CS 455: Introduction To Distributed Systems [Networking]

Shrideep Pallickara
Computer Science
Colorado State University

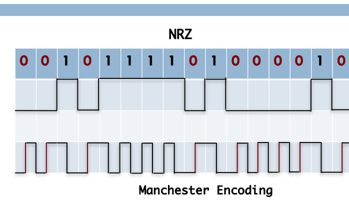
Frequently asked questions from the previous class survey

- Why not spawn processes per connection server-side?
- Encoding schemes: Why? Baseline wander: Analogy? Manchester encoding: why does it transition so much?
- ServerSocket and Socket communications
- Can we use ServerSockets and Sockets for HW1? How can we program on multiple computers?
- 3rd argument (multi-homed hosts) for ServerSocket
- What if the queue length for incoming connections is breached? Connection refused
- Is there a limit on the number of socket connections we can open on a computer? Number or range that gives best performance?
- Laptops and Labs

Topics covered in this lecture

- Encoding Wrap-up
- Bandwidth and Latency
- Multiplexing
- Network Architecture
- Encapsulation

Manchester encoding and NRZ



NRZI (Non return to zero inverted)

- Make a transition from current signal to encode a 1
  - **Stay** at current signal to encode a 0
- Solves the problem of consecutive 1's
  - But does nothing for consecutive 0's

4B/5B encoding

- Attempts to address inefficiencies in Manchester encoding
  - Without suffering from problems due to extended high/low signals
- The crux here is to insert **extra** bits into bitstream
  - Breakup long sequences of 1s or 0s
  - 4 bits of actual data encoded in a 5-bit code
  - 5-bit codes are carefully selected
    - No more than 1 leading 0 & no more than 2 trailing 0s

4B/5B encoding

4B	5B
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

4B/5B: Rules for the conversion of 4-bit codes to 5-bit codes

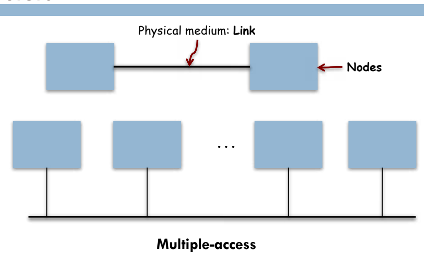
- Objective is to ensure that in each translation there is:
  - No more than one leading 0
  - No more than two trailing 0's
  - When sent back-to-back
    - No pair of 5-bit codes results in more than 3 consecutive 0's being transmitted
- 5-bit codes are transmitted using NRZI
  - This is why they are so concerned with consecutive 0's

NETWORKING

Expectations that we have of a network

- Application **programmer**
  - Error-free and timely delivery of messages
- Network **designer**
  - Cost effective design
  - Effective and fair allocation of resources
- Network **provider**
  - Easy to administer and manage
  - Isolate faults and account for usage

A network must provide connectivity among a set of computers



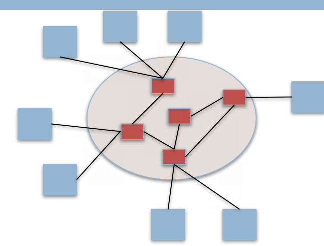
Multiple access links are limited in size

- Geographical **distances** that can be covered
- **Number** of nodes that can be connected

Connectivity between nodes need not imply a direct physical connection. Otherwise …

- Networks would be very **limited** in the number of nodes they could connect
- Number of wires out the back of a node
  - Unmanageable
  - Very expensive

Switched networks: Indirect connectivity among cooperating nodes



Switched networks: Indirect connectivity among cooperating nodes

- Nodes with *at least* two links
  - Run software that forwards data on one link out on another
- Types
  - Circuit switched
  - Packet switched

Switched networks: Circuit switched networks

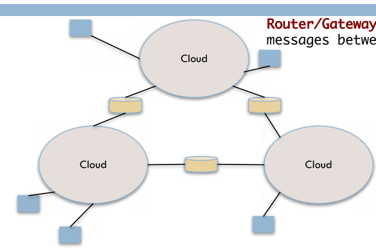
- Establish a **dedicated** circuit
  - Across a set of links
  - No one else can use this till termination
- Allows source to send a stream of bits
  - **Across circuit** to the destination node
- Employed by the telephone system

Switched networks: Packet switched networks

- Nodes in the network send **discrete** data blocks to each other
- Use **store-and-forward**
  1. Receive complete packet over some link
  2. Store packet in internal memory
  3. Forward complete packet to another node
- Used by the **overwhelming majority** of computer networks

Interconnection of networks

`Router/Gateway` forwards messages between networks



Addressing: A node must be able to say which nodes it wishes to communicate with

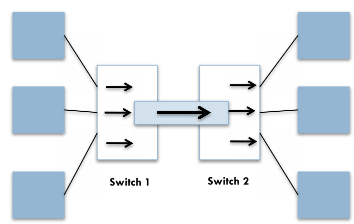
- Assign an address (byte string) to each node
  - Distinguish node from other nodes in the network
- Source specifies address of the destination node
- Switches and routers use address to forward messages *towards* the destination node
  - **Routing**

