



Computers and Society

Privacy

Chris Brooks

Department of Computer Science

University of San Francisco



A Brief overview of TCP/IP

- Goal: Understand how a network transmits messages at different layers.
- How is a network composed?
- What really happens when Firefox opens a connection to a web server?
- Note: this will be an overview: For more details, you should take Prof. Buckwalter's class.

Layering

- Modern network design takes advantage of the idea of *layering*
- A particular service or module is constructed as a black box.
- Users of that service do not need to know its internals, just its interface.
- This makes it easy to later build new modules (or layers) that use the lower layers.
- For example, HTTP is built on top of TCP.
 - A web browser does not typically need to worry about the implementation of TCP, just that it works.
- Unlike modules in a typical OO system, the layers in a networked system comprise protocols that span multiple machines.

The OSI seven-layer model

- ISO (a standards body) developed a reference model called OSI that defines the different layers needed for communication, and specifies which should do each job.
- The goal is to produce an open protocol that allows for heterogeneous, extensible systems.
- A *protocol* is a specification describing the order and format of messages.
- An open protocol is one in which all of this information is publicly available.

The OSI seven-layer model

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

Message transmission across layers

- An application (such as a web browser) wants to send a message to another computer.
- That application constructs a message and passes it to the application layer.
- The application layer attaches a header to the message and passes it to the presentation layer.
- The presentation layer attaches a header and passes it to the session layer, and so on.

Message transmission across layers

- On the other end, the message is received by the physical layer, who strips off the appropriate header and passes the message up to the data link layer.
- This continues until the message reaches the application layer of the receiving machine.
- High-level layers don't need to worry about lower-level layers.
- Lower-level layers treat everything from higher layers as data to be sent.

Physical Layer

- This is the lowest-level layer, responsible for transmitting 0s and 1s.
- Governs transmission rates, full or half-duplex, etc.
- A modem works at the physical layer.
- Lots of interesting problems at this level that we won't get into ...

Data Link Layer

- The data link layer provides error handling for the physical layer.
- Individual bits are grouped together into frames.
- A checksum is then computed to detect transmission errors.
- The data link layer can then request a retransmission of an error is detected.
- Messages are numbered; receiver can request re-transmission of any message in a sequence.
- Each frame is a separate, distinct message.
- The data link layer provides error-free transmission to upper-level layers.

Network Layer

- The network layer is responsible for routing and flow control.
- The network layer removes the data link header and examines the resulting packet for a destination, and then forwards it as appropriate.
- The Internet Protocol (IP) is one of the best-known network-layer protocols.
- Primary role: move packets from a sending host to a receiving host. This involves:
 - Routing: determine the path that a packet should take to get to its destination.
 - Forwarding: When an incoming packet is received, place it on the output link that takes it to the next hop in its route.

IP Layer

- IP provides connectionless (datagram) delivery throughout the Internet.
- *Unreliable* delivery - no guarantees that a datagram will not get lost.
- No mechanism for flow control
- No Quality-of-service controls.
- IP addresses are used to determine where packets are routed.

Routing

- One of the major functions of IP is to move packets across the network.
- This process is called *routing*
- *Routers* don't contain the entire path from sources to destinations.
 - Instead, they contain the next hop for all destinations.
 - BGP is an exception to this.

Routing

- A router contains a *forwarding table* - when an incoming packet is received, the router compares it to this table to determine where to send it next.
 - This is forwarding.
- These forwarding tables are configured by means of a routing algorithm.
- For example, the link-state algorithm is a version of Dijkstra's algorithm - this computes a global routing table.
- Internet routing algorithms (such as BGP) are more complex, and use a decentralized routing table.
- In a nutshell, BGP lets subnets figure out how to reach other subnets via a gateway. That gateway is then responsible for routing within the subnet.

Transport Layer

- The network layer still operates at the level of individual packets, or datagrams.
- The transport layer provides a connection between the IP layer and a specific *port*
 - This allows applications to consume datagrams
- UDP is basically a transport-layer protocol.
 - Still unreliable and connectionless - packets may be dropped, or arrive out of order.
 - Useful for applications that don't want the overhead of setting up and tearing down a session.

