - SSRC (synchronisation source) and CSRC (contributing source) to identify originators of flows within and RTP session. Any ⟨IP address, SSRC⟩ pair must be unique so that multiple flows from the same host can be distinguished. The SSRC is randomly generated.

An end-station will generate an SSRC to be carried in the RTP header. The packets in a flow may pass through a **translator** or a **mixer**. When passing through a translator, the flow may be altered, e.g. transcoded, but this is transparent to the receiver. When a flow passes through a mixer, the mixer may decide to merge and/or translate flows. When flows are merged (mixed) the mixer identifies itself as the SSRC, but also identifies the original sources of the mixed flows by putting their respective SSRC IDs into the CSRC list of the packet header. (A maximum of 15 flows can be mixed.)

Media specific header extensions are defined in the relevant RFC documents.

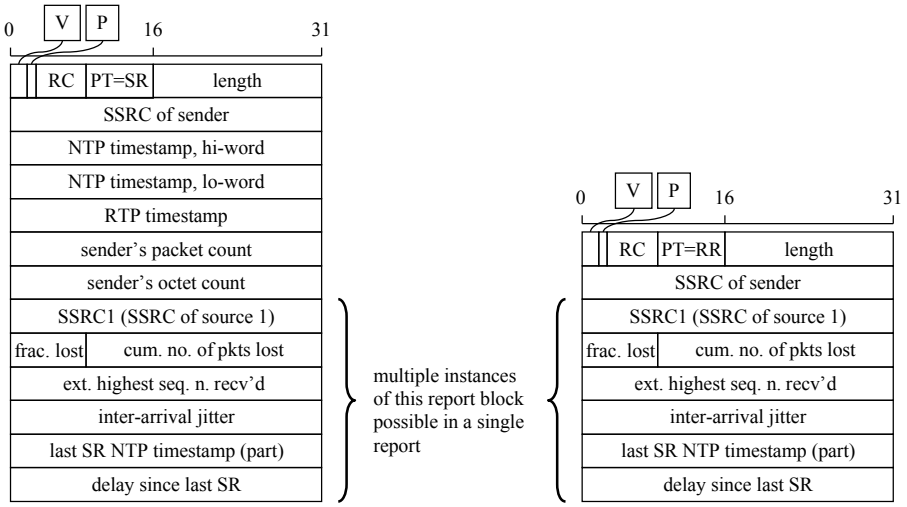
RTCP - Real time Control Protocol

- Provides feedback to senders/receivers
- QoS info for flow:
  - packet info: loss, delay, jitter
  - end-system info: user info
  - application-specific or flow-specific info
- RTCP message types:
  - RR and SR: Receiver Report and Sender Report
  - SDES: Source DEScription
  - BYE: leave a RTP session
  - APP: application-specific

RTCP provides simple information about the flow. Reports are sent by senders and receivers. The RTCP messages defined in RFC1889 carry information about the loss, delay and jitter for a flow, as well as some end-system user information. Additionally, application-specific information is defined for particular media-flows in he relevant RFC documents.

The generation of control message is controlled by an algorithm that seeks to limit the amount of RTCP traffic to around 5% the available network capacity.

SR and RR messages



The Receiver Report and Sender Report are used to convey information about the flow throughout the lifetime of the flow.

The SSRC is used to identify the sender of the RR/SR and then the rest of the message consists of Report Blocks. Each report block identifies the source using an SSRC and the gives the following information for each:

- fraction of lost packets for the flow
- cumulative number of lost packets
- the last received sequence number, and also the number of times the sequence number has cycled (wrapped)
- estimate of the variance of inter-packet arrival time
- part of the last NTP timestamp sent in the SR as received by this SSRC
- the delay since the last SR was received

The SR also has:

- NTP timestamp
- RTP timestamp (flow-specific)
- sender's packet count
- sender's octet count

This information allows the applications to evaluate the QoS being received by particular flows from particular senders. This may allow the application to co-ordinate adjustments to the flow based on the QoS information.

NTP is the Network Time Protocol. The RTP timestamp provides application/flow specific timing information whilst the NTP timestamp provides a measure of global time.

Both the RR and SR can be extended with profile specific information.

SDES

- Source DEScription: all ASCII strings
- Information types from RFC1889:
  - CNAME: canonical identifier (mandatory)
  - NAME: name of user
  - EMAIL: address user
  - PHONE: number for user
  - LOC: location of user, application specific
  - TOOL: name of application/tool
  - NOTE: transient messages from user
  - PRIV: application-specific/experimental use

DigiComm II-38

SDES message are simple ASCII strings that contain information that is typically application-specific. RFC1889 defines 8 types that can be carried in the SDES message, most of which will have application-specific values:

• CNAME: this is the only mandatory type and is used to uniquely identify a participant in a conference. It is normally generated automatically by the application and usually takes the form: *user@host* (or just *host* on single user systems), e.g. saleem@darhu.cs.ucl.ac.uk

• NAME: the real name of the user (or any other identifying string, e.g. nickname, etc.)

• EMAIL: RFC822 e-mail address of the user, e.g. jon.crowcroft@cl.cam.ac.uk (this value could also be used for CNAME)

• PHONE: international format phone number, e.g. "+44 20 7679 3249"

• LOC: physical location of the user (application-specific detail required here)

• TOOL: identifies the name of the application/tool e.g. "blob-talk audio tool v42"

• NOTE: for transient message from the user, e.g. "out to lunch"

• PRIV: to allow application-specific SDES contents and for experimental use

BYE and APP

- BYE - leave RTP session:
  - SSRC (or SSRC and CSRC list if mixer)
  - reason for leaving
- APP - application-specific packets:
  - SSRC (or SSRC and CSRC list if mixer)
  - ASCII string for name of element
  - application-specific data

The BYE message allows end-points to signal that they are leaving a session. The packet can contain a SSRC if sent by a single system or an SSRC and CSRC list if sent by a mixer. Optionally, a string giving the reason for leaving may be included.

If a mixer receives a BYE message, it should forward it unchanged. If the mixer itself shuts down, then it should send a BYE message with itself as the SSRC and CSRC for all its contributing sources.

The APP message is a mechanism that can be used for application specific messages. This mechanism is also intended for use in development and testing of a new media flow or application before making specific RTP/RTCP modifications that may be documented a separate RFC.

Application-level signalling

User-to-network

- Telco network:
  - common channel signalling (CCS)
  - separate data path and signalling path
  - equipment designed to handle data and signalling separate
- IP:
  - RSVP carried in IP packets along data path
  - scaling issues (RFC2208)
  - need aggregated signalling towards the core (use INTSERV with DIFFSERV?)

DigiComm II-41

Telco networks use **common channel signalling (CCS)**, which provides physically separate channels for signalling. Telco equipment is designed to have separate data paths and signalling paths. Signalling also allows the switching of channels to be aggregated. IP has none of these facilities, and these of signalling is relatively new to the IP world. While level-4 protocols such as TCP do have handshaking, and there are application-specific session information exchanges, these are all carried as IP packets along the same path that will eventually carry the data. This means that routers must look for signalling packets as they handle data, a function that slows down the processing of data packets. Also, signalling such as RSVP can not be aggregated in the same way as can telco signalling. Indeed we have already seen that [RFC2208] points out the scaling limitations of RSVP, as used in its current form. Perhaps the solution would be to use RSVP as an edge-system mechanism, and map flows into DIFFSERV pipes, e.g. map Guaranteed service-level request to EF PHB pipes, and Controlled-load service level to AF11 PHB pipes.

User-to-user signalling

- Call/session set-up
- Capabilities exchange
- Directory services
- PBX-like facilities
- Application-level signalling supported by network
- MMUSIC IETF WG:
  - application architecture
  - SDP
  - SIP (now has its own WG)
- H.323:
  - umbrella document for existing standards
  - uses ITU and IETF standards
  - currently more mature than MMUSIC work
  - wide support available (e.g. Microsoft NetMeeting)
  - IMTC: www.imtc.org

DigiComm II-42

There is also a need for application-specific signalling in establishing multimedia sessions. The kind of information that is required is typically configuration and control information to allow a session to take place, e.g. multicast address, time and duration of session, audio and video profile to use, etc. Additional signalling mechanisms can be envisaged that allow capabilities negotiation (allowing terminals to establish negotiate use of various audio and video codecs), and directory services allowing location of users to be determined. Also, there is the desire to build in more traditional PBX-like functions into the software environment, such as call forwarding, call waiting etc. While this is application-level signalling, the transmission of the this signalling information may need to be supported by service providers for Internet-wide use, but of course as with telco PBXs, virtual private networks (VPNs) are possible.

Within the Internet community, the Multi-party Multimedia Session Control (MMUSIC) workgroup of the IETF is defining an architecture for multimedia applications as well as protocols for describing sessions (Session Description Protocol – SDP) and initiating calls or session (Session Initiation Protocol – SIP). SIP supports functions such as call waiting, call forwarding etc. SIP is designed to be very compatible with HTTP and other existing Internet standards.

The ITU world has documented a similar infrastructure in the Recommendation H.323. This umbrella document from the ITU describes how existing ITU and Internet protocols can be used together to offer build multimedia terminal equipment as well as control infrastructure such as Gatekeepers for call control, multi-point control units (MCUs) for conferencing as well as resource control. The H.323. work is more mature than the MMUSIC work, with H.323v1 (1996) and H.323.v2 (1998) now fairly widely accepted and implemented. More information about H.323 and related standards can be found at the WWW site of the Internet Multimedia Technical Consortium (IMTC) which is an industry forum promoting the use of H.323 and related standards.

Summary

- Need QoS mechanisms for IP
- Per flow:
  - INTSERV
  - RSVP
  - does not scale well, hard to provision
- Customer/provider services:
  - DIFFSERV
  - still maturing
- Support for application: RTP and signalling

Routing for Integrated Services

New routing requirements

- Multiparty communication:
  - conferencing (audio, video, whiteboard)
  - remote teaching
  - multi-user games
  - networked entertainment – “live broadcasts”
  - (distributed simulations)
  - (software distribution)
  - (news distribution)
- Support for QoS in routing

As we have already discussed, there are a whole new range of applications that will support **Integrated Services** – one network all services. However, in order for Integrated Services to be possible on an IP-based network we need additional support – things that were not specified in the original IPv4 specification.

One aspect of communication that is increasing rapidly is that of multiparty communication. This is the ability to have a communication session that is not just one-to-one, but perhaps one-to-many or many-to-many. Such application including multimedia conferencing, remote teaching and multi-user games. These may demanding have QoS requirements as well as the requirement for many-to-many communication. (Other multi-party communication applications distributed simulation, software distribution and news distribution whose main requirement may be reliable multiparty communication.)

Let us also consider the current mechanisms for routing and forwarding. These are built around the use of destination addresses for building routing tables, and not other constraints are applied. Traditionally, there is only one route between a source and destination. However, what if we would like to perform routing specifying QoS criteria, allowing alternative route selection based on, for example, the requirement for low-end-to-end delay and loss? Traditionally, the use of such QoS constraints are not used generally in constructing routing information.

Questions

- How can we support multiparty communication?
- How can we provide QoS support in routing?

So we would like to answer two questions in this section:

How can we support many-to-many communication? This is not a simple case of having $O(N^2)$ point-to-point unicast connections for our N end-points. Such a nave solution is not practical – it will not scale.

Also, how can we provide QoS-based decision making for constructing and selecting routes? Again, this is not a simple case of adding extra information about QoS parameters to routing updates as we must consider carefully the implications for the operation of the routing algorithms and protocols, especially the intra-domain and inter-domain interactions.

Many-to-many communication: IP multicast

Group communication using IP

- Many-to-many:
  - many senders and receivers
  - **host group** or **multicast group**
- One transmission, many receivers
- Optimise transmissions:
  - e.g. reduce duplication
- Class D IP address:
  - 224.0.0.0 - 239.255.255.255
  - **not** a single host interface
  - some addresses reserved
- Applications:
  - conferencing
  - software update/distribution
  - news distribution
  - mutli-player games
  - distributed simulations
- Network support:
  - LAN
  - WAN (Internet routers)
  - scoped transmission: IP TTL header field

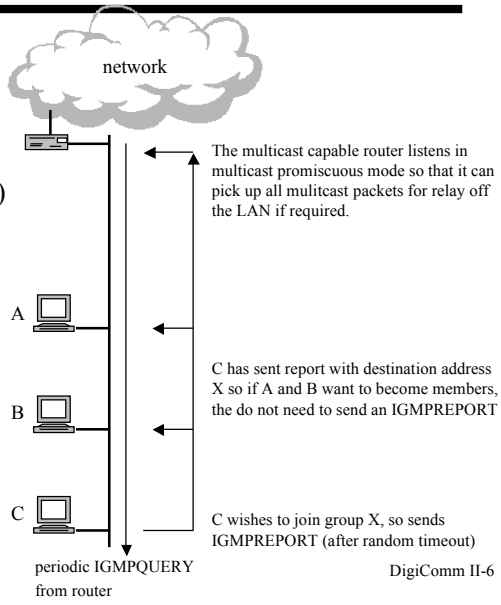
Multicast can be defined, loosely, as the ability to logically connect a group of hosts in a network in order that they perform many-to-many communication. This group of hosts is called a **multicast group** or a **host group**. In a an IP network, multicast is the process whereby *a source host or protocol entity sends a packet to multiple destinations simultaneously using* ***a single 'transit' operation*** which implies that the packet transit only takes place once from sender to all destinations in the group rather than once for each destination. The connectionless nature of packet switched network means that the packet sender is not necessarily in the multicast group. A packet switched network is said to provide a multicast service if it can deliver a packet to a set of destinations (a multicast group), rather than to just a single destination. Basically, a multicast service can offer many benefits to network applications in terms of reducing the transmission overhead on the sender, reducing the overhead on the network and time taken for all destinations to receive all the information when an application must send the same information to more than one destinations. The key to efficient multicast is to optimise the duplication of the transmitted data in some sense. Normally, this means keeping the duplication of the transmitted information to a minimum.

IP multicast uses Class D IP addresses in the range 224.0.0.0 – l 239.255.255.255. These addresses do not identify a single host interface as unicast IP addresses do, but a group of hosts that may be widely, geographically dispersed. This means that special routing procedures are required in the wide-area to enable multicast connectivity. Some of these are reserved, e.g. 224.0.0.1 is the"all systems" address which all hosts must listen to. To contain the scope of IP multicast packets, the TTL field in the IP header is used to limit the maximum number router hops that a multicast packet can traverse before it should be silently discarded.

