Common Protocols

Digital Communications II

Michaelmas Term 2007
Based on Prof. Jon Crowcroft's notes, and thus transitively on S. Keshav's "An Engineering Approach to Computer Networking"

Explore some common protocols

- Much discussion of 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

<table>
<thead>
<tr>
<th>Data Plane</th>
<th>Control Plane (SS7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>Voice/Fax ASE/ISDN-UP</td>
</tr>
<tr>
<td>Session</td>
<td>TCAP</td>
</tr>
<tr>
<td>Transport</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>SCCP/MTP-3</td>
</tr>
<tr>
<td>Datalink</td>
<td>SDH/Sonet/PDH MTP-2</td>
</tr>
<tr>
<td>Physical</td>
<td>Many MTP-1</td>
</tr>
</tbody>
</table>

Traditional digital transmission

- Long distance trunks carry multiplexed calls
- Standard multiplexing levels
- Digital transmission hierarchy
  - DS0 is 64Kbps (a single call)

<table>
<thead>
<tr>
<th>US and Japan</th>
<th>E Europe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Signal level (i.e. multiplexing)</td>
<td>T class.</td>
</tr>
<tr>
<td>DS1</td>
<td>T1</td>
</tr>
<tr>
<td>DS2</td>
<td>T2</td>
</tr>
<tr>
<td>DS3</td>
<td>T3</td>
</tr>
<tr>
<td>DS4</td>
<td>T4</td>
</tr>
<tr>
<td>DS5</td>
<td>T5</td>
</tr>
</tbody>
</table>
Plesiochronous hierarchy

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

![Diagram: 2:1 MUX, A's data, B's data, Overhead]

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 – requires use atomic clocks
- Justification uses pointers

<table>
<thead>
<tr>
<th>Data Rate (Mbps)</th>
<th>SONET Name</th>
<th>SDH Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>51.84</td>
<td>OC-1</td>
<td>STM-0</td>
</tr>
<tr>
<td>155.52</td>
<td>OC-3</td>
<td>STM-1</td>
</tr>
<tr>
<td>466.56</td>
<td>OC-9</td>
<td>n/a</td>
</tr>
<tr>
<td>622.08</td>
<td>OC-12</td>
<td>STM-4</td>
</tr>
<tr>
<td>933.12</td>
<td>OC-18</td>
<td>n/a</td>
</tr>
<tr>
<td>1244.16</td>
<td>OC-24</td>
<td>STM-8</td>
</tr>
<tr>
<td>1866.24</td>
<td>OC-36</td>
<td>n/a</td>
</tr>
<tr>
<td>2488.32</td>
<td>OC-48</td>
<td>STM-16</td>
</tr>
<tr>
<td>9953.28</td>
<td>OC-192</td>
<td>STM-64</td>
</tr>
</tbody>
</table>
**SDH (SONET) frame**

- **Framing ID** identifies the OC-1 number (1..N) in an OC-N frame
- **OA&M**
- **Reserved**
- **Multi**
- Payload Container Header

- **3 bytes**
- **87 bytes**

*ID field identifies the OC-1 number (1..N) in an OC-N frame
**Multi field indicates that payload spans multiple payload envelopes

---

**SDH**

- 9 rows, 90 columns
- Each payload container (SPE) served in 125 microseconds
- One byte = 1 call (1*8*(1000/125) kbit/s = 64kbit/s)
- 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

---

**Signaling System 7 (SS7)**

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

- Call forwarding
- To register
  - Call special number
  - Connects to ASE (Application Service Element)
  - Authenticated 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

**Internet stack**

<table>
<thead>
<tr>
<th>Data Plane</th>
<th>Control Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>HTTP</td>
</tr>
<tr>
<td>Session</td>
<td>Sockets/Streams</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>Network</td>
<td>IP</td>
</tr>
<tr>
<td>Datalink</td>
<td>Many</td>
</tr>
<tr>
<td>Physical</td>
<td>Many</td>
</tr>
</tbody>
</table>

**MTP (Message Transfer Part) Header**

```
<table>
<thead>
<tr>
<th>0</th>
<th>Flags</th>
<th>Back Seq #</th>
<th>Forward Seq #</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Backward indicator bit</td>
<td>Forward indicator bit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Service info</th>
<th>Destination Point Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC (cont.)</td>
<td>OrgPoint Code</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>DPC (cont.)</th>
<th>SLS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
```

**IP**

- Unreliable
- Best effort
- End-to-end
- IP on everything: interconnect the world
### IPv4

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>3 bits, identifies IPv4</td>
</tr>
<tr>
<td>Header Length</td>
<td>16 bits, length of header in bytes</td>
</tr>
<tr>
<td>Total Length (bytes)</td>
<td>20 bits, total length of packet in bytes</td>
</tr>
<tr>
<td>Identification</td>
<td>16 bits, identifies fragments</td>
</tr>
<tr>
<td>Flags</td>
<td>8 bits, fragmentation information</td>
</tr>
<tr>
<td>Fragment Offset (13 bits)</td>
<td>Fragment offset for reassembly at receiver</td>
</tr>
<tr>
<td>TTL</td>
<td>8 bits, time to live, decremented on each hop</td>
</tr>
<tr>
<td>Protocol</td>
<td>8 bits, identifies protocol</td>
</tr>
<tr>
<td>Source IPv4 address</td>
<td>32 bits, source IPv4 address</td>
</tr>
<tr>
<td>Destination IPv4 address</td>
<td>32 bits, destination IPv4 address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>40 bits, optional fields</td>
</tr>
</tbody>
</table>