Session Layer

- The session layer was designed to provide support for access rights and synchronization.
- TCP is the closest match in the TCP/IP suite to a session-layer protocol. (Some people call it a transport-level protocol)
- TCP provides *connection-oriented* service.
 - Guaranteed, in-order delivery.
 - State is maintained.
- This layer will also manage quality-of-service and congestion control.

Reliable Delivery

- TCP provides reliable in-order delivery on top of IP.
- Sender A sends a packet to receiver B.
- B returns an acknowledgment that the packet was received.
- If A does not receive an ACK before a timer expires, the packet is resent.

Improving Performance

- To improve transmission efficiency, TCP uses a concept called *sliding windows*.
 - The sender has a “window” of size n . It sends all packets within that window.
 - As the lowest-numbered packet in the window is acknowledged, the window “slides” upward, and more packets are sent.
- This improves transmission rates - the goal is for the network to be completely saturated.

Congestion

- The problem is how to deal with congestion.
 - Packets may be dropped by the receiver, or by intermediate hosts.
 - When should the sender resend?
 - Too slow → inefficiency
 - Too quickly → oversaturation is worsened.
- TCP uses an adaptive retransmission policy.
 - As connection performance changes, so does timeout duration.

Congestion

- The TCP congestion algorithm does the following (loosely):
 - When a packet is lost, halve the window size and double timeout.
 - If all packets in a window are transmitted successfully, increase window size by 1.
- There are lots of details in the implementation of this that I'm glossing over.
- The key point is this: This protocol works wonderfully, *as long as everyone else also uses it*.
 - Designed to minimize congestion over the entire Internet.

Congestion

- In the early days of the Internet, this was not a problem.
 - Small number of users, fewer bandwidth-saturating apps.
- Parallel download of images from web pages was the first concern.
- Later, non-TCP-based applications, such as video and VoIP implemented their own congestion control algorithms.
- These applications are not necessarily tuned to any sort of global optimum.

Tragedy of the Commons

- This is an example of a problem known as *tragedy of the commons*.
 - Cost of using a resource is not borne equally by the beneficiaries of that resource.
- Leads to overuse.
- Shared resources, such as networks, tend to be vulnerable to this problem.

Presentation Layer

- The presentation layer controls display of packet information.
- This may include encryption/decryption, compression, translation between character formats.
- SSL is a good example.

Application Layer

- This is the layer that most of us are most familiar.
- It consists of user-level protocols built on top of the existing layers.
 - HTTP
 - FTP
 - SMTP
 - P2P protocols
 - Instant messaging
 - RTSP/streaming video
 - etc.

An example: HTTP

- HTTP is the protocol that drives the Web.
 - A side note/axe to grind: WWW != Internet!!
- It is a stateless protocol that uses TCP as its underlying protocol.
 - The client sends a request, which is processed by the server.
 - The server sends a reply, and the exchange is ended.

HTTP requests

- HTTP has a very simple message format.

```
GET /~brooks/index.html HTTP/1.1
Host: www.cs.usfca.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language: en
```

- You can try this out for yourself with telnet ...

What TCP/IP provides

- Layered architecture - each layer provides additional functionality
- “end-to-end” delivery - each layer treats upper layers as opaque
- Routing - Network automatically handles damage and capacity
- Ability to build new applications on top of old ones.

What TCP/IP doesn't provide

- QoS at the IP layer
- Ability to prioritize packets of a particular type or source
- Ability to reserve bandwidth
- Ability to guarantee delay
- These are not necessarily problems for applications like telnet/ssh or email, but they are big problems for VoIP and video.

Summary

- the modern networking stack can be conceptually broken into a set of layers.
- Each layer has a specific, well-defined function.
 - Acts as a black box
- Higher-level layers build on the functionality of lower-level layers.
- How to move this model toward 21st century applications?