Reliable Transport

1. Correctness condition
   if-and-only if again

2. Design space
   timeliness vs efficiency vs …

3. Examples
   Go-Back-N & Selective Repeat

A reliable transport design is correct if...

A packet is always resent if the previous packet was lost or corrupted
A packet may be resent at other times

Correct!

To improve timeliness, reliable transport protocols send multiple packets at the same time

approach

add sequence number inside each packet

add buffers to the sender and receiver

sender store packets sent & not acknowledged
receiver store out-of-sequence packets received
Bob

Alice

4 packets sent w/o ACKs

packet 1
packet 2
packet 3
packet 4

4 packets sent w/o ACKs

Sending multiple packets improves timeliness, but it can also overwhelm the receiver

supercomputer

overwhelmed smartphone

packet 1
packet 2
 ...
packet 1000

sends 1000 packets/s

can process 10 packets/s

Using a sliding window enables flow control

Sender keeps a list of the sequence # it can send known as the sending window

Receiver also keeps a list of the acceptable sequence # known as the receiving window

Sender and receiver negotiate the window size sending window <= receiving window

Example with a window composed of 4 packets

0 1 2 3 4 5 6 7

unACK'ed packets

forbidden packets

1 2 

ACKed packets

available packets

The efficiency of our protocol essentially depends on two factors

receiver feedback

How much information does the sender get?

behavior upon losses

How does the sender detect and react to losses?

What about fairness?

Design a correct, timely, efficient and fair transport mechanism knowing that

packets can get lost
corrupted reordered
delayed duplicated

Seeking an exact notion of fairness is not productive. What matters is to avoid starvation.

equal per flow is good enough for this

When n entities are using our transport mechanism, we want a fair allocation of the available bandwidth

Communication Networks | Mon 20 March 2017
Intuitively, we want to give users with "small" demands what they want, and evenly distributes the rest.

Max-min fair allocation is such that:
- the lowest demand is maximized after the lowest demand has been satisfied,
- the second lowest demand is maximized after the second lowest demand has been satisfied,
- the third lowest demand is maximized and so on…

Reliable Transport

Correctness condition: if and only if again

Design space: timeliness vs efficiency vs ...

Examples: Go-Back-N & Selective Repeat

This week on Communication Networks

We’ll do that layer-by-layer, bottom-up, starting with the Link layer

Communication Networks
Part 2: The Link Layer

How do local computers communicate?

http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

This week we start speaking about How the Internet actually works

HTTP(S) TCP/UDP IP Ethernet

Application Transport Network Link

eth0 eth1

How do local computers communicate?

#1 What is a link?
#2 How do we identify link adapters?
#3 How do we share a network medium?
#4 What is Ethernet?
#5 How do we interconnect segments at the link layer?
### Communication Networks
Part 2: The Link Layer

#### #1

- What is a link?
- How do we identify link adapters?
- How do we share a network medium?
- What is Ethernet?
- How do we interconnect segments at the link layer?

#### Network adapters communicate together through the medium

- Sending node
- Link layer protocol
- Receiving node

#### The Link Layer provides a best-effort delivery service to the Network layer

- L3: Network
  - Global best-effort delivery
- L2: Link
  - Local best-effