Introduction

Motivations for networking

Information sharing
Enhanced availability through redundancy
Metcalf's law: Value of the network is ~ N^2 (the number of possible different connections where N is the number of nodes in the network (e.g., the early telephone system)

The hierarchy of networks

Networks *internal* to computer systems.

Local busses (Data flow, CPU to cache, High speed video etc.)
System bus (Processor to memory and I/O controllers)
I/O channels (I/O controllers to I/O devices, SCSI)

Networks *connecting* computer systems.

Local Networks
Long Haul Networks
Internetworks

Physical organization of computer networks

- *Hosts*

  The *end users* of the network.

- *Switching elements*

  Dedicated computers historically called

  Routers, Switches, IMPs, DSEs, COMCs, etc

  Responsible for routing of data flowing through the network

- *Transmission elements (called Channels or Links)*

  The physical transport medium interconnecting hosts and switching elements

Physical characteristics of transmission elements

- *Physical Media*

Wire	- low frequency electrical signaling
Space	- radio frequency electromagnetic waves
Optical	- visible light frequency electromagnetic waves

General rule: The higher the signaling frequency, the greater the bandwidth and the greater the bit rate.

- *Access Characteristics*

Point to point

Dedicated wired or optical connectionst between two switching elements

Shared medium (wireless, cable TV networks, early ethernet)

Multiple elements share a single channel or *broadcast domain*
All attached elements hear every transmission
Must have rules of access (Medium Access Control (MAC) protocols)
Original Ethernet is the canonical example

Hybrid

Physically point to point
Provides transparent broadcast capability
Examples include switched ethernets and ATM emulated LAN's

- *Topology*

Linear
Ring
Star
Mesh

Media characteristics that impact performance and protocol design

Bit rate: The inverse of the time it takes to send 1 bit of information

Latency: The time it takes a bit to travel from one network node to another (at approximately the speed of light) . Bit rate and latency *are not coupled.*

Error probability: The probability that a bit will be received in a state that is the inverse of the state in which it was sent (0 -> 1) or (1 -> 0)

High values of bit rate, latency and error rate all increase the difficulty of design and implementation of efficient protocols!

Historical development of networks and network architectures

Two major approaches (1960's and 70's)

Hierarchical networks -

Mainframe (MF) hierarchies developed by IBM

```
MF      ----   Controller   -   Device
 |      \                   -   Device
1980      \-   Controller   -   Device
 |                          -   Device
MF      ----   Controller   -   Device
                            -   Device
```

Communication links between mainframes was not supported until MSNF was introduced around 1980!.

Peer to Peer networks built around DEC minis

```
Mini  -  Mini  -  Mini  - Mini
```

One common set of problems was based upon the fact that the early network protocols were *built into the application* instead of the *OS kernel.*

Lack of standardization in protocols inhibits interoperability
Application specific protocols require links dedicated to specific apps.
Embedding network code in applications created applications that were:

- Expensive to develop
- Hard to maintain
- Excessively large and complex
- Non portable from one type of net (serial line) to another (LAN).

The solution

The solutions evolved in two phases

Phase I

Define *standard protocol (s).*
Divorce network code from apps, move it to the OS kernel and create *well defined nework API*

```
Applications
--- API ---
Network access code (in OS kernel)
```

The remaining problem

Network access code (now in the O/S kernel) was

Large, complex, and difficult to maintain
Not well suited for use with *heterogeneous components* (LAN's and long haul links).

Phase II of the solution (borrowed from OS developers of late 1960's)

Organize network software as a *hierarchy of separate layers*

```
Layer n                      Layer n
--------                   ----------
      :                        :
--------                   -----------
Layer 1  <--- peers --->     Layer 1
--------                   -----------
Layer 0          ---         Layer 0
```

Principles of layered design

Each layer knows only about the layer above and below it.
Layers should have well defined functions and interface(s)
Information flow across the interfaces should be minimized
Not too many and not too few layers.

