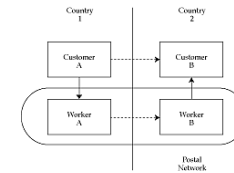


Protocol Layering

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

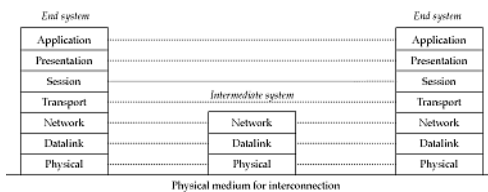
ISO OSI reference model

- 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 properties
- 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

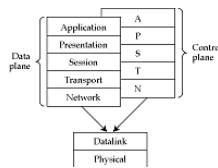
- Logically concatenates a set of links to form the abstraction of an *end-to-end* link
- Allows an end-system to communicate with any other end-system by computing a route between them
- Hides idiosyncrasies of datalink layer
- Provides unique network-wide addresses
- Found both in end-systems and in 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
 - ◆ responsible 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 rate 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 together
- 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