Multicast has many benefits over unicast communication in certain areas, e.g. conferencing, software distribution/updates and news distribution. To enable multicast communication, support is needed in the end-systems (hosts and LANs) as well as in the wide-area Internet.

IP multicast and IGMP

- Features of IP multicast:
  - group of hosts
  - Class D address
  - leaf nodes (hosts) and intermediate nodes (routers)
  - dynamic membership, leaf-initiated join
  - non-group member can send to group
  - multicast capable routers
  - local delivery mechanism
- IGMP: group membership control



DigiComm II-6

Here we briefly introduce the fundamentals of IP multicast:

• IP multicast allows efficient simultaneous communication between hosts in a logical group called the **host group** or **multicast group**. A host/multicast group which includes a set of zero or more hosts, is identified by a single IP destination address from a specially designated address space.

• The group communication path is modelled as a tree network with the hosts (senders and receivers) within the group located at the **leaf nodes** of the tree, and the intermediate nodes representing distribution/replication points of the communication path.

• The membership of a host group is dynamic; i.e., hosts may join and leave groups at any time (leaf initiated join). This is achieved using the Internet Group Management Protocol (IGMP). There are no restrictions on the physical location or the number of members in a multicast group. A host may be a member of more than one multicast group concurrently.

• A host need not be a member of a group to send packets to the multicast group.

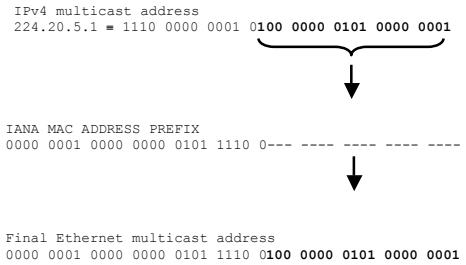
• Inter-network IP multicast is supported by multicast routing mechanisms. This means that inter-network forwarding of IP multicast packets is handled by multicast routing mechanisms residing in "multicast capable routers". The intermediate nodes of the communication path should be multicast capable routers.

• IP multicast relies on the existence of an underlying multicast delivery system to forward data from a sender to all the intended receivers within a sub-network.

IGMP is a very simple protocol with only to messages, IGMPQUERY (sent by a router to see if there are any members of a particular group) and IGMPREPORT (sent by a node to indicate it is leaving or joining a group). Each message refers to a single multicast group, i.e. a single IP multicast address. For Internet-wide connectivity every LAN must have at least one **multicast router** that can listen out for hosts that send group membership reports. If at least one group member exists, then the router should forward multicast packets for that group. To minimise traffic, hosts set random timers and do not send a IGMPREPORT for joining groups until a random timer has expired. IGMP messages are only used in the local area.

Multicast: LAN

- Need to translate to MAC address
- Algorithmic resolution:
  - quick, easy, distributed
- MAC address format:
  - IANA MAC address allocation
  - last 23-bits of Class D
  - not 1-1 mapping
- Host filtering required at IP layer



DigiComm II-7

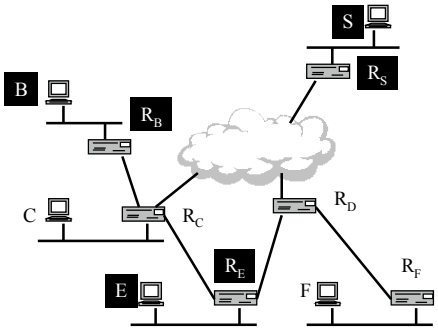
Single LAN multicast is possible without the need for a multicast router. However, LANs do not understand IP addresses they understand MAC addresses. We need address resolution.

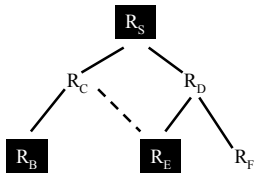
MAC multicast addresses cannot be hardwired into LAN adaptor cards in the same way as ordinary MAC addresses. They need to be configured at run-time, i.e. the host must tell its LAN adaptor which multicast MAC addresses to listen for. This must be done the first time a process on the host expresses interest in joining a particular IP multicast group. At this point, the host needs to map the IP multicast group address to a MAC multicast address which it can pass to the adaptor. The mapping must be identical in all hosts and in the router since all participants in the group must end up listening to the same MAC multicast address. This could be done through consultation with a server or, perhaps, a broadcast address resolution protocol could be devised. In fact, the decision made was that the mapping should be algorithmic.

IANA owns a block of Ethernet addresses in the range 00:00:5e:00:00:00 to 00:00:5e:ff:ff:ff and allocates the lower half of these for multicast. The Ethernet convention is that the first byte must be set to 01 to indicate a multicast address. Therefore the range we can use for multicast is 01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff . This means we have 23 bits to play with. These bits are set to the low-order 23 bits of the IP multicast group address to generate the MAC address. So, the address 224.20.5.1, which is e0.14.05.01 in hex, will map to the MAC address 01:00:5e:14:05:01. This is shown in binary below. (We have shown the bit ordering in the conventional way so that 0x01 appears as 00000001. In fact the bits are inserted into the Ethernet frame fields with each byte reversed - so, for example, that the first byte goes out on the wire as 10000000.)

Now, this is obviously not a 1-1 mapping and it is possible that we end up with two IP multicast groups on a LAN mapped to the same MAC multicast address. This is unfortunate, but not disastrous. It means that a host which has joined the group with address 224.20.5.1 will also receive datagrams intended for (say) 224.148.5.1 and will have to filter these out in software. However, many LAN interface cards do not filter multicast traffic efficiently, so this software filtering will need to be present in any case.

Multicast routing [1]





- First refinement
  - **reverse path broadcast (RPB)**
  - duplication

- Starting point: **flood**
  - creates looping

DigiComm II-8

IGMP allows routers to determine which multicast group addresses are of interest in the LAN. We now need a routing mechanism which ensures that all transmissions to a multicast address reaches the correct set of routers and hence the correct set of LANs. Therefore, we need an efficient dynamic multicast routing protocol. This turns out to be a hard problem to crack and is still the subject of much research. In this section we look at the problem and examine some of the protocols which have been developed to date.

The host S is transmitting to a multicast group address. Hosts B and E have joined the group and have announced the fact to R_B and R_E via IGMP. We need to calculate a spanning tree which interconnects the relevant routers. We can approach a solution through a series of refinements:

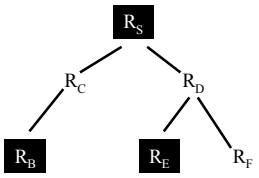
Starting point: Flood a multicast datagram to all neighbours except the one which sent it.

The problem with this is that we will get loops; R_C will forward to R_D, R_D to R_E and R_E to R_C. One way of solving this problem would be for each router to keep a list of the datagrams it has seen, check this each time it receives a datagram, and delete it if it is in the list. This is clearly not feasible for a multicast which might last several hours and involve millions of datagrams.

First refinement: Reverse Path Broadcasting

It turns out that routers already have quite a lot of the information they need in order to calculate a spanning tree simply from the operation of normal unicast routing protocols. In particular, each node will have a notion of the shortest path from itself to R_S - at the very least, they will know the length of this path and the identity of the first hop on it. This is true irrespective of which unicast routing protocol they are using. We can adopt the following rule - "flood a datagram that comes from the first-hop (on the path back to the source),but delete all others". Now, when R_C forwards to R_D, R_D will delete the datagram because it did not arrive from its "first-hop to source" (which, for R_D, is R_S itself). This technique is called **reverse path broadcasting (RPB)**.

Multicast routing [2]



- Second refinement
  - eliminate duplicates
  - need routing information

- Distance vector:
  - need next hop information
  - (or use **poisoned reverse**)
- Link state:
  - construction of all SP trees for all nodes possible
  - "tie-break" rules required

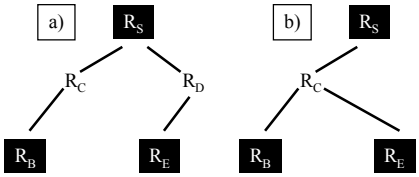
Second refinement: Duplicate elimination

As things stand, even with RPB, both R_C and R_D will forward a multicast datagram to R_E. R_E will delete one of these on the basis of the RPB rule. However, we have still wasted effort with a useless transmission to R_E. If R_C and R_D knew that R_E's path to R_S was via R_D (say) then R_C need not forward to R_E. How can R_C and R_D learn about R_E's paths? There are two cases to consider:

1) **distance-vector routing:** the distance-vectors R_E sends will contain distances but no indication of first-hop. One possibility is to modify the protocol to include this information. A second possibility is to make use of the **poisoned reverse** rule – send a hop count of "infinity" (i.e. value 16) back to the first hop on the route.

2) **link state routing:** link-state algorithms flood link-state information to all other nodes in the network. By this means, each node ends up with a complete picture of the state of every link in the network. In a unicast link-state algorithm, a node now proceeds to calculate a shortest path tree from itself to every other node in the network. In fact, each node has enough information to calculate shortest path trees for *every* node in the network. All the routers shown can calculate shortest-path trees with R_S as source. If we ensure that they all perform **precisely** the same calculation, they will all end up with the same result. This means that the calculation algorithm has to be formally part of the protocol and needs to specify unambiguous "tie-breaking" rules to select between equal length routes. For example, there are clearly two equal-length routes from R_E back to R_S – we must ensure that all routers make the same choice between them. This can be done, for example, by choosing the router with the numerically higher IP address.

Multicast routing [3]



- Third refinement:
  - **pruning**
  - need to refresh tree – **soft-state**
  - **reverse path multicasting (RPM)**
- RPM:
  - used in many multicast protocols
  - per-sender, per-group state

- Networks with no group members pruned from tree
- Must somehow allow tree to re-grow
- Soft-state:
  - timeout – re-flood
  - downstream nodes prune again
- Explicit **graft**:
  - downstream nodes join tree

DigiComm II-10

Third refinement: Pruning

By careful application of rules such as those above, it is possible for the routers to agree on a spanning-tree for the whole network. However, we are still wasting effort in forwarding datagrams to R_F when it has no group members. The solution is to introduce special **prune** messages.

When a router such as R_F receives a datagram for a multicast group which has no members on its attached LAN, it sends a prune message back to the router which forwarded the datagram. This router (R_D in this case) now adjusts its routing database to remove R_F from the tree. If we are in the situation of b), R_D will now know it has no-one to forward to, in which case it can, itself, send a prune message to R_S. With the addition of pruning, RPB becomes **reverse path multicasting (RPM)**. We need to have a method of restoring pruned links in case a host the other side of the link joins the group. We can either let prunes time-out (at which point the flow is restored and then, maybe, pruned again) or we can add explicit **graft** messages to the protocol. The former mechanism is a use of **soft-state** which is applied extensively in Internet protocols. Anticipating that state information is perishable in this way and building in mechanisms to restore it is fundamental to the operation of the Internet. It is key concept in making the Internet robust.

By using all these refinements, we can arrive at a reasonably efficient spanning tree. The two possibilities are shown. Both of these use shortest path routes from the source router (R_S) to R_B and R_E. On the face of it, the tree in diagram b) is more efficient since it involves one fewer transmission hop. However, this is not necessarily so since the network cloud might might be a LAN. If it is, then R_S can reach R_B and R_C with one transmission. We may then prefer diagram b) since it shares the forwarding load between the two routers.

DVMRP and the MBONE

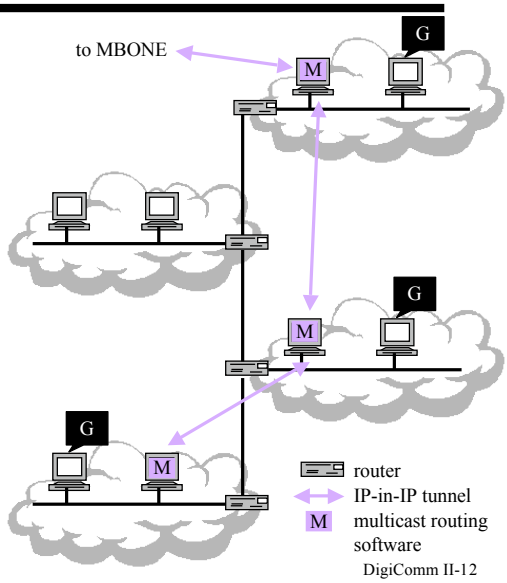
- DVMRP:
  - RPM
  - used on MBONE
- MBONE:
  - virtual overlay network
  - distance vector routing

MBONE Visualisation Tools
http://www.caida.org/Tools/Manta/
http://www.caida.org/Tools/Otter/Mbone/

The Internet's first multicast routing protocol - **Distance Vector Multicast Routing Protocol (DVMRP)** [RFC1075] – is a RPM protocol. It is based on RIP includes all the refinements outlined above, including the poisoned reverse trick. However, it suffers all the well-known problems of distance-vector algorithms and is regarded very much as a simple, interim solution intended to get Internet multicasting off the ground (in which it succeeded mightily). DVMRP has been used extensively in the **MBONE (multicast backbone)**.

MBONE configuration

- Routers not multicast aware:
  - use virtual network
- Multicast islands:
  - connected by virtual links
  - can not use normal routing info – use multicast hops
- IP tunnelling:
  - software runs on a host
  - *ad hoc* topology
- Use TTL for scope:
  - TTL expiry: **silent discard**
  - **administrative scope** possible



DigiComm II-12

The MBONE is a multicast network that spans the Internet, but consists of multicast islands connected together. It is a virtual network that is overlaid on the existing Internet unicast infrastructure. This approach was adopted in order to get experience of multicasting at a time when very few Internet routers actually supported it. The links between the multicast routers are **virtual links**. In order to send multicast datagrams along these links, they must be encapsulated within an ordinary (non-multicast) IP datagram with the destination address being the IP address of the multicast router at the end of the virtual link. This is called **IP-in-IP encapsulation** or **IP tunneling** [RFC1853]. This datagram is then forwarded by the normal routers in the ordinary way. On arrival, the multicast router extracts the multicast datagram and routes it according to the multicast group address it contains – it will have to re-encapsulate it in order to send it along the next virtual link. This arrangement is necessary because most "normal" routers do not yet understand multicast group addresses. In practice, the multicast routers are usually instances of the freely available **mrouted** program which runs on Sun workstations. The topology of the MBONE is *ad hoc*. To become part of the MBONE you simply negotiate the establishment of an IP tunnel between your site and a site that is already connected to the MBONE.

