Why Network Software?

- Sending data through raw hardware is awkward and inconvenient.
- Network software is used to handle most low-level communication details.
- Application programs rely on network software to communicate.
Protocols

network or computer communication protocol - a set of rules specifying the format and meaning of messages exchanged among computers on a network.

protocol software - software that implements a protocol.

protocol suite - a set of related protocols that work together.
Protocol suite designers...

- analyze a communication problem;
- divide the problem into smaller problems;
- design a protocol for each subproblem.
A well designed protocol suite...

- is efficient and effective;

- allows replacement of individual protocols without changes to other protocols.
Layering

Layering model - a conceptual framework used to explain the purpose of and interaction among a set of protocols.

Examples:
1. The TCP/IP 4-layer model.
   Useful for Java network programming.
2. The ISO 7-layer reference model.
   Most modern protocols do not fit the ISO model. Primarily of historical interest.
3. We look at another model later.
Structure of Protocol Software

• Protocol software on each computer is divided into modules with each module corresponding to a layer.

• Software modules in one layer communicate only with software modules in adjacent layers.

(protocol) stack - an informal term for an implementation of a protocol suite.
On the sending computer, each layer...

- accepts an outgoing message from the layer above.
- adds a header and other processing.
- passes the resulting message to the layer below.
On the receiving computer, each layer...

- receives an incoming message from the layer below.
- removes the header for that layer and performs other processing.
- passes the resulting message to the layer above.
Layering Principle

Layer n software on the receiving computer must receive the exact same message that was sent by layer n on the sending computer.

• Thus, whatever operation is performed in layer n on the sending computer must be completely reversed in layer n on the receiving computer.
• Layering simplifies protocol design and testing. Sending and receiving software in each layer can be designed, implemented, and tested independently of the other layers.
Protocol Headers

• The software in a layer on the sending computer communicates with the corresponding layer on the receiving computer through information in a header.

• Each layer adds its header to the front of the message it receives from the layer above.

• Headers are nested at the front of the message as the message traverses the network.
Thus, ...

- If a layer on the sending computer prefixes a header to a frame, the corresponding layer on the receiving computer must remove that header.

- If a layer on the sending computer encrypts a frame, the corresponding layer on the receiving computer must decrypt the frame.
Out-of-Order Delivery

- Packets may be delivered out of order.
- Transport protocols detect and correct out-of-order delivery using sequencing.
  - The sending side attaches a sequence number to each packet.
  - The receiving side uses the sequence numbers to put the packets in order and to detect missing packets.
  - If a packet arrives in order, it is delivered to the above layer. If a packet arrives out of order, it is held until it is the next in order.
Duplicate Delivery

• It’s possible for packets to be duplicated during transmission.

• Sequencing can be used to detect duplicate packets.

• If the sequence number of a packet matches the sequence number of a packet already delivered, the packet is discarded.
Lost Packets

- One of the most widespread problems.
- Any error such as bit error or incorrect length causes the receiver to discard a packet.
- Protocols use positive acknowledgment with retransmission to detect and correct lost packets.
  - The receiver sends a short message acknowledging receipt of a packet.
  - The sender interprets a missing acknowledgment as a lost packet.
  - The sender retransmits a lost packet.
Lost Packets

- The sender sets a timer for each outgoing packet. If the timer expires before acknowledgment is received, the sender retransmits the packet.
- In case of a network failure, protocols specify an upper bound on the number of retransmissions.
Replay

- **Replay** is a condition in which the arrival of a delayed packet from an earlier communication is inserted into a later communication. Confusion results.

- Suppose two computers exchange data with packets numbered 1 to 5 and that packet 4 experiences a large delay. Protocol software on the sending computer retransmits packet 4.

- Now suppose a little later the two computers exchange data with packets numbered 1 to 10.
Replay

- Packet 4 from the earlier communication arrives during the second communication and is interpreted as the second communication’s packet 4.
- To prevent this, protocols mark the packets from a session with a session ID number (such as the time the session started).
Flow Control

- **Data overrun** occurs when a sender transmits data faster than a receiver can process it.
- **Flow control** mechanisms are used to control the data flow rate.
- **Stop-and-go** flow control.
  - The receiver sends a small control message when it is ready for the next packet.
  - The sender waits for the message before sending another packet.
  - Stop-and-go can result in very inefficient use of network bandwidth.
Flow Control

- **Sliding window** flow control.
  - The sender transmits several packets before receiving an acknowledgment.
  - The window size is the maximum amount of data that can be sent at a time.
  - As acknowledgments are received, the window “slides” along the data.

- Let L denote the **latency** or network delivery time. For stop-and-go, each packet requires 2L time for delivery, and so 4 packets require $4 \times 2L = 8L$ time. With a window size of 4, sliding window needs only 2L time.
Flow Control

- More generally, if $T_g$ is the stop-and-go throughput and $W$ is the window size, then the sliding window throughput $T_w$ is
  \[ T_w = W \cdot T_g \]

- Since the underlying network has a finite bandwidth $B$, the formula should be
  \[ T_w = \min(W \cdot T_g, B) \]
Network Congestion

- **Network congestion** is a condition in which packets experience excessive delay because the network is overrun with packets. Similar to traffic congestion.

- If congestion persists, a packet switch (a traffic light for packets) will run out of memory and begin discarding packets. Since the sender never receives acknowledgments, it retransmits lost packets. Ultimately, the network experiences **network collapse**.

- Protocols try to avoid congestion and recover from network collapse.
Network Congestion

- Two approaches are used:
  - Have packet switches inform senders when congestion occurs.
  - Use packet loss as an estimate of congestion.

- The second approach is valid because modern networks are reliable and rarely lose packets through hardware failure. Most packet losses result from congestion.

- Thus, missing acknowledgments can be interpreted by the sender as network congestion.
Protocol Design

• Protocol design combines engineering and art.
  - The techniques for solving specific problems are well known.
  - However, those techniques interact in subtle ways.
  - The resulting protocol suite must account for the interactions.

• Efficiency, effectiveness, and economy must be properly balanced.