Cost effective resource sharing

How do all hosts that want to communicate share the network …

- At the same time?
- How about **sharing** links?
  - Hosts want to use it at the same time
- **Multiplexing** …
  - Analogy: Sharing CPU among multiple processes

Data sent by multiple users can be multiplexed over the physical links



Multiplexing data onto a physical link

- Synchronous time division multiplexing (STDM)
  - Divide time into quanta
  - Assign quanta in round-robin fashion
- Frequency division multiplexing (FDM)
  - Transit data *flows* at different frequencies

Problems with STDM and FDM

- {Problem-1} **Limited** to specific situations
  - Max number of flows is *fixed*
  - Known *ahead* of time
- {Problem-2} If one of the flows does not have data?
  - Its share of the physical link remains **idle**
- In computer communications:
  1. Amount of time a link is idle can be very large
  2. Data flows are fluid

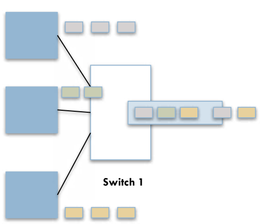
Statistical multiplexing

- Physical link is shared over time
- Data is transmitted from each flow **on demand**
  - Not a predetermined slot
  - When there is only one flow
    - *No need to wait* for quantum to come around

Limiting transmissions so that other flows can have a turn

- Upper bound on **size** of data block that each flow is allowed to transmit
  - **Packet**
- Larger application messages
  - Fragmented into several packets
  - Receiver reassembles these
- Each flow sends packets over the link
  - Decision made on a **packet-by-packet basis**

Multiplexing packets from multiple sources onto a shared link



Deciding which packet to send over a shared link

- In some cases, decision is made by switches
- Service packets using
  - FIFO
  - Round robin
    - Ensure flows receive a certain **share** of the bandwidth
    - Maximum **threshold** for delays for certain packets
- Networks that allow special treatment of flows
  - Quality of Service

Bandwidth and Latency

Network performance is measured in two fundamental ways

- **Bandwidth**
  - Number of bits transmitted over the network in a given time (e.g., 10 **m**illion **b**its **p**er **s**econds 10 Mbps)
  - Also called *throughput*
- **Latency**
  - How long it takes for message to go from one end of the network to another?

Components of latency

- Speed-of-light **propagation** delay
  - 3×10^8 m/sec in vacuum
  - 2.3×10^8 m/sec in cable
- Amount of time to **transmit** a unit of data
- **Queuing** delays

The Delay x Bandwidth product

Viewing the Network as a pipe

The Delay x Bandwidth product

- The product gives us information about *how many bits* **fit** in the pipe
- Transcontinental channel
  - 50 ms one-way latency
  - Bandwidth: 45 Mbps
  - Can hold: 50×10^{-3} seconds $\times 45 \times 10^6$ bits/second
    - 2.25×10^6 bits = 280 KB

The Delay x Bandwidth product

- Corresponds to how many bits the sender must send
  - Before first bit arrives at the receiver
- Bits in the pipe are said to be **in flight**

Bandwidth and latency improvements are not in lockstep

- Over past 35-40 years approximately
  - Bandwidth improvements: 220-1200 times
  - Latency improvements: 4-20 times
- Ethernet 802.3 (1978)
  - 10 Mbps
  - Latency 3 millisecond
- Ethernet 802.3ae (2003)
  - 10,000 Mbps (1000 times)
  - Latency 0.19 millisecond (15 times)

What does not change as the bandwidth increases?

- Speed of light
- High-speed **does not** mean that latency improves at the same rate as bandwidth
  - Transcontinental RTTs 100 ms for
    - 1-Mbps/1-Gbps link

Sending 1 MB data over a cross country link. Delay 100 ms

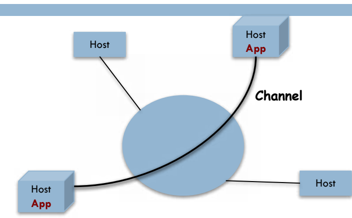
- 1 Mbps link
  - Pipe: $100 \times 10^{-3} \times 10^{6}$ = 100 Kb = 0.1 Mb
  - So you need 80 pipes to transmit 1 MB
    - 8 Mb/0.1 Mb = 80
- 1 Gbps
  - Pipe: $100 \times 10^{-3} \times 10^{9}$ = 100 Mb
    - So you need 8 Mb/100 Mb = approx 1/12th of the pipe is utilized

Support for common services

More accurate to think of network as allowing applications to communicate

- When 2 applications need to communicate
  - Lot of things need to happen
  - Beyond just sending messages between the hosts
- Build all functionality into each app?
- Identify and build right set of **common services**
  - Hide complexity without constraining functionality

Processes communicating over an abstract channel



Guarantees provisioned in the channel

- Guaranteed delivery?
- Ordered delivery?
- Thwart eavesdropping?

Not just *which* functionality, but *where* they will be provided

- View network as a **bit pipe**
  - High-level communication semantics provided by end hosts
  - Keeps switches in the middle *very simple*
- Alternative: Push functionality **onto** switches
  - End hosts are dumb devices
    - Telephones

Mask failures so that the network appears more reliable than it really is

- Bit errors
- Burst errors: Consecutive bits are corrupted
- Packet failures
  - Discarded because the switch buffer is full
    - **Congested**
  - Routing mistakes
- Node and link failures
  - Route around failed nodes and links

Network Architecture

All communications in distributed systems based on sending/receiving messages

- **No shared memory**
- Sending message from **A** to **B**
  - *Build* message in **A**'s address space
  - *Send* message over the network
  - *Reconstruct* message at **B**

But A and B must agree on the *meaning* of the bits

- Signaling **1**'s and **0**'s
- What is the *last bit* of the message?
- Detect if the message is lost or damaged
  - Respond to problems
- Representation of data types

Layering and Protocols

- Start with services provided by hardware
- Add a **sequence** of layers
  - Each providing higher level of service
- Services at higher layers implemented **in terms of** lower layers

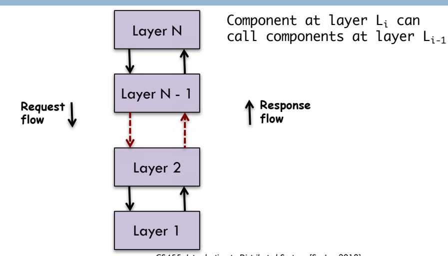
Advantages of layering

- Decomposes problem into **manageable** components
- Provides **modular** design
  - Adding functionality may result only in minor modifications

Advantages of layering

- Each layer can be changed **independently** of the other
  - Change as technology improves

Layered Architectures: Requests go down the hierarchy; results flow upward



Example of a layered network system

Application programs
Process-to-process channels
Host-to-Host connectivity
Hardware

Layered system with alternative abstractions at a given layer

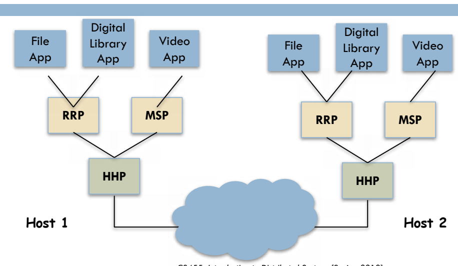
Application programs	
Request/Reply channels	**Message stream channel**
Host-to-Host connectivity	
Hardware	

Abstract objects that comprise layers of a network system are called **protocols**

- Provides a **service interface** to other objects on the same computer
  - Wishing to use its communication services
- Defines the **form** and **meaning of messages** exchanged by protocol peers
- Protocol also refers to modules that implement a specification

ENCAPSULATION

Example of a protocol graph



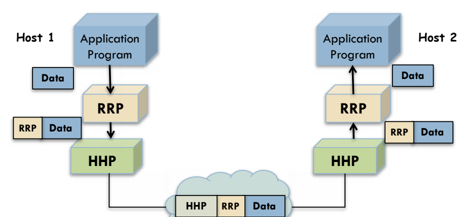
Encapsulation

- RRP receives a set of bytes to transmit from the application
  - E-mail, integers, images etc
- RRP is responsible for sending this data to its peer at the other end
  - Must communicate **control info** to its peer
  - Instruct how to **handle** the message

When asked to transmit info, lower level layers add information to the message

- Attach a **header** to the message
  - Small data structure
  - Few bytes to several dozen bytes
- Control info at the end of message: **trailer**
- Format is specific to the protocol
- Data being transmitted: body or **payload**
- Application data is said to be **encapsulated**

Encapsulating high-level messages inside low-level messages



Encapsulation: Some more info

- Low-level protocol does not **interpret** message given to it by high-level protocol
  - *Cannot extract* meaning
- Low-level protocol may apply simple **transformations** to the data it is given
  - Compress
  - Encrypt

Multiplexing is applicable up-and-down the protocol graph too

- RRP attaches header to every message that goes through it
  - Header include information to indentify the application
    - Called demultiplexing key or **demux key**
- At the destination host, RRP strips its header
  - Examines demux key
  - Demultiplexes message to correct application

Demux key is used at all levels of the protocol stack

- Some use an 8-bit field {**TCP** (6), **UDP** (17)}
  - Can support only 2^8 (256) high level protocols
  - Can also be 16/32-bits
- There could be a **single** demultiplexing field
  - Same demux key used at both ends
- There could be a **pair** of demultiplexing fields
  - Each side uses different key to identify high-level protocol

The contents of this slide-set are based on the following references

- *Computer Networks: A Systems Approach. Larry Peterson and Bruce Davie. 4th edition. Morgan Kaufmann. ISBN: 978-0-12-370548-8.* [Chapters 1 and 2]
- *Distributed Systems: Principles and Paradigms. Andrew S. Tanenbaum and Maarten Van Steen. 2nd Edition. Prentice Hall. ISBN: 0132392275/978-0132392273.* [Chapter 4]