Unfortunately, when operating in an overlay network like the MBONE, we cannot use normal RIP distance-vectors directly. Normal RIP distance vectors will refer to the real nodes and links and not to the multicast nodes and virtual links. Therefore, DVMRP has to send its own distance-vectors containing information related to the MBONE itself. The poisoned reverse rule (which is optional in RIP) is used. In typical Internet fashion, DVMRP uses soft-state (explicit prunes) to maintain the tree.

To control the scope of transmission (how far they are transmitted on the network), the **time-to-live (TTL)** in the IPv4 header is used. The TTL is set by the transmitter to indicate how many MBONE router hops this packet should "live" for. When the TTL becomes zero, the packet is subject to **silent discard** – no ICMP TIME EXCEEDED message is generated to avoid packet implosion to the sender. The use of administrative scope by controlling the use of multicast addresses and controlling forwarding policy at multicast routers is also possible.

MOSPF

- Link-state algorithm
- RPM
- Intended for larger networks
- Soft-state:
  - router advertisement sent on group join
  - tree evaluated as routing update for a group arrives
- Still suffers from scaling problems:
  - a lot of state-required at each router
  - per-group, per-link information required

DigiComm II-13

A link-state based algorithm called **Multicast Extensions to Open Shortest Path First (MOSPF)** [RFC1584] is also available. MOSPF ends up being quite complex since it has to deal with OSPF's concepts of Areas and Autonomous Systems. It is designed to cope with large networks, however it still has some scaling problems. In larger networks, there could be hundreds of multicast groups in existence at any time. Only a few of these will pass through any particular node. Therefore it makes no sense for each node to pre-calculate trees for every possible source and every possible group. Instead, trees are calculated on the fly when a multicast datagram is received. Like DVMRP, MOSPF uses a soft-state approach, but does not need to use flood-and-prune (as DVMRP does). This is because when a router detects a group join from a leaf node, it send a routing update to the network to let other MOSPF routers know of the new group member. However, this is also MOSPF's short-coming: it needs to send many routing updates and holding routing information on a per-group, per-link basis, resulting in a large database of information. Also, it needs to evaluate the shortest-path algorithm for every source in the group, which is computationally expensive if there are many senders.

CBT

- Core router(s):
  - core distribution point for group
- Leaf sends IGMP request
- Local router sends *join request* to core
- *Join request* routed to core via normal unicast
- ✓ Intermediate routers note only incoming i/f and outgoing i/f per group

- ✓ Explicit join and leave:
  - no pruning
  - no flooding
- ✗ Distribution tree may be sub-optimal
- ✗ Core is bottleneck and single-point-of-failure:
  - additional core maybe possible
- Careful core placement required

DigiComm II-14

In **Core Based Trees (CBT)** [RFC2201] routers are explicitly designated as **core routers** for the group – in the simplest case, there will be a single core router. When a host wishes to join the group, it informs its local multicast router via IGMP. This router then forwards an explicit join message towards a core router. This is contained in a perfectly ordinary unicast IP datagram and so follows a route which has been established by unicast routing protocols in the normal way. Eventually a single shared tree results; we no longer require routers to be able to calculate different trees for each source as they had to for DVMRP and MOSPF. In fact, the state information retained by the on-tree routers is little more than the identity of the parent and child routers in the tree. Intermediate routers need only to maintain information about which interface a packet came in on, and which interface it was forwarded on. This information need is per group only, so the amount of information is $O(G)$ for multicast, as opposed to $O(G.S)$ for DVMRP and OSPF (where G is the number of groups and S is the number of senders). Also, join and leave request in CBT are explicit, and so CBT is quite well suited to sparsely populated groups.

The disadvantages with CBT are:

•that a tree may be sub-optimal and is heavily influenced by the location of the core; careful core location may be required

•the core router becomes a single point of failure, though a recovery mechanism is being added

PIM

- PIM:
  - can use any unicast routing protocol info
  - two modes: **dense mode** and **sparse mode**
- Dense mode:
  - RPM
  - flood-and-prune with explicit join
- Sparse mode:
  - similar to CBT
  - core (rendezvous point) or shortest-path possible
  - rendezvous point sends keep-alive
  - explicit graft to tree

DigiComm II-15

An important observation is that some groups are quite **dense** - heavily populated and in a relatively small geographical area. Other groups are **sparse** - lightly populated and spread right around the globe. For dense trees there is a lot of scope for link-sharing and it is worth exchanging state information frequently and expending computational effort to achieve this. For sparse trees there is unlikely to be much link-sharing. This has serious implications for a global Internet in which thousands of multicast groups might exist concurrently. The **Protocol Independent Multicast (PIM)** protocol incorporates these concepts having both dense and sparse modes - in fact it is really two protocols. PIM dense mode is a RPM algorithm. PIM sparse mode [RFC2362] uses an explicit graft mechanism to allow addition to a tree, similar to CBT.

Multicast address management

- Some addresses are reserved:
  - 224.0.0.1 all systems on this sub-net
    224.0.0.2 all routers on this sub-net
    224.0.0.4 all DVMRP routers
    (plus many others)
- No central control as in unicast addresses
- Others generated pseudo-randomly:
  - 28-bit multicast ID (last 28 bits of Class D address)

Unlike the unicast address space in which address allocation is controlled, the multicast address space is (almost) a free-for-all. Some addresses have been reserved and there are certain allocations of ranges of addresses for particular use. However, within these constraints, if a multicast addresses are chosen on an *ad hoc* basis. To help avoid clashes of different addresses, suggestion have been made as to how readily available information (such as time of day, IP address of the host initiating the group, etc.) might be used to produce the last 28 bits – the multicast ID – of a Class D address in a pseudo-random fashion.

Multimedia conferencing [1]

- Multimedia applications:
  - voice - *RAT*
  - video - *VIC*
  - text - *NTE*
  - whiteboard - *WBD*
- Support:
  - session directory - *SDR*
  - gateway - *UTG*
- All use **IP multicast**:
  - local – direct
  - wide area – MBONE
- RTP/RTCP
- IP multicast:
  - 224.2.0.0 - 224.2.255.255
  - different address per application per session
- Scoping:
  - IP TTL header field:
    16 local (site)
    47 UK
    63 Europe
    127 world
  - administrative

DigiComm II-17

UCL have been heavily involved with networked multimedia, especially multimedia conferencing. The standards for such applications are still developing. Example applications can be found at:

http://www-mice.cs.ucl.ac.uk/multimedia/

which include an audio tool (*RAT*), a video tool (*VIC*), a text editor (NTE) and a whiteboard (*WBD*). All these applications can run as standalone applications or can be run together within an integrated user interface. All are designed to operate over IP multicast for group communication (on a single LAN or across the MBONE), but unicast (one-to-one) communication is possible. Two additional support applications are a session directory (*SDR*) for a allowing advertisements multicast sessions on the MBONE and a transcoding gateway (*UTG*) for supporting dial-up users and allowing receiver heterogeneity.

All the applications use RTP and RTCP.

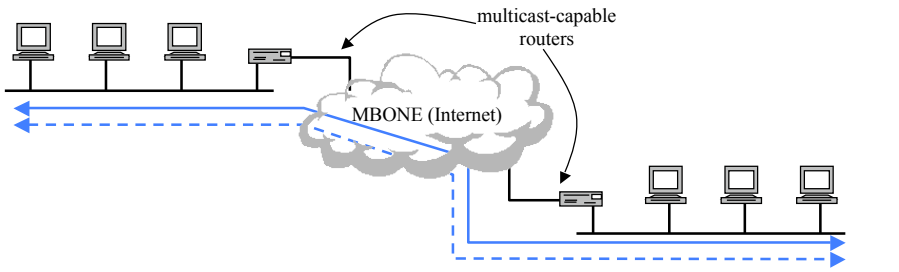
When used on the MBONE, the IP multicast addresses used are in the range 224.2.0.0 - 224.2.255.255. These have been designated by IANA for MBONE use by conferencing applications. Each application uses a different multicast address for each multicast **session**.

To restrict the extent of the transmission of the multicast traffic - its scope - the TTL field of the IPv4 header is used. This currently the most common mechanism used as it is simple to implement but there is a move to adopt a more administratively controlled approach, based on the actual values of multicast addresses being used.

A multicast conference may consist of the use of one or more of the user applications. The support applications may be required for configuration (*SDR*) and supporting LAN users (*UTG*).

Multimedia conferencing [2]

- Two multicast channels per application per session:
  - RTCP and RTCP
- Stand-alone - *ad hoc*:
  - individual applications
- Advertised conference:
  - SDR
  - configuration information



DigiComm II-18

Each application establishes a multicast session. This consists of two logical channels for multicast traffic, one for RTP traffic (the application data) and one for RTCP traffic (signalling and control). These two channels share the same multicast address but have different port numbers. The convention is that a multicast address, *D*, and an even port number greater than 5000, *K*, is chosen by the application user. The session then consists of two channels at *D*/*K* for the RTP traffic and *D*/(*K*+1) for the RTCP traffic.

This configuration is true whether or not the multicast session is to be local or to be sent across the MBONE. If the MBONE is to be used, the LAN requires a multicast capable router to distribute the local traffic and to act as a relay for any traffic from remote group members. The applications default to use local scope but this can be overridden through a command line option or via a configuration menu to change the TTL field as required (unless administrative scoping is being used).

Applications can be started individually as required. However, if the session is to be used on the MBONE, the **Session Directory Rendezvous (*SDR*)**, can be used to advertise the session beforehand, along with configuration parameters. *SDR* listens on some well-known multicast addresses and ports designated for *SDR* to pick up advertisements for other multicast sessions. *SDR* can be seen as the equivalent of a TV guide for the MBONE. When a session is advertised, it may include timing information (when the session is to be executed) as well as information about the media flows to be used. *SDR* can be configured to launch particular applications in order to process certain media types, e.g. *RAT* for audio.

Multimedia conferencing [3]

- Inter-flow synchronisation:
  - e.g. audio-video (lip-synch)
  - RTP/RCTP time-stamps
  - e.g. *RAT+VIC*: synch to *RAT* flow
- Inter-application communication:
  - conference bus
  - local communication (e.g. pipes)
- Heterogeneity:
  - data rates
  - (QoS)
- Gateway:
  - **transcoding**
  - multicast-to-unicast
  - supports dial-up users via BR-ISDN
  - (similar to H.323 Gatekeeper)

DigiComm II-19

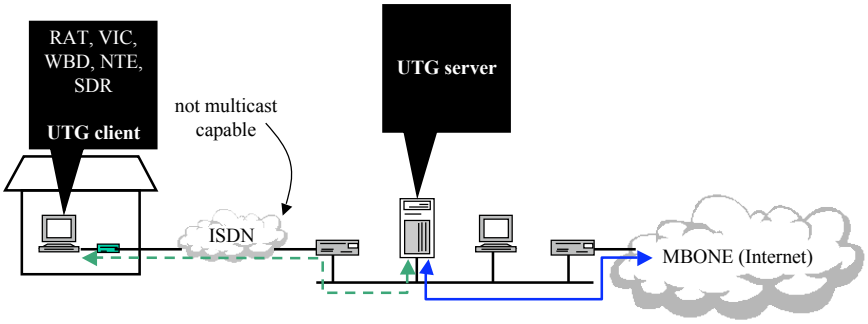
When several applications are used together to process different media flows, there my be a requirement to have inter-flow synchronisation, e.g. to achieve lip-synchronisation between audio and video in a virtual meeting. On the MBONE, as there is no timing signals from the network itself (unlike say, ISDN), the timing information for synchronisation must be built into higher layers. In fact, the timing information is carried in RTP packets and RTCP packets. NTP timestamps give the absolute time, and media-specific timestamps give the intra-flow synchronisation. By comparing the flow-specific timestamp with the NTP timetsamp, it is possible to achieve inter-flow synchronisation. Inter-process communication is required between the application instances on a particular host. This is typically achieved by the use of pipes (for example) and the use of a a well-defined set of message on a **conference bus**. The bus is a mechanism for allowing the transfer of control and configuration information between application instances. It can be seen as a signalling channel.

When many different users exist in a large multicast group, there is likely to be some heterogeneity in the capability of the end-systems and their connectivity. We have also seen that the MBONE leaf-nodes are assumed to be on a LAN. What if the end-user is a dial-up user, with lower data rates than a LAN and no multicast relay? To support such users, **transcoding gateways** can be used to transform the data in multicast flows and redistribute as required. Transcoding the is process of converting a media flow encoding into a different format, e.g. reducing the audio data rate by converting from PCM (64Kb/s) to ADPCM (32Kb/s). A transcoding gateway may perform such flow transformations, as well as act as a relay between a multicast-capable network and users not connected to multicast network, for example users connecting to an office network using BR-ISDN.

(Transcoding and providing relay services between connection-oriented and connectionless networks are two of the functions that are performed by the Gatekeeper function that is described in H.323.)

Multimedia conferencing [4]

- *UTG* server:
  - performs transcoding and relay
  - *UTG* clients register with server
- Dial-up users:
  - unicast to UTG client
  - local multicast at remote (client) host



DigiComm II-20

In the UCL toolkit, the transcoding functionality is provided by the **UCL Transcoding Gateway (*UTG*)**. The UTG consists of a client and a server. The server is a central point of contact for users wishing to have a transcoding and relaying service. The user executes the normal MBONE applications locally on their workstation. The workstation must be multicast capable. The user also executes a UTG client process that liases with the UTG server. The client registers with the server and provides information about its capability, e.g. data rate of the link, whether it requires a relay service, which audio and video formats it can support. It can also register which multicast groups the user wishes to join or it can use *SDR* via the *UTG* to dynamically join groups. The *UTG* server then provides the services requested.

For example, consider a dial-up user connecting using BR-ISDN (128Kb/s). This user would like to connect to a conference that will audio and video flows but knows that it will not be able see the full video rate as well as receive good quality audio. The *UTG* client at the remote site registers with the UG server at the main site (which could be, for example, a main office site for a teleworker, or an ISP PoP site). The *UTG* client asks that the *UTG* server provide a 32Kb/s audio flow and a 96Kb/s video flow. (Video flow data-rate reduction can be achieved by reducing the number of colours used, the frame refresh rate, the size of the picture, etc.) The actual multicast conference may be using 64Kb/s audio and 384Kb/s video. The *UTG* server joins the relevant multicast groups, transcodes the data audio flow and video flow, and the sends them to the *UTG* client using IP-in-IP tunnelling. The *UTG* client, on receiving the tunnelled packets, removes the inner multicast packet and redistributes locally.