### IP fields

- **TTL**
  - Decremented on each hop
  - Decremented every 500 ms at endpoint
  - 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

### IPv6

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4 bits, identifies IPv6</td>
</tr>
<tr>
<td>Traffic Class</td>
<td>20 bits, identifies traffic class</td>
</tr>
<tr>
<td>Flow Label</td>
<td>20 bits, identifies flow label</td>
</tr>
<tr>
<td>Payload Length</td>
<td>32 bits, length of payload in bytes</td>
</tr>
<tr>
<td>Source IPv6 address</td>
<td>128 bits, source IPv6 address</td>
</tr>
<tr>
<td>Destination IPv6 address</td>
<td>128 bits, destination IPv6 address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>40 bits, optional fields</td>
</tr>
</tbody>
</table>

### 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 packet with Don’t fragment flag set
  - If ICMP error, try smaller

• Lockup
• Error multiplication
### ICMP
- Destination unreachable
- Source quench
- Redirect
- Router advertisement
- Time exceeded (TTL)
- Fragmentation needed, but Don’t fragment flag set

### TCP
- Multiplexed
- Duplex
- Connection-oriented
- Reliable
- Flow-controlled
- Byte-stream
Fields

- Port numbers (16 bits each)
- Sequence and ACK number (32 bits each)
- Header length
- Window size
  - 16 bits → 64 Kbytes (more with scaling)
  - Receiver controls the window size
    - If 0: sender persistence timer (window change ACK)
  - *Silly window syndrome* – window reduces to tiny size
- Checksum
- Urgent pointer – seldom used (if not counting WinNuke)
- Options
  - Max segment size, window scale, …

HTTP

- Request/response – stateless connections
- Protocol is simple, browser is complex (and usually buggy)
- Address space encapsulation
- Request types
  - GET, HEAD, POST
  - PUT, DELETE
  - CONNECT, TRACE, OPTIONS
- Response
  - Status
  - Headers
  - Body

HTTP’s WWW success

- Representational State Transfer (REST) approach
- Initially transfer was stateless
  - Facilitates efficient caching
  - Transparent proxies
- Cookies can violate statelessness
  - Observe different behaviours of the browser “back” button
- In contrast to RPC
  - Resource names are more complex
    - E.g. URIs that include attributes
  - Fewer specific action types
    - E.g. mostly GET, POST – very different from RPC

ATM stack

<table>
<thead>
<tr>
<th></th>
<th>Data Plane</th>
<th>Control Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>UNI/PNNI</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>Q.2931</td>
<td></td>
</tr>
<tr>
<td>Session</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>SSCOP</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>AAL1-5</td>
<td>S-AAL (AAL5)</td>
</tr>
<tr>
<td>Data Link</td>
<td>ATM</td>
<td>ATM</td>
</tr>
<tr>
<td>Physical</td>
<td>Many</td>
<td>Many</td>
</tr>
</tbody>
</table>
ATM
- Connection-oriented
- In-sequence
- Unreliable
- Quality of service assured
- In User-Network Interface (UNI) cells there is a GFC field
  - Generic Flow Control
- In Network-Network Interface (NNI) cells the first byte is VPI data only

Virtual paths
- VP essentially the high order bits of VCI
  - All VCIs in a VP share path and resource reservation
- Saves table space in switches
  - Faster lookup
- Avoids signalling
- May waste resources
- Dynamic renegotiation of VP capacity may help
- Set of virtual paths defines a virtual private network

ATM Adaptation Layer (AAL)
- Was supposed to provide “rest of stack”
  - For a wide variety of different workloads
- 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)
  - (i.e. Common Part Convergence Sublayer-Protocol Data Unit – CPCS-PDU)
- Then fragment it and add ATM headers

![Diagram of AAL 3/4 frame structure]

AAL 3/4

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

```
AAL 3/4 Frame
+-------------------+-------------------+
| CPI               | Btag              |
|                   | Blen              |
|                   | Data              |
| Pad               | AL                |
|                  | Etag              |
| Length            |                   |
```

```
AAL 3/4 Cell
+-------------------+-------------------+-------------------+-------------------+
| Type              | Seq #             | MID               | Data              |
|                   |                   |                   |                   |
|                   |                   |                   |                   |
|                   |                   |                   |                   |
```

AAL 5

- Violates layering, but efficient (as a direct consequence)
- Bit in header marks end of frame

![Diagram of AAL 5 frame structure]

AAL5 frame format

```
AAL 5 EPDU
+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
| Data              | Pad               | UU                | CPI               | Length            | CRC               |
|                   |                   |                   |                   |                   |                   |
|                   |                   |                   |                   |                   |                   |
|                   |                   |                   |                   |                   |                   |
|                   |                   |                   |                   |                   |                   |
```

Service Specific Connection Oriented Protocol (SSCOP)
- Reliable transport for signalling 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 multi-protocol 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 principle
  - 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