Rules governing the exchange of information between *peer (not adjacent)* layers are called *protocols.*

The final touch: Network architecture definitions (borrowed from OO design)

The definition of *each layer* in the *ideal* protocol stack is further refined into three components:

Services	=	create a stack, push an element, pop and element
Interface	=	prototypes of public functions
Protocols	=	implementation of the private functions

Services:

The definition of what a layer does.
Not how it works (the protocol definition)
Not how to access it
The set of operations provided by a layer to the layer above it.

Interfaces:

Network API definition
How to access the layer from above or below

The need for formal *service* and *interface* definitions is in practice layer dependent.

Formal service (TCP shall provide a connection-orientened reliable byte stream service) and interface (the sockets API) are necessary for correct and portable applications.

Formal service and interface specification at the Link layer are required for device driver development.

For other layers in which components of one vendors protocol stack interoperate only with other internal layers of the same stack the need is not compelling.

Protocols:

Irrelevant and invisible to the user of the layer.

Sets of rules that defines the *syntax*, *semantics*, and *sequencing* of messages transmitted in a network.

Syntax - the locations and lengths and format of specific fields within a message or message header. (e.g.: The address is specified as a two byte binary number at offset 8 within the header).

Semantics - A value of 0x83 in the packet type field specifies that this is a standalone acknowledgment.

Sequencing - You will not send more than 7 packets before receiving an acknowledgment. You will not send packet class "A" while in state 19.

These rules govern the interchange of data and state information between *peer entities* in *different* nodes of the net.

Summary

A *network architecture (or protocol stack)* is the collection of service, interface and protocol definitions... one set for each layer in the network definition.

Clear separation of ideas of service, interface, and protocols has been claimed by some to be of critical importance in the design of reliable protocols.

The formal specifciation of service/interface/protocol definitions appeared primarily in ISO/OSI and ATM protocols --- *NOT in TCP/IP.*

The ISO/OSI network architecture

```
Application                                        Application
Presentation          Host Resident                Presentation
Session                                            Session
Transport                                          Transport
-------------------------------------------------------------
Network            Network          Network     Network
Datalink           Datalink         Datalink    Datalink
Physical---------Physical-------Physical---Physical

                   Communication Subnet
```

Headers and peer protocols:

As data travels down the stack each layer passes the data to be transmitted (called an nPDU) to the next lower layer.

Next lower layer may add a header (creating an (n-1)PDU).

As data travels back up the stack. Each layer removes the header created by its peer at the other end.

Layer N *never sees* Layer i headers for i < n.

Layer N treats Layer j headers as "user data" for j > n.

Interoperability issues

Much more *problematic in peer-protocol interactions* than in intra-stack interactions
The stack was created (and hopefully tested) by a single software vendor
Peer interactions may involve the products of unrelated entities (MS / Linux)

Protocol layer also affects interoperability

Physical layer and link layer interactions

Inherently *point-to-point* involving only two peers.
Interoperabilty can be tested when hardware is installed
NIC vendors are responsible for ensuring interoperability

Transport and higher interactions

Inherently *end-point to end-point*
But not as easy to verify as physical / link

Network layer interactions

Involve all routers through which the packet must flow.
By far the largest potential for trouble

==> KEEP THE NETWORK LAYER SIMPLE

IP success is an example of the importance of this principle
The lack of success of X.25 and switched ATM are too!!

Overview of *historical* layer functions:

Physical Layer:

How to physically encode 0's and 1's
How long each bit lasts.
Whether data can flow both ways simultaneously on same channel

Data Link:

Framing (identifying the start and end of a packet).
Addressing on shared medium channels.
Access management on shared medium channels
Error control (on channels with high error rates)
- Sequencing
- Error detection
- Acknowledging
- Retransmission

Network:

Routing
Congestion control
Billing --- (only possible in connection oriented network layers)

Transport:

Breaking logical messages into packets of constrained size
Error control if required by the application
Routing of packets to proper application (incoming messages for different apps carry the *same network address* but *different transport addresses)*.