Multimedia conferencing [5]

- *RAT*:
  - packet audio: time-slices
  - numerous audio coding schemes
  - redundant audio for repair
  - unicast or multicast
  - data-rate configurable
- *VIC*:
  - packet-video: frames
  - numerous video coding schemes
  - unicast or multicast
  - data-rate configurable

RAT and *VIC* are both multicast tools that use RTP to transport audio and video (respectively) across IP networks.

RAT sends time-slices of audio in 20ms, 40ms, 80ms or 160ms chunks (configurable). Larger time-slices are preferable, but packet loss then leaves larger gaps in the audio flow at the receiver. Numerous audio encoding techniques allow use of lower data-rate channels:

linear: 16-bit linear, 128Kb/s

PCM: μ-law companded Pulse Code Modulation, 64Kb/s

DVI: Digital Video Interactive (Intel), 32Kb/s

GSM: Global System for Mobile communication, 13.2Kb/s

LPC: Linear Predictive Coding, 5.8Kb/s

as we go down this list, quality decreases, as does required data-rate, and computational cost increases. All use 8KHz sampling. Typically, linear or PCM is used on the LAN, PCM or DVI over the Internet and GSM or LPC over a modem.

RAT also uses redundant encoding to allow repair of the audio stream to counter packet loss.

VIC sends single time-slices - single frames (not to be confused with link-level frames) - of video at anywhere between 1 frame per second (fps) and 30 fps (which is suitable for full motion video). It supports the following video encodings at various image sizes:

raw: 24-bit frame-by-frame dumps

JPEG: motion JPEG

MPEG: MPEG1

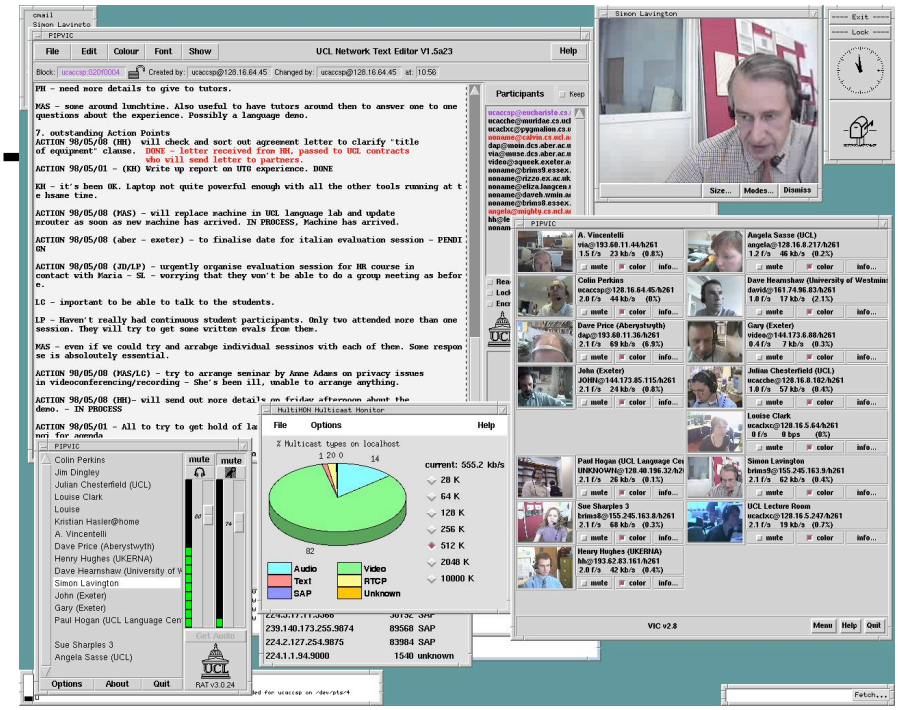
H.261: intra-frame H.261

H.263: intra-frame H.263

CellB: Sun Microsystems proprietary encoding

NV: Xerox PARC Network Video encoding

The frame rate and overall data rate can be adjusted independently for fine-grained control of the video transmission rate.



DigiComm II-22

Multicast conferencing [7]

- Floor control:
  - who speaks?
  - chairman control?
  - distributed control?
- Loose control:
  - one person speaks, grabs channel
- Strict control:
  - application specific, e.g.: lecture
- Resource reservation:
  - not supported on the MBONE(!)
  - ~500Kb/s per conference (using video)
- Per-flow reservation:
  - audio only
  - video only
  - audio and video

DigiComm II-23

In a conference, discussion or seminar, there is normally an orderly way that humans conduct themselves. This has to be available in multimedia conferencing tools and is called **floor control**. Floor controls requires communication between the humans using the applications as well as some automatic communication between the applications themselves. This latter communication is sometimes also referred to as **application-level signalling**. The floor control models are currently an area of research but two basic concepts exist:

• **loose floor control:** when anyone who speaks grabs the floor. This model is suitable for discussions or *ad hoc* meetings

• **strict floor control:** a chairman has explicit control controls which participants speak. This model is suitable for conferences or lectures.

To enable such control, the applications in the multicast groups must be able communicate. This is enabled through signalling between the applications based on the chose floor control model.

Resource reservation may also be required in a conference in order to allow adequate capacity for audio and video flows. A typical conference with several several tens of participants using audio and head-and-shoulders 8fps video may require around 500Kb/s for operation. The MBONE does not currently support resource reservation, so it may not be possible to have an audio and video conference across the MBONE (remember that the MBINE is an overlay network across the Internet so sees the same QoS as other Internet applications.) Typically, it may be required that some sites in the conference remove the video stream in order to allow continued participation in the conference. Within a LAN environment, if there is a light load on the network then a single-LAN conference is possible without requiring resource reservation (as loss, delay and jitter are likely to be low). Indeed it is often not possible to make resource reservations in a LAN environment based on certain network technology (e.g. Ethernet).

If reservation is used, it could be applied independently to each of the audio video and data flows. For example, human users are fare more intolerant to loss in video flows than audio flows so a reservation could be made for the video flow and the audio (with its relatively modest data rate requirements) could continue operation at "best-effort" service.

QoS-based routing

What is QoS-based routing?

- Traditional routing:
  - destination address chooses path/route
  - routers have one "optimal" path to destination
  - routing metrics are single values
- QoS routing:
  - multiple paths possible
  - alternative paths have different QoS properties
  - routing updates include QoS parameter information
  - use destination address, source address, ToS, etc.
- RSVP/INTSERV/DIFFSERV:
  - signalling may still be required

DigiComm II-25

Traditionally, routing involves routers exchanging information about connectivity/reachability with a single metric to indicate some kind of "cost" that makes sense to the routing table algorithm. This metric may be hop count (e.g. RIP/DVMRP) or link cost (e.g. OSPF/MOSPF). The router uses this single metric to create a single "optimal" path to a destination. The path is optimal with respect to the single metric being used. Other, sub-optimal paths may exist, but they are not used.

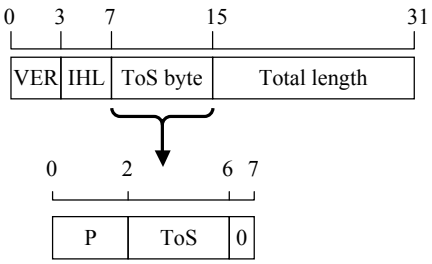
With **QoS-based routing** (also called **constraint-based routing**), multiple paths are possible between sender and destination, and the choice of which path is followed is based on policy criteria selected by looking at packet header information such as source address, the ToS/DIFFSERV byte, etc. This requires that the router hold information about multiple paths per destination, running its routing algorithm multiple times to set up this information, and to include various QoS-related metrics in its routing updates. This is a non-trivial change to the operation of the router and the network as a whole.

A good overview of the issues in QoS-based routing is presented in [RFC2386].

Note that the aim of QoS routing is to indicate that paths with suitable QoS characteristics are available, but other mechanisms (such as RSVP and/or INTSERV and/or DIFFSERV) may still be required in order to ensure that resources along that path remain for the duration of the flow.

IPv4 ToS byte

- IPv4 header – ToS byte:
  - 3-bit precedence, P
  - 4-bit ToS
- Precedence:
  - 000: lowest
  - 111: highest
- ToS – flags:
  - 1xxx: minimise delay
  - x1xx: maximise throughput
  - xx1x: maximise reliability
  - xxx1: minimise cost (£)
  - 0000: "normal" service



- Not widely used:
  - no global agreement
  - (some use in Intranets)
- RFC1349 – now historic:
  - superseded by DIFFSERV
  - not compatible with ECN

DigiComm II-26

In [RFC1349] is documented as way of using the 8-bit **Type of Service (ToS)** byte in the IPv4 header to provide a class of service indicator. The byte is plit into two fields, a precedence indicator, P, and a set of flags indicating the type of service (ToS) required for the packet. P takes values from 0 – 7, with 0 being the lowest precedence and 7 being the highest. The ToS flags indicate whether the packet requires minimum delay, maximum throughput, maximum reliability (low loss) or minimum (monetary) cost. The terms "maximum" and "minimum" are not that well defined.

This system was not widely use across the Internet, but found its way into use in some intra-domain (intra-AS) routing mechanisms. Although [RFC1349] is now historic (superseded by the DIFFSERV work), it serves to illustrate how we might perform QoS routing by indicating, in a packet, some simple handling requirements.

Multi-metric routing

- Use multiple metrics:
  - minimum delay path
  - maximum throughput path
  - maximum reliability path
  - minimum cost path
- Example – OSPF:
  - QoS parameters passed in link-state packets
  - ToS byte used in IPv4
  - multiple executions of shortest-path algorithm
- Sequential filtering:
  - filter paths using metrics
- Granularity of QoS:
  - can be per-flow, but requires much state in routers
- Router overhead:
  - more per packet processing
  - larger router updates
  - more state at routers
  - possibility of instability during routing updates

Multiple metrics can be used to establish multiple paths based on QoS parameter criteria. For example, OSPF [RFC2328] allows the use of delay, throughput loss and cost information to establish routes. Information about these parameters is included in link-state packets emitted by OSPF routers. When routing tables are evaluated, the the SP algorithm is run multiple times, once for each metric and the resulting routes are stored. When a packet arrives with a ToS marking, say, for "maximum reliability" in its ToS markings, the router makes a path selection based on the routing table evaluated using the loss/reliability information.

In general, where multiple selection criteria are specified, sequential filtering can be used to select a path. For example, if "high throughput" and "low delay" are selected, initially some candidate paths are selected by applying the "high throughput" criteria only. Then, these candidate paths are filtered based on the "low delay" criteria so selecting the path(s) with both "high throughput" and "low delay". This allows flexibility but requires extra processing, compared to using a single metric to describe/summarise both "high throughput" and "low delay". The added processing could increase the latency of transmission, at least for the first packet in a flow, before the selected path is cached to the routers forwarding table.

The granularity of such an approach is generally kept quite coarse in order to keep processing overhead low. It could be possible to define polices that select packets based on header information down to a per-flow level, but this would introduce a large amount of extra processing and storage of state at the routers.

General disadvantages of multi-metric routing are that there is an increased overhead on the router, in terms of per-packet processing, generating and processing router updates, holding state for paths. There is also the possibility on instability and routing loops during updates, or if inconsistent implementation of routing policy causing conflicts in routing behaviour, e.g. routers in the same domain find they have different routing tables even though they have seen the same routing updates.

Route pinning and path pinning

- Dynamic routing:
  - path change → QoS change
- Keep route fixed for flow?

Route pinning

- Ensure that route is fixed while packet forwarding in progress
- Disrupts normal routing behaviour
- May cause congestion conditions

Path pinning

- Allow route to change:
  - existing flows remain on fixed path
  - new flows use new route
- Allow different paths for different flows:
  - pin separate flows to separate paths
- Inconsistency:
  - could affect stability if flow is long lived
- (Use of RSVP?)

DigiComm II-28

We have already noted that changes in QoS for a flow can occur due to the changes in the network path being followed by the packets in the flow. This is a natural consequence of the dynamic routing changes that give IP its robustness. However, routing changes can often occur when a the existing route is still serviceable, but just not "optimal". Remember that, traditionally, routers only compute one optimal route based on the routing metric. This itself could cause instability as much traffic may be re-routed, but also this will normally result in an observable QoS change. Holding a routes constant – **pinning routes** – for the duration of a flow (e.g. based on some caching/time-out criteria) might help to alleviate this. However, this could disrupt the network stability, as routers with active flows may not change their routing tables, whilst other routers in the domain do, and routing loops and congestion effects could result.

An alternative is to use **path pinning**, allowing the routing table to be updated as normal but keeping knowledge of the current path for exiting flows. So, any existing flows continue to use the same path but new flows would use a different path. This could still lead to instability and consistency in the network if there are many long-lived flows that hold paths pinned for a long time.

Another proposal is to use RSVP to signal path pinning for some flows, and where paths really do have to change, to try and use RSVP to establish provide some likeness of QoS one the new path as was present on the old path.

These are still research areas.

MPLS

- Multi-protocol label switching:
  - fast forwarding
  - IETF WG
- MPLS is an enabling technology:
  - claimed to help scaling
  - claimed to increase performance
  - forwarding still distinct from routing
- Intended for use on NBMA networks:
  - e.g. ATM, frame-relay
- Many supporters:
  - e.g. Cisco
- Many cynics:
  - introduces much more complexity into routers
  - more state required at routers
  - (non)-interaction with routing protocol operation may cause instability
  - may not work very well at high speeds
  - other IP-level mechanisms exist

DigiComm II-29

The Multi-Protocol label Switching (MPLS) WG of the IETF is seeking to define a standard that will support fast-forwarding mechanisms.

It is intended that the use of MPLS in place of traditional IP forwarding will allow better performance and scaling in certain IP network scenarios. Its is intended that such mechanisms will help scaling an and performance of IP networks in certain environments, i.e. where it is likely that the layer-2 technology will offer a faster forwarding mechanism than the layer-3 forwarding of IP.

MPLS is designed to be complementary to existing routing mechanisms. Indeed, routing information is used to establish the forwarding entries used by MPLS.

Although independent of any particular bearer technology and any particular layer-3 technology, there is particular interest in finding MPLS solutions tailored to provide IP-over-ATM and IP-over-FR (Frame Relay) – Non-Brodcast Multiple Access (NBMA) network technologies.

Intra-domain routing

- Can use agreed single/multiple metrics
- Allow autonomy in domains to remain
- Should indicate disruptions to QoS along a path
- Must accommodate best-effort traffic:
  - no modification to existing, best-effort applications
- Optionally support multicast:
  - allow receiver heterogeneity and shared reservations
- Still a research issue

DigiComm II-30

Intra-domain QoS routing may be achievable by using mechanisms such as OSPF with ToS or DIFFSERV or traffic engineering in the underlying network. Multi-metric routing is possible with OSPF as we have already said.

The requirements listed in [RFC2386] for intra-domain QoS routing include:

• allow autonomy of operation within domains, as exist at the current time

• flow must be routed along a path with QoS requested or requested/indicated or a notification must be generated to say that such QoS capability can not provided at this time

• indications of QoS disruption should be signalled during the lifetime of a flow if disruption is due topological changes

• must accommodate best-effort flows without requiring changes to the applications that generate them

• optionally support multicast and allow receiver heterogeneity and shared reservations

A QoS routing protocol that fulfils alls these criteria does not exist … yet.

Inter-domain

- **Must be scaleable**
- QoS-routing should not be highly dynamic:
  - few router updates, relatively small amounts of information
  - may have to rely on traffic engineering and capacity planning
- Must not constrain intra-domain routing mechanisms
- Allow QoS information aggregation
- Optionally support multicast

DigiComm II-31

For inter-domain routing the key property that any QoS-based routing mechanism must possess is scalability. As there are large amounts of traffic between AS boundaries and the stability of the boundary routers is key to connectivity, we must ensure that such nodes are not subject to excessive load/processing due to the QoS-based routing mechanisms. To ensure this, [RFC2386] lists the following requirements:

• QoS routing mechanisms must not be highly dynamic, there must be relatively few routing updates with small amounts of information. So, there may be a need to rely on more traditional forms of engineering, such as capacity planning, in order to ensure that border routers are kept lightly loaded

• metrics should be agreed and consistent. Internal AS/domain specific metrics may need to be mapped to metrics that have global semantics

• path computation should not be constrained, and be allowed to use QoS request for flows, path metrics, local policy, heuristics as well as other reachability information available from normal operation

• flow aggregation should be supported as it will not be practical to maintain state for thousands of individual flows. Mechanisms must be defined to ensure that aggregate flow descriptions for QoS are consistent with the combined requirements of the individual flows so composition and comparison rules for QoS metrics must be established

• optionally support multicast

QoS-based routing for multicast

- Reliable multicast:
  - retransmissions from sender does not scale
  - research issue
- QoS for multicast:
  - need to support widely/sparsely dispersed groups
  - dynamic membership changes
  - must scale across domains (across AS boundaries)
  - should allow heterogeneity in group
  - support for shared reservations
  - research issue

DigiComm II-32

QoS for multicast is still a research issue.

For the moment, there is work in progress to develop reliable multicast, for example the Reliable Multicast Transport (RMT) WG of the IETF. Normal, sender-based based retransmissions coupled with acknowledgements form the receiver does not scale to the multicast environment.

RSVP/INTSERV was designed with multicast very much in mind but we have already seen it has scaling problems and does not support receiver heterogeneity very well. Also, reservation merging is inflexible. So, [RFC2386] lists these key requirements for QoS-based multicast routing:

• support widely and sparsely dispersed groups

• allow dynamic membership changes for groups

• scale across domains

• allow heterogeneity within groups

• support shared reservation styles

Needless to say, this is still a research issue.

Summary

- Many-to-many communication:
  - IP multicast
  - DVMRP, MOSPF, CBT, PIM
  - conferencing example
- QoS-based routing:
  - multi-metric
  - route/path pinning
  - intra-domain and inter-domain
  - QoS-based routing for multicast

Traffic management

An Engineering Approach to Computer Networking

An example

- Executive participating in a worldwide videoconference
- Proceedings are videotaped and stored in an archive
- Edited and placed on a Web site
- Accessed later by others
- During conference
  - Sends email to an assistant
  - Breaks off to answer a voice call

What this requires

- For video
  - *sustained bandwidth of at least 64 kbps*
  - *low loss rate*
- For voice
  - *sustained bandwidth of at least 8 kbps*
  - *low loss rate*
- For interactive communication
  - *low delay (< 100 ms one-way)*
- For playback
  - *low delay jitter*
- For email and archiving
  - *reliable bulk transport*

What if…

- A million executives were simultaneously accessing the network?
  - What *capacity* should each trunk have?
  - How should packets be *routed*? (Can we spread load over alternate paths?)
  - How can different traffic types get different *services* from the network?
  - How should each endpoint *regulate* its load?
  - How should we *price* the network?

- These types of questions lie at the heart of network design and operation, and form the basis for **traffic management.**

Traffic management

- Set of policies and mechanisms that allow a network to *efficiently* satisfy a *diverse* range of service requests
- Tension is between diversity and efficiency
- Traffic management is necessary for providing *Quality of Service (QoS)*
  - Subsumes congestion control (congestion == loss of efficiency)

Why is it important?

- One of the most challenging open problems in networking
- Commercially important
  - AOL ‘burnout’
  - Perceived reliability (necessary for infrastructure)
  - Capacity sizing directly affects the bottom line
- At the heart of the next generation of data networks
- Traffic management = Connectivity + Quality of Service

Outline

- Economic principles
- Traffic classes
- Time scales
- Mechanisms
- Some open problems

Basics: utility function

- Users are assumed to have a *utility function* that maps from a given quality of service to a level of satisfaction, or utility
  - Utility functions are private information
  - Cannot compare utility functions between users
- *Rational* users take actions that maximize their utility
- Can determine utility function by observing preferences

Example

- Let $u = S - a\,t$
  - u = utility from file transfer
  - S = satisfaction when transfer infinitely fast
  - t = transfer time
  - a = rate at which satisfaction decreases with time
- As transfer time increases, utility decreases
- If $t > S/a$, user is worse off! (reflects time wasted)
- Assumes linear decrease in utility
- S and a can be experimentally determined

Social welfare

- Suppose network manager knew the utility function of every user
- *Social Welfare* is maximized when some combination of the utility functions (such as sum) is maximized
- An economy (network) is *efficient* when increasing the utility of one user must necessarily decrease the utility of another
- An economy (network) is *envy-free* if no user would trade places with another (better performance also costs more)
- Goal: maximize social welfare
  - subject to efficiency, envy-freeness, and making a profit

Example

- Assume
  - Single switch, each user imposes load 0.4
  - A's utility: 4 - d
  - B's utility : 8 - 2d
  - Same delay to both users
- Conservation law
  - 0.4d + 0.4d = C => d = 1.25 C => sum of utilities = 12-3.75 C
- If B's delay reduced to 0.5C, then A's delay = 2C
  - Sum of utilities = 12 - 3C
- Increase in social welfare need not benefit everyone
  - A loses utility, but may pay less for service

Some economic principles

- A single network that provides heterogeneous QoS is better than separate networks for each QoS
  - unused capacity is available to others
- Lowering delay of delay-sensitive traffic increased welfare
  - can increase welfare by matching service menu to user requirements
  - BUT need to know what users want (signaling)
- For typical utility functions, welfare increases more than linearly with increase in capacity
  - individual users see smaller overall fluctuations
  - can increase welfare by increasing capacity

Principles applied

- A single wire that carries both voice and data is more efficient than separate wires for voice and data
  - ADSL
  - IP Phone
- Moving from a 20% loaded10 Mbps Ethernet to a 20% loaded 100 Mbps Ethernet will still improve social welfare
  - increase capacity whenever possible
- Better to give 5% of the traffic lower delay than all traffic low delay
  - should somehow mark and isolate low-delay traffic

The two camps

- Can increase welfare either by
  - matching services to user requirements *or*
  - increasing capacity blindly
- Which is cheaper?
  - no one is really sure!
  - small and smart vs. big and dumb
- It seems that smarter ought to be better
  - otherwise, to get low delays for some traffic, we need to give *all traffic* low delay, even if it doesn't need it
- But, perhaps, we can use the money spent on traffic management to increase capacity
- We will study traffic management, assuming that it matters!

Traffic models

- To align services, need to have some idea of how users or aggregates of users behave = traffic model
  - e.g. how long a user uses a modem
  - e.g. average size of a file transfer
- Models change with network usage
- We can only guess about the future
- Two types of models
  - measurements
  - educated guesses

Telephone traffic models

- How are calls placed?
  - call arrival model
  - studies show that time between calls is drawn from an exponential distribution
  - call arrival process is therefore *Poisson*
  - memoryless: the fact that a certain amount of time has passed since the last call gives no information of time to next call
- How long are calls held?
  - usually modeled as exponential
  - however, measurement studies show it to be *heavy tailed*
  - means that a significant number of calls last a very long time

Internet traffic modeling

- A few apps account for most of the traffic
  - WWW
  - FTP
  - telnet
- A common approach is to model apps (this ignores distribution of destination!)
  - time between app invocations
  - connection duration
  - # bytes transferred
  - packet interarrival distribution
- Little consensus on models
- But two important features

Internet traffic models: features

- LAN connections differ from WAN connections
  - Higher bandwidth (more bytes/call)
  - longer holding times
- Many parameters are heavy-tailed
  - examples
    - # bytes in call
    - call duration
  - means that a *few* calls are responsible for most of the traffic
  - these calls must be well-managed
  - also means that *even aggregates with many calls not be smooth*
  - can have long bursts
- New models appear all the time, to account for rapidly changing traffic mix

Outline

- Economic principles
- Traffic classes
- Time scales
- Mechanisms
- Some open problems

Traffic classes

- Networks should match offered service to source requirements (corresponds to utility functions)
- Example: telnet requires low bandwidth and low delay
  - utility increases with decrease in delay
  - network should provide a low-delay service
  - or, telnet belongs to the low-delay *traffic class*
- Traffic classes encompass both *user requirements* and *network service offerings*

Traffic classes - details

- A basic division: guaranteed service and best effort
  - like flying with reservation or standby
- Guaranteed-service
  - utility is zero unless app gets a minimum level of service quality
    - bandwidth, delay, loss
  - open-loop flow control with admission control
  - e.g. telephony, remote sensing, interactive multiplayer games
- Best-effort
  - send and pray
  - closed-loop flow control
  - e.g. email, net news

GS vs. BE (cont.)

- Degree of synchrony
  - time scale at which peer endpoints interact
  - GS are typically *synchronous* or *interactive*
    - interact on the timescale of a round trip time
    - e.g. telephone conversation or telnet
  - BE are typically *asynchronous* or *non-interactive*
    - interact on longer time scales
    - e.g. Email
- Sensitivity to time and delay
  - GS apps are *real-time*
    - performance depends on wall clock
  - BE apps are typically indifferent to real time
    - automatically scale back during overload

Traffic subclasses (roadmap)

- ATM Forum
  - based on sensitivity to bandwidth
  - GS
    - CBR, VBR
  - BE
    - ABR, UBR
- IETF
  - based on sensitivity to delay
  - GS
    - intolerant
    - tolerant
  - BE
    - interactive burst
    - interactive bulk
    - asynchronous bulk

ATM Forum GS subclasses

- Constant Bit Rate (CBR)
  - constant, cell-smooth traffic
  - mean and peak rate are the same
  - e.g. telephone call evenly sampled and uncompressed
  - constant bandwidth, variable quality
- Variable Bit Rate (VBR)
  - long term average with occasional bursts
  - try to minimize delay
  - can tolerate loss and higher delays than CBR
  - e.g. compressed video or audio with constant quality, variable bandwidth

ATM Forum BE subclasses

- Available Bit Rate (ABR)
  - users get whatever is available
  - zero loss if network signals (in RM cells) are obeyed
  - no guarantee on delay or bandwidth
- Unspecified Bit Rate (UBR)
  - like ABR, but no feedback
  - no guarantee on loss
  - presumably cheaper

IETF GS subclasses

- Tolerant GS
  - nominal mean delay, but can tolerate “occasional” variation
  - not specified what this means exactly
  - uses *controlled-load* service
    - book uses older terminology (predictive)
  - even at “high loads”, admission control assures a source that its service “does not suffer”
  - it really is this imprecise!
- Intolerant GS
  - need a worst case delay bound
  - equivalent to CBR+VBR in ATM Forum model

IETF BE subclasses

- Interactive burst
  - bounded asynchronous service, where bound is qualitative, but pretty tight
    - e.g. paging, messaging, email
- Interactive bulk
  - bulk, but a human is waiting for the result
  - e.g. FTP
- Asynchronous bulk
  - junk traffic
  - e.g netnews

Some points to ponder

- The only thing out there is CBR and asynchronous bulk!
- These are application requirements. There are also organizational requirements (link sharing)
- Users needs QoS for other things too!
  - billing
  - privacy
  - reliability and availability

Outline

- Economic principles
- Traffic classes
- Time scales
- Mechanisms
- Some open problems

Time scales

- Some actions are taken once per call
  - tell network about traffic characterization and request resources
  - in ATM networks, finding a path from source to destination
- Other actions are taken during the call, every few round trip times
  - feedback flow control
- Still others are taken very rapidly,during the data transfer
  - scheduling
  - policing and regulation
- Traffic management mechanisms must deal with a range of traffic classes at a range of time scales

Summary of mechanisms at each time scale

- Less than one round-trip-time (cell-level)
  - Scheduling and buffer management
  - Regulation and policing
  - Policy routing (datagram networks)
- One or more round-trip-times (burst-level)
  - Feedback flow control
  - Retransmission
  - Renegotiation

Summary (cont.)

- Session (call-level)
  - Signaling
  - Admission control
  - Service pricing
  - Routing (connection-oriented networks)
- Day
  - Peak load pricing
- Weeks or months
  - Capacity planning

Outline

- Economic principles
- Traffic classes
- Mechanisms at each time scale
  - Faster than one RTT
    - scheduling and buffer management
    - regulation and policing
    - policy routing
  - One RTT
  - Session
  - Day
  - Weeks to months
- Some open problems

Renegotiation

Renegotiation

- An option for guaranteed-service traffic
- Static descriptors don't make sense for many real traffic sources
  - interactive video
- Multiple-time-scale traffic
  - burst size B that lasts for time T
  - for zero loss, descriptors (P,0), (A, B)
    - P = peak rate, A = average
  - T large => serving even slightly below P leads to large buffering requirements
  - one-shot descriptor is inadequate

Renegotiation (cont.)