Layers not present in TCP/IP

Session:

Establish sessions between two applications.
Checkpoint / recovery.
Data flow control (Token management).

Presentation:

Code transformations: ASCII - EBCDIC
Encryption
Compression

Application:
E-Mail
Web
File transfer
Remote session

Service characteristics (applicable to services at *all* layers)

Connection orientation

Connection oriented

The telephone system model
Requires a connection protocol before data exchange
May provide
Message stream (preserves message boundaries)
Byte stream (does not guarantee to preserve message boundaries)
Adv: Easier to provide QoS guarantees and to bill for service

Connection less

The postal system model
Each packet transmitted as a standalone entity
Also known as *datagram service*
Adv: Simple

Reliability

Unreliable
Loses, damages, duplicates, and/or reorders packets
Reliable
Doesn't do any of the above.

Different layers *within a single network archtecture* may provide different service characteristics

HDLC	(link layer)	- Reliable connection oriented message stream
IP	(network layer)	- Unreliable connectionless
TCP	(transport layer)	- Reliable connnection oriented byte stream
ATM	(cell transport)	- Unreliable connection oriented message (cell) stream

An idealized operational perspective

View each layer as consisting of two threads

The *up* thread is in charge of moving packets up the stack
The *down* thread moves them down

Two message queues exist *between* each layer

The *up* queue holds packets moving up the stack
The *down* queue holds packets moving down

Both threads act as consumer/producer processes

The *up* thread consumes from the *up* queue below
It performs necessary processing on the packet
- Header removal
- Error checking

It then produces onto the *up* queue above

The down thread works analogously

While this approach leads to a clean and robust design, it suffers from *prohibitive overhead.*

Using standard sized frames on 1 Gbps Ethernet packet transmission time is about 12 micro seconds!!

The layered structure of the implementation may refine that of the protocol

The Linux implementation breaks the *transport* and *link* layers into identifiable subcomponents

socket	A thin layer with a single implementation instance
protocol_family	A functional layer with one instance per protocol family (e..g PF_INET, PF_ATMPVC, PF_X25, etc.) It contains the generic components of the transport layer
transport_protocol	For each protocol family there is one instance of the this layer for each transport protocol. It contains the esoteric components of the particular transport (e.g., UDP, TCP, etc.). The CPSC 853 project will be to build one of these.
network_layer	One instance of this for each protocol family. For PF_INET this component contains the implementation of IP.
generic device(dev)	There is a single instance of the generic device layer that service all protocol families. It contains the NIC independent aspects of the MAC layer protocol, output packet scheduling, and demultiplexing of input packets with deliver to the proper network layer.
device class	There is one instance of this layer for each device class (e.g., ethernet, ATM, token ring, etc.) It contains components that are common to the MAC protocol but independent of the requirements of any specific NIC.
device driver	There is one instance of this layer for each specific NIC or family of NICs (e.g. 3C59x, e100, e1000, sk98lin)

Key:

transport layer
network layer
link layer

Standards

Apocalypse of the two elephants... standards must not be developed
Too early (bad standards)
Too late (defacto standards (M$ Window$) emerge)

Standards working groups

```
ITU (UN)          ISO                          IEEE              ISOC
   |               |                                          /       \
CCITT             ANSI -- BSI -- DIN                      IETF        IRTF
  |
National PTTs
```

Example Networks

ISO - OSI
- Reasons for lack of success
  - Bad timing ... see the apocalypse
    - Bad technology
    - Too telephone oriented
    - Too "designed by a committee"
    - Too complicated
    - Too inefficient
  - Bad politics
    - Too closely associated with European PTT's
  - Bad implementations
    - Too expensive
    - Too buggy
    - Too slow
    - And TCP/IP was FREE

TCP/IP

Application - telnet, ftp, smtp, http, ssl
Transport - TCP or UDP
Network - IP
Datalink/physical (many)

IBM - SNA

Presentation (NAU services)
Session (NAU services / data flow control)
Transport(Transmission Control)
Network (Path Control)
Datalink (SDLC + various LAN)