- Renegotiation matches service rate to traffic
- Renegotiating service rate about once every ten seconds is sufficient to reduce bandwidth requirement nearly to average rate
  - works well in conjunction with optimal smoothing
- Fast buffer reservation is similar
  - each burst of data preceded by a reservation
- Renegotiation is not free
  - signaling overhead
  - call admission ?
    - perhaps measurement-based admission control

RCBR

- Extreme viewpoint
- All traffic sent as CBR
- Renegotiate CBR rate if necessary
- No need for complicated scheduling!
- Buffers at edge of network
  - much cheaper
- Easy to price
- Open questions
  - when to renegotiate?
  - how much to ask for?
  - admission control
  - what to do on renegotiation failure

Outline

- Economic principles
- Traffic classes
- Mechanisms at each time scale
  - Faster than one RTT
  - One RTT
  - Session
    - Signaling
    - Admission control
  - Day
  - Weeks to months
- Some open problems

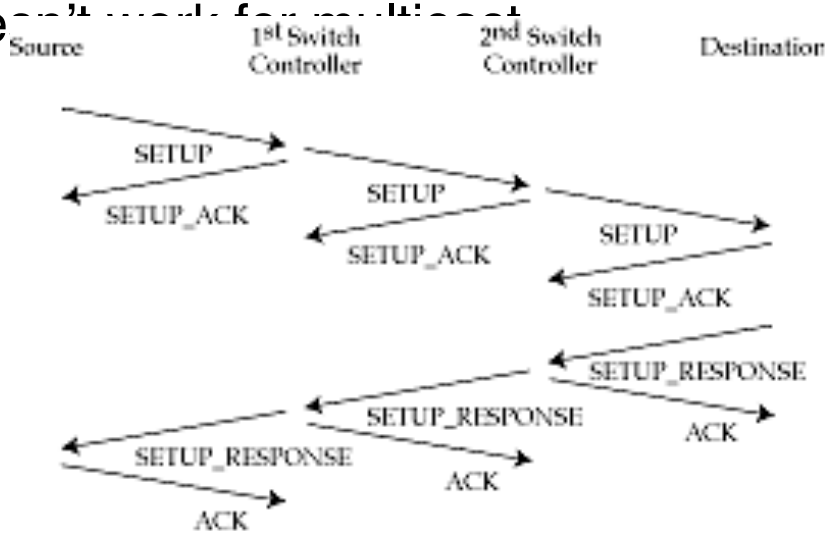
Signaling

Signaling

- How a source tells the network its utility function
- Two parts
  - how to carry the message (transport)
  - how to interpret it (semantics)
- Useful to separate these mechanisms

Signaling semantics

- Classic scheme: sender initiated
- SETUP, SETUP_ACK, SETUP_RESPONSE
- Admission control
- Tentative resource reservation and confirmation
- Simplex and duplex setup
- Doe[illegible]



Resource translation

- Application asks for end-to-end quality
- How to translate to per-hop requirements?
  - E.g. end-to-delay bound of 100 ms
  - What should be bound at each hop?
- Two-pass
  - forward: maximize (denial!)
  - reverse: relaz
  - open problem!

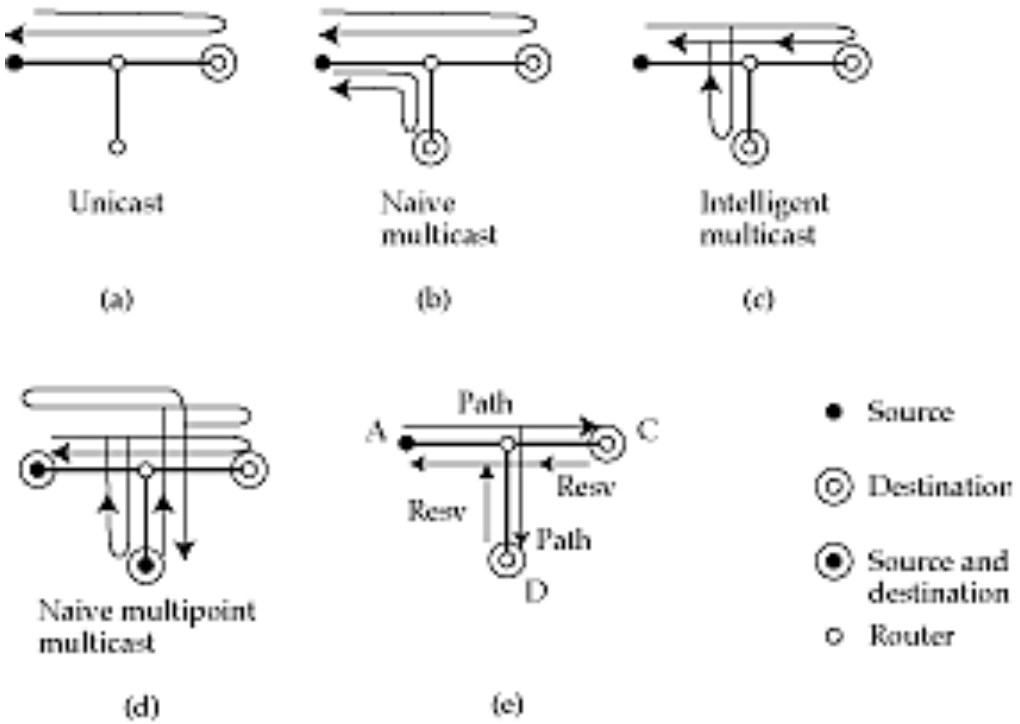
Signaling: transport

- Telephone network uses Signaling System 7 (SS7)
  - Carried on Common Channel Interoffice Signaling (CCIS) network
  - CCIS is a datagram network
  - SS7 protocol stack is loosely modeled on ISO (but predates it)
- Signaling in ATM networks uses Q.2931 standard
  - part of User Network Interface (UNI)
  - complex
  - layered over SSCOP (a reliable transport protocol) and AAL5

Internet signaling transport: RSVP

- Main motivation is to efficiently support multipoint multicast with resource reservations
- Progression
  - Unicast
  - Naïve multicast
  - Intelligent multicast
  - Naïve multipoint multicast
  - RSVP

RSVP motivation



Multicast reservation styles

- Naïve multicast (source initiated)
  - source contacts each receiver in turn
  - wasted signaling messages
- Intelligent multicast (merge replies)
  - two messages per link of spanning tree
  - source needs to know all receivers
  - and the rate they can absorb
  - doesn't scale
- Naïve multipoint multicast
  - two messages per source per link
  - can't share resources among multicast groups

RSVP

- Receiver initiated
- Reservation state per group, instead of per connection
- PATH and RESV messages
- PATH sets up next hop towards source(s)
- RESV makes reservation
- Travel as far back up as necessary
  - how does receiver know of success?

Filters

- Allow receivers to separate reservations
- Fixed filter
  - receive from eactly one source
- Dynamic filter
  - dynamically choose which source is allowed to use reservation

Soft state

- State in switch controllers (routers) is periodically refreshed
- On a link failure, automatically find another route
- Transient!
- But, probably better than with ATM

Why is signaling hard ?

- Complex services
- Feature interaction
  - call screening + call forwarding
- Tradeoff between performance and reliability
- Extensibility and maintainability

Outline

- Economic principles
- Traffic classes
- Mechanisms at each time scale
  - Faster than one RTT
  - One RTT
  - Session
    - Signaling
    - Admission control
  - Day
  - Weeks to months
- Some open problems

Admission control

Admission control

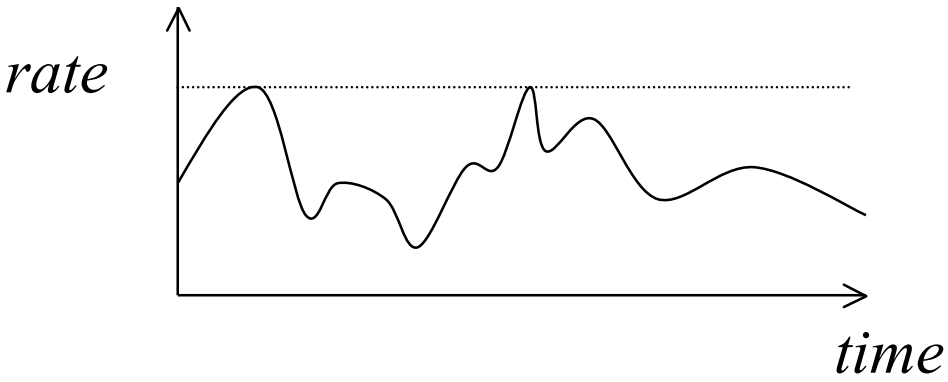
- Can a call be admitted?
- CBR admission control
  - simple
  - on failure: try again, reroute, or hold
- Best-effort admission control
  - trivial
  - if minimum bandwidth needed, use CBR test

VBR admission control

- VBR
  - peak rate differs from average rate = *burstiness*
  - if we reserve bandwidth at the peak rate, wastes bandwidth
  - if we reserve at the average rate, may drop packets during peak
  - key decision: how much to overbook
- Four known approaches
  - peak rate admission control
  - worst-case admission control
  - admission control with statistical guarantees
  - measurement-based admission control

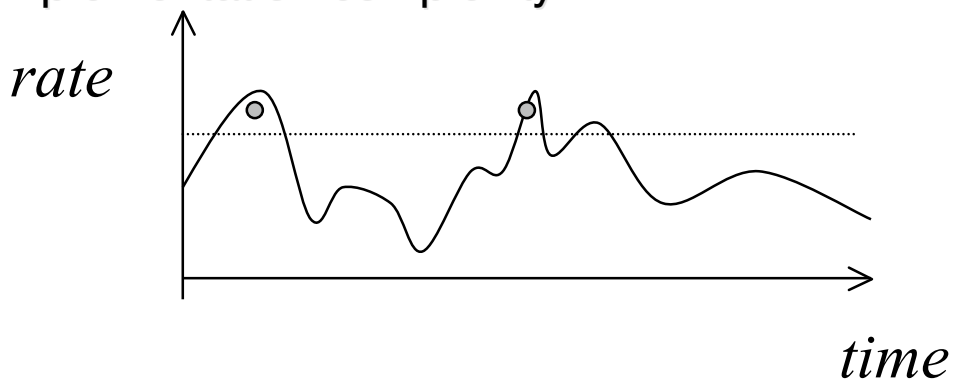
1. Peak-rate admission control

- Reserve at a connection's peak rate
- Pros
  - simple (can use FIFO scheduling)
  - connections get zero (fluid) delay and zero loss
  - works well for a small number of sources
- Cons
  - wastes bandwidth
  - peak rate may increase because of scheduling jitter



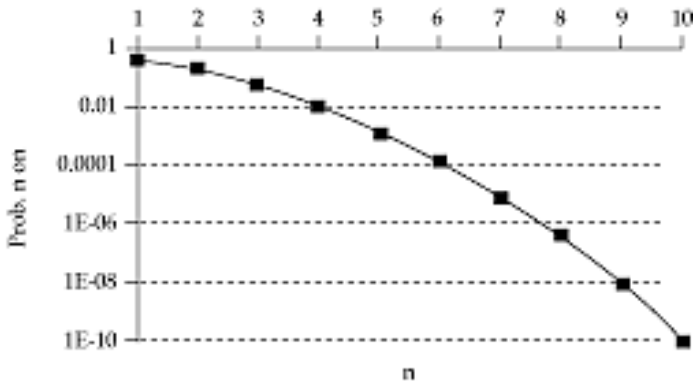
2. Worst-case admission control

- Characterize source by ‘average’ rate and burst size (LBAP)
- Use WFQ or rate-controlled discipline to reserve bandwidth at average rate
- Pros
  - may use less bandwidth than with peak rate
  - can get an end-to-end delay guarantee
- Cons
  - for low delay bound, need to reserve at more than peak rate!
  - implementation complexity



3. Admission with statistical guarantees

- Key insight is that as # calls increases, probability that multiple sources send a burst decreases
  - sum of connection rates is increasingly smooth
- With enough sources, traffic from each source can be assumed to arrive at its average rate
- Put in enough buffers to make probability of loss low



3. Admission with statistical guarantees (contd.)

- Assume that traffic from a source is sent to a buffer of size *B* which is drained at a constant rate *e*
- If source sends a burst, its delay goes up
- If the burst is too large, bits are lost
- *Equivalent bandwidth* of the source is the rate at which we need to drain this buffer so that the probability of loss is less than *l* and the delay in leaving the buffer is less than *d*
- If many sources share a buffer, the equivalent bandwidth of each source decreases (why?)
- Equivalent bandwidth of an ensemble of connections is the sum of their equivalent bandwidths

3. Admission with statistical guarantees (contd.)

- When a source arrives, use its performance requirements and current network state to assign it an equivalent bandwidth
- Admission control: sum of equivalent bandwidths at the link should be less than link capacity
- Pros
  - can trade off a small loss probability for a large decrease in bandwidth reservation
  - mathematical treatment possible
  - can obtain delay bounds
- Cons
  - assumes uncorrelated sources
  - hairy mathematics

4. Measurement-based admission

- For traffic that cannot describe itself
  - also renegotiated traffic
- *Measure* 'real' average load
- Users tell peak
- If peak + average < capacity, admit
- Over time, new call becomes part of average
- Problems:
  - assumes that past behavior is indicative of the future
  - how long to measure?
  - when to forget about the past?

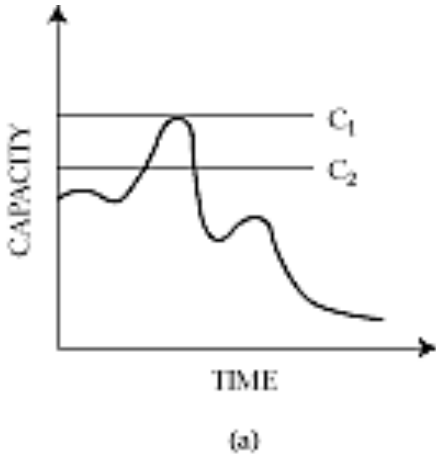
Outline

- Economic principles
- Traffic classes
- Mechanisms at each time scale
  - Faster than one RTT
  - One RTT
  - Session
  - Day
  - Weeks to months
- Some open problems

Peak load pricing

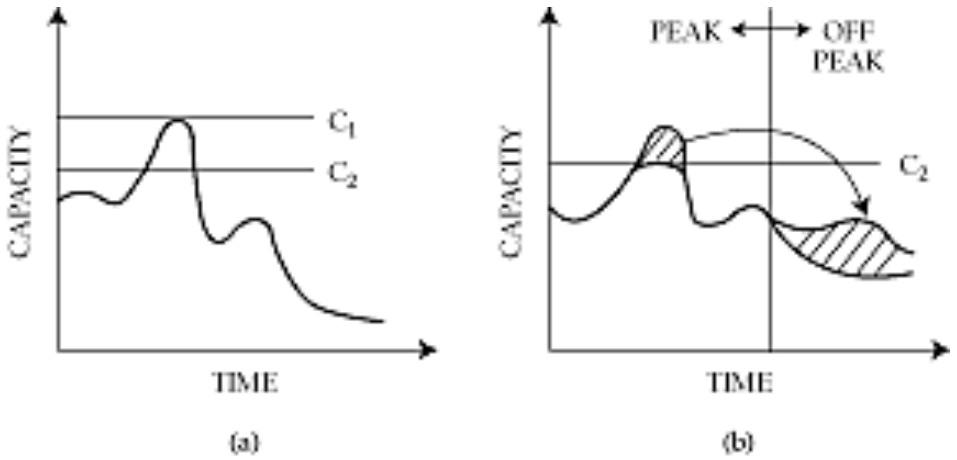
Problems with cyclic demand

- Service providers want to
  - avoid overload
  - use all available capacity
- Hard to do both with cyclic demand
  - if capacity C1, then waste capacity
  - if capacity C2, overloaded part of the time



Peak load pricing

- Traffic shows strong daily peaks => cyclic demand
- Can shift demand to off-peak times using pricing
- Charge more during peak hours
  - price is a *signal* to consumers about network preferences
  - helps both the network provider and the user



Example

- Suppose
  - network capacity = C
  - peak demand = 100, off peak demand = 10
  - user's utility = -total price - overload
  - network's utility = revenue - idleness
- Price = 1 per unit during peak and off peak times
  - revenue = 100 + 10 = 110
  - user's utility = -110 -(100-C)
  - network's utility = 110 - (C - off peak load)
  - e.g if C = 100, user's utility = -110, network's utility = 20
  - if C = 60, user's utility = -150, network's utility = 60
  - increase in user's utility comes as the cost of network's utility

Example (contd.)

- Peak price = 1, off-peak price = 0.2
- Suppose this decreases peak load to 60, and off peak load increases to 50
- Revenue = 60*1 + 50*0.2 = 70
  - lower than before
- But peak is 60, so set C = 60
- User's utility = -70 (greater than before)
- Network's utility = 60 (same as before)
- Thus, with peak-load pricing, user's utility increases at no cost to network
- Network can gain some increase in utility while still increasing user's utility

Lessons

- Pricing can control user's behavior
- Careful pricing helps both users and network operators
- Pricing is a *signal* of network's preferences
- Rational users help the system by helping themselves

Outline

- Economic principles
- Traffic classes
- Mechanisms at each time scale
  - Faster than one RTT
  - One RTT
  - Session
  - Day
  - Weeks to months
- Some open problems

Capacity planning

Capacity planning

- How to modify network topology, link capacity, and routing to most efficiently use existing resources, or alleviate long-term congestion
- Usually a matter of trial and error
- A more systematic approach:
  - measure network during its busy hour
  - create traffic matrix
  - decide topology
  - assign capacity

1. Measure network during busy hour

- Traffic ebbs and flows during day and during week
- A good rule of thumb is to build for the worst case traffic
- Measure traffic for some period of time, then pick the busiest hour
- Usually add a fudge factor for future growth
- Measure bits sent from each endpoint to each endpoint
  - we are assuming that endpoint remain the same, only the internal network topology is being redesigned

2. Create traffic matrix

- # of bits sent from each source to each destination
- We assume that the pattern predicts future behavior
  - probably a weak assumption
    - what if a web site suddenly becomes popular!
- Traffic over shorter time scales may be far heavier
- Doesn't work if we are adding a new endpoint
  - can assume that it is similar to an existing endpoint

3. Decide topology

- Topology depends on three considerations
  - *k*-connectivity
    - path should exist between any two points despite single node or link failures
  - geographical considerations
    - some links may be easier to build than others
  - existing capacity

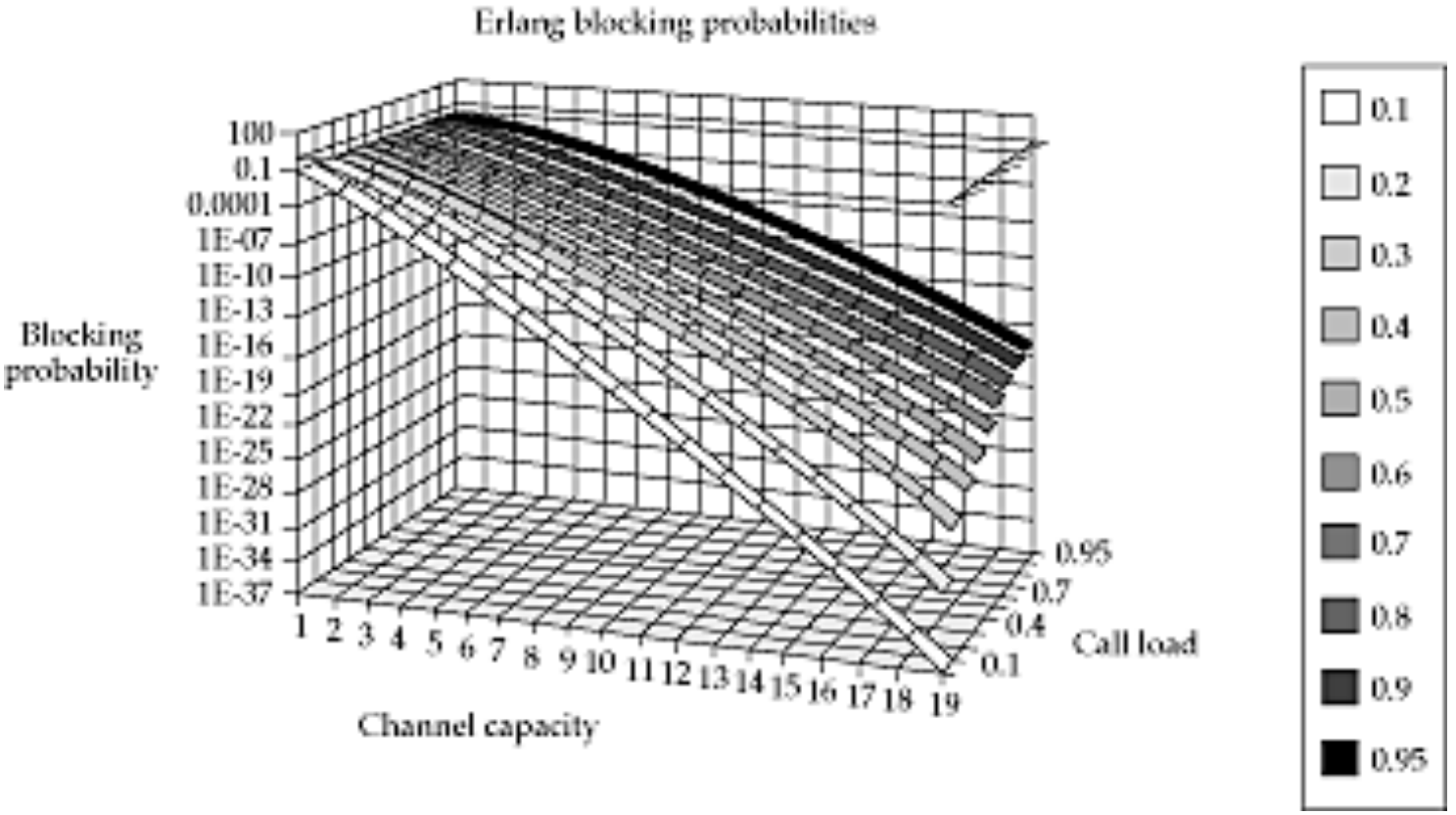
4. Assign capacity

- Assign sufficient capacity to carry busy hour traffic
- Unfortunately, actual path of traffic depends on routing protocols which measure instantaneous load and link status
- So, we cannot directly influence path taken by traffic
- Circular relationship between capacity allocation and routing makes problem worse
  - higher capacity link is more attractive to routing
  - thus carries more traffic
  - thus requires more capacity
  - and so on…
- Easier to assign capacities if routing is *static* and links are always up (as in telephone network)

Telephone network capacity planning

- How to size a link so that the call blocking probability is less than a target?
- Solution due to Erlang (1927)
- Assume we know mean # calls on a trunk (in erlangs)
- Mean call arrival rate = l
- Mean call holding time = m
- Then, call load A = lm
- Let trunk capacity = N, infinite # of sources
- Erlang's formula gives blocking probability
  - e.g. N = 5, A = 3, blocking probability = 0.11
- For a fixed load, as N increases, the call blocking probability decreases exponentially

Sample Erlang curves



Capacity allocation

- Blocking probability along a path
- Assume traffic on links is independent
- Then, probability is product of probability on each link
- Routing table + traffic matrix tells us load on a link
- Assign capacity to each link given load and target blocking probability
- Or, add a new link and change the routing table

Capacity planning on the Internet

- Trial and error
- Some rules of thumb help
- Measurements indicate that sustained bandwidth per active user is about 50 Kbps
  - add a fudge factor of 2 to get 100 Kbps
- During busy hour, about 40% of potential users are active
- So, a link of capacity C can support 2.5C/100 Kbps users
- e.g. 100 Mbps FDDI ring can support 2500 users

Capacity planning on the Internet

- About 10% of campus traffic enters the Internet
- A 2500-person campus usually uses a T1 (closest to 10 Mbps) and a 25,000-person campus a T3 (close to 100 Mbos)
- Why?
  - regional and backbone providers throttle traffic using pricing
  - e.g. T1 connection to Uunet costs about $1500/month
  - T3 connection to Uunet costs about $50,000/month
  - Restricts T3 to a few large customers
- Regionals and backbone providers buy the fastest links they can
- Try to get a speedup of 10-30 over individual access links

Problems with capacity planning

- Routing and link capacity interact
- Measurements of traffic matrix
- Survivability

Outline

- Economic principles
- Traffic classes
- Mechanisms at each time scale
- Some open problems

Some open problems

Six open problems

- Resource translation
- Renegotiation
- Measurement-based admission control
- Peak-load pricing
- Capacity planning
- A metaproblem

1. Resource translation

- Application asks for end-to-end quality in terms of bandwidth and delay
- How to translate to resource requirements in the network?
- **Bandwidth is relatively easy, delay is hard**
- One approach is to translate from delay to an equivalent bandwidth
  - can be inefficient if need to use worst case delay bound
  - average-case delay usually requires strong source characterization
- Other approach is to directly obtain per-hop delay bound (for example, with EDD scheduling)
- How to translate from end-to-end to per-hop requirements?
  - Two-pass heuristic

2. Renegotiation

- Static descriptors don't make sense for interactive sources or multiple-time scale traffic
- Renegotiation matches service rate to traffic
- Renegotiation is not free- incurs a signaling overhead
- Open questions
  - when to renegotiate?
  - how much to ask for?
  - admission control?
  - what to do on renegotiation failure?

3. Measurement based admission

- For traffic that cannot describe itself
  - also renegotiated traffic
- Over what time interval to measure average?
- How to describe a source?
- How to account for nonstationary traffic?
- Are there better strategies?

4. Peak load pricing

- How to choose peak and off-peak prices?
- When should peak hour end?
- What does peak time mean in a global network?

5. Capacity planning

- Simultaneously choosing a topology, link capacity, and routing metrics
- But routing and link capacity interact
- What to measure for building traffic matrix?
- How to pick routing weights?
- Heterogeneity?

6. A metaproblem

- Can increase user utility either by
  - service alignment *or*
  - overprovisioning
- Which is cheaper?
  - no one is really sure!
  - small and smart vs. big and dumb
- It seems that smarter ought to be better
  - for example, to get low delays for telnet, we need to give *all traffic* low delay, even if it doesn't need it
- But, perhaps, we can use the money spent on traffic management to increase capacity!
- Do we really need traffic management?

Macroscopic QoS

- Three regimes
  - scarcity - micromanagement
  - medium - generic policies
  - plenty - are we there yet?
- Example: video calls
- Take advantage of law of large numbers
- Learn from the telephone network

Miscellania

- Some topics that don't quite fit…

Lecture objectives

Broader Considerations for real-time applications:

- Systems Questions:
  - Scaling & Stability
  - Mobility
  - Management
- Non-technical Questions
  - economic and user aspects
    - Pricing and Provisioning
  - implementation context:
    - **Active Networks**
    - **MPLS/"Circuits"**

Scaling and Stability

References

•Vern Paxson, End-to-end Routing Behavior in the Internet
ACM CCR, vol. 26, no. 4, pp. 25-38, Oct. 1996.
http://www.acm.org/sigcomm/ccr/archive/1996/conf/paxson.html

•Floyd, S., and Jacobson, V.,
The Synchronization of Periodic Routing Messages
IEEE/ACM ToN, V.2 N.2, p. 122-136, April 1994.
href="http://www.aciri.org/floyd/papers/sync_94.ps.Z

~

Scaling (or Complexity) - 1

- All mechanisms that we add to IP Have some cost - we would like ideally, this cost to be O(C) (Order constant) - I.e. if we add QoS, the cost in terms of messages, router and end system memory, router and end system CPU should just be a constant, ideally! In practice though…
- Its likely that some mechanisms will be O(n), where n is the number of…
- end systems or routers - or can we do better?
- Diff-serve versus Int-serve is based around this...

Scaling (or Complexity) - 2

- So per flow-queues are at least going to have a data structure in a router per active pair (tree) of sender/receiver(s)
- Whereas per class queues have some data structure per class although edge systems may have to do per source policing and/or shaping - which implies that overall, we may have O(ln(n))
- Need tostate overall architecture to see overall system costs!

Stability - 1

- Ideally, Traffic, whether user or management (e.g. signaling, routing updates etc) should be stable.
- Conditions for stability complex - basically need to do control theoretic analaysis
- Even if oscillatory, should converge or be bounded, not diverge….
- Reasons for instability or divergence:
  - Positive Feedback
  - Correlation/phase effects...

Stability - 2

- End-to-end congestion control systems are designed to be stable - damped feedback
- Routing systems are designed to be stable - randomized timers
- QoS systems (especially call admision and QoS routing) need to be stable too.
- Needs careful thought and smart engineering…
- e.g. don’t want to do alternate path routing and admission control on same timescales.

Mobility

Reference:

- Anup Kumar Talukdar, B. R. Badrinath and Arup Acharya, "Integratedservices packet networks with mobile hosts: architecture and performance",Wireless Networks, vol. 5, no. 2, 1999
- Jarkko Sevanto, Mika Liljeberg, and Kimmo Raatikainen, "Introducingquality-of-service and traffic classes into wireless mobile networks",Proceedings of first ACM international workshop on Wireless mobile multimedia, October 25-30, 1998, Dallas, TX USA

- Links…
- Patterns…
- Resources...

Mobile 1 - Wireless Links

- Wireless links can have variable characteristics, e.g. delay, throughput, loss
- Offering hard QoS is hard
- GPRS and other wireless links offer shared media
- May be able to coordinate QoS via shared media MAC layer management and handoff management (see ISSLL work in IETF) - requires cooperation
- Opposite of trend on fixed nets (e.g. shared media LANs moving to switched approaches!)

Mobile 2 - Patterns

- Mobile access patterns may be quite different from fixed ones
- Simply don’t know yet, but may entail lots more state refresh (e.g. re-sending RSVP path/resv triggered by moves)
- Mobiel multicast with source or sink moving may be complex (involve re-building tree)

Mobile 3 - Resources

- Some QoS approaches are based on the netwrk running largely underloaded
- e.g. EF and AF may only work for IP telephony if it constitutes a small part of traffic
- This is not the case on many wireless links today.
- Need to look at hard QoS schemes - particularly for low latency (e.g. interactive voice/games) - even down to the level of limited frame/packet sizes - leads to interleave problems...

Management

All this needs managing by someone, at the very least the policies need configuration…..

Management-1

- User account management
- QoS auditing
- MIBs for queues, signalling protocols, etc
- risk analysis and trend prediction tools
- security (authentication and privacy aspects of payment for qos - see next)

Pricing and Provisioning

Reference: http://www.statslab.cam.ac.uk/~richard/PRICE

Pricing 1

- If you don't charge for QoS, won't everyone just ask for first-class?
- What are the users paying for?
- What are they prepared to pay?
- If you do charge, how to stop arbitrage (rich buy all the bandwidth and then re-sell at different price).

Pricing 2

- Typically, access fee can cover actual cost of infrastructure
- Bill is often just an *incentive scheme* (to stop users hogging capacity in a class)
- Parameters:
  - time of day and duration
  - distance (geographic, provider hops, AS-count?)
  - capacity
  - delay (iff possible) and jitter control
  - Loss (possibly)

Pricing 3

- Can price by effective capacity
- Do we want to vary price with network conditions? (optimal in theory but complex - too complex for user - in practice) - *congestion pricing*
- security associated with payment and policing necessary
- Predictable bills are often more important than cheapest fare (c.g. mobile phones).

Provisioning

- Users don’t like being refused access (prefer degraded service, but…)
- Need to dimension network for the user satisfaction and revenue levels
- Base on traffic measured. Look at frequency of overload or call rejection for RSVP…
- IP telephony - can (if pricing and patterns match) base on Erlang models…traditional - may not apply - e.g. either or both of call and packet arrival independence may be wrong...

Implementation Novelties

Active Networks &
MPLS

Active Networks

Reference: D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, G. J.Minden, "A Survey of Active Network Research, IEEE Communications Mag.,Vol. 35, No. 1, pp 80-86. January 1997

- Active networks subject of large DARPA program, and quite a few european projects.

- Interpose processing of user data in network path by dynamically moving code there….radical idea based in strong distributed computation
- Originated in observation that it has become very hard in telephony and IP networks to deploy new services of any kind due to scale (and inflexibility) of the infrastructure.

Active Networks 2

- Weak model just puts code in place at application level points -either call handling (e.g. dynamic singlaing protocol code -*switchware, switchlets* IEEE programmable networks work) or at application level relays (e.g. non transparent caches)
- Strong model - re-programs switches on the fly possibly per packet - packet header is now code for VM in switch instead of data for fixed program in switch.

Active Networks 3

- Jury is out on AN
- Looks like at least some ideas will make it through to prime time though….
- Main problems
  - with strong AN is code performance, safety and liveness
  - with weak AN is management - could be very useful for generalized VPNs though...

MPLS

- Datagrams Meets Circuits
- Based on strong idea of “flow”

Performance

- Getting data from source to destination(s) as fast as possible
- Higher data rates required for:
  - large files …
  - multimedia data
  - real-time data (video)
- **Fast forwarding**
- Not the same as QoS provisioning, but closely linked

Forwarding vs. Routing

- Routers have to:
  - maintain routes
  - forward packets based on routing information
- Forwarding:
  - moving a packet from an input port to an output port
  - make a forwarding decision based on route information
  - get the packet to an output port (or output queue) fast
- Routing:
  - knowing how to get packets from source to destination

IP forwarding

- Packet arrives (input buffer?)
- Check destination address
- Look up candidate routing table entries:
  - destination address
  - routing entry
  - address mask
- Select entry:
  - longest prefix match selects next hop
- Queue packet to output port (buffer)

Flows

- A sequence of IP packets that are semantically related:
  - packet inter-arrival delay less than 60s
- Flows may be carrying QoS sensitive traffic
- Many thousands of flows could exist when you get to the backbone
- Detect flows and use label-based routing:
  - make forwarding decisions easier
  - make forwarding decisions faster

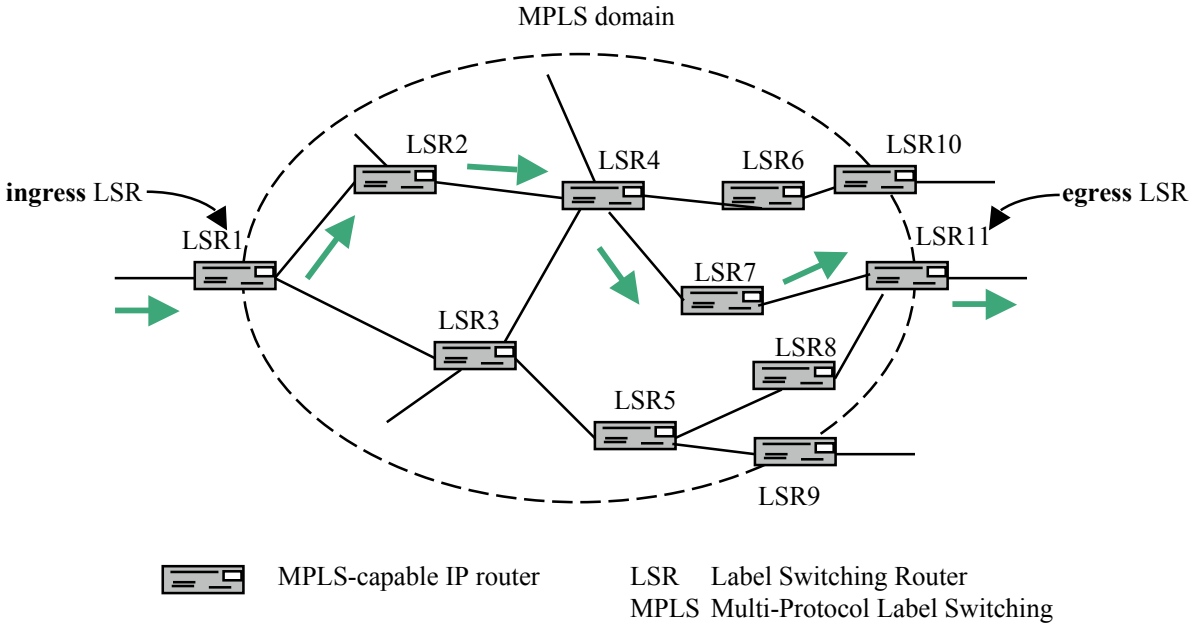
MPLS

- Multi-protocol label switching:
  - fast forwarding
  - IETF WG
- MPLS is an enabling technology:
  - helps scaling
  - increases performance
  - forwarding still distinct from routing
- Intended for use on NBMA networks:
  - e.g. ATM, frame-relay

MPLS architecture [1]

- IETF work in progress - requirements:
  - integrate with existing routing protocols
  - support unicast, multicast, QoS, source routing
- MPLS uses label-swapping
- Flows are labelled:
  - special shim header
  - can use existing labels in bearer technology (e.g. VCI)
- **LSR (Label Switching Router)**:
  - simple, fast link-level forwarding

MPLS architecture [2]



Label switching

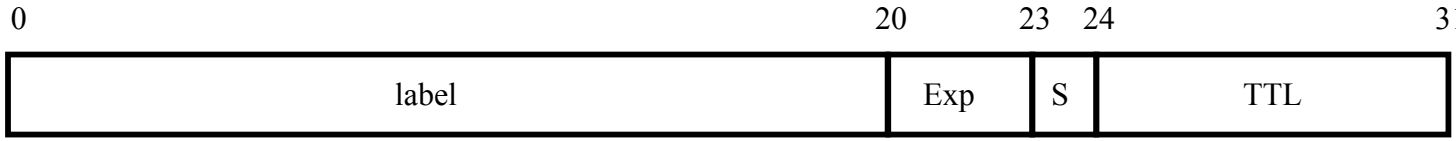
- Packet enters ingress router
  - lookup label: **Forwarding Equivalency Class (FEC)**
  - packet forwarded with label
- At next hop (next LSR):
  - label used in table lookup: **LIB** and **NHLFE**
  - new label assigned
  - packet forwarded with new label
- Saves on conventional look-up at layer 3
- Need label distribution mechanism

Labels [1]

- Label:
  - short
  - fixed-length
  - local significance
  - exact match for forwarding
- Forwarding equivalency class (FEC):
  - packets that share the same next hop share the same label (locally)
  - packets with the same FEC and same route: **streams**

Labels [2]: shim header

- Generic: can be used over any NBMA network
- Inserted between layer-2 and layer-3 header
- label: 20 bits
- Exp: 3 bits (use not yet fully defined - CoS)
- S: 1 bit stack flag (1 indicates last in stack)
- TTL: 8 bits



Label granularity

- IP prefix:
  - aggregation of several routes
- Egress router:
  - all IP destinations with common egress router for LSP
- Application flow:
  - per-flow, end-to-end
- Others possible:
  - e.g. host pairs, source tree (multicast)

Label distribution [1]

- Routing information used to distribute labels:
  - piggy-back label info on existing protocols?
- Performed by downstream nodes
- Each MPLS node:
  - receives outgoing label mapping from downstream peer
  - allocates/distributes incoming labels to upstream peers
- **Label Distribution Protocol (LDP):**
  - LDP peers (LDP adjacency)

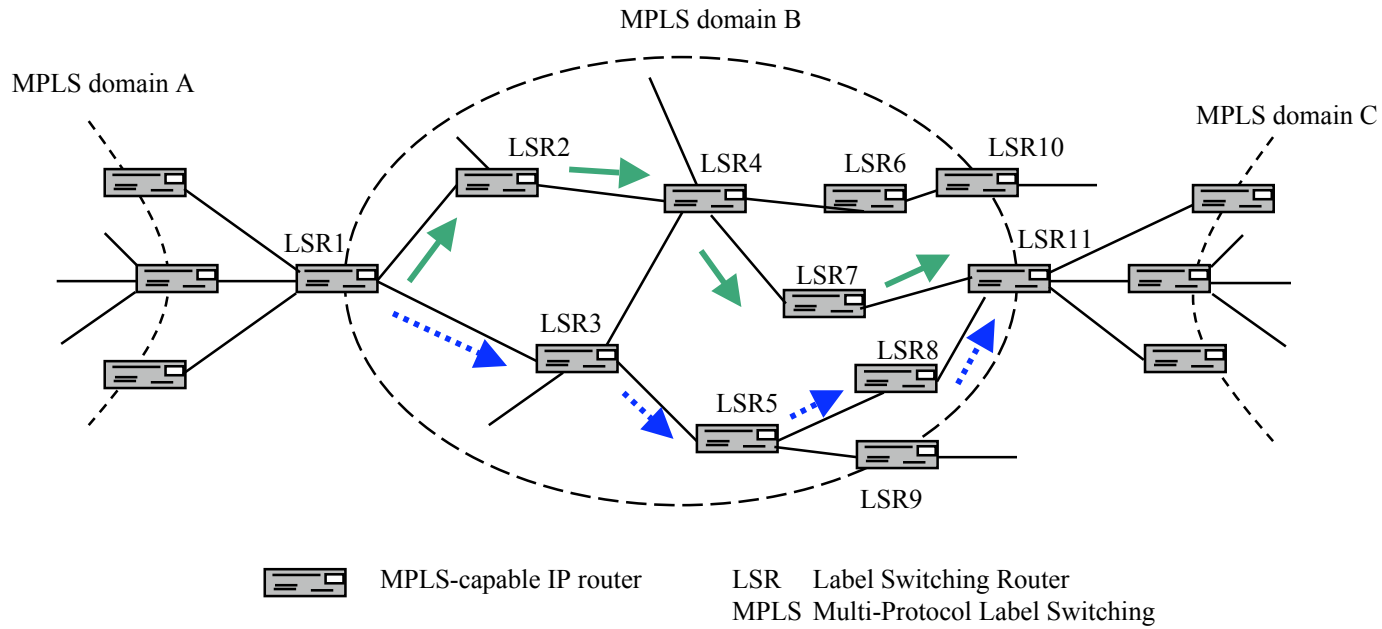
Label distribution [2]

- Distribution of label info from LSR only if:
  - egress LSR
  - LSR has an outgoing label
- **Downstream**: LSR allocates and distributes
- **Downstream-on-demand**: upstream LSR requests allocation from a downstream node
- Address prefix-based FEC/forwarding:
  - **independent** distribution: any node in LSP
  - **ordered** distribution: egress LSR

Label stacking [1]

- Two mechanisms:
  - equivalent to IP source routing
  - hierarchical routing
- Multiple labels are stacked by the ingress LSR
- LSRs along the route can pop the stack:
  - makes forwarding even faster

Label stacking [2]



MPLS-like implementations

- Control-based:
  - tag-switching: cisco
  - ARIS (Aggregated Routing and IP Switching): IBM
  - IP-Navigator (Ascend)
- Request-based: RSVP
- Traffic-based:
  - IP switching: Ipsilon
  - CSR (cell switch router): Toshiba
- Many others …

Other performance issues

- Router architectures
- Fast route-table lookup
- Fast packet-classification (QoS)
- Better address aggregation (e.g. CIDR, IPv6)
- Traffic engineering (differentiated services)
- Faster boxes or smarter software?

Summary

- Reference: Scott Shenker, "Fundamental design issues for the future Internet",IEEE J. Selected Areas Comm, 13 (1996), pp 1176-1188
- QoS isn’t that simple!
- Push something out of one part of the architecture, it will show up somewhere else
- e.g. if you remove statelessness by ading RSVP, you need to do congestion control of signaling
- e.g. if you remove adaption by adding connection admission (e.g. for TCP), users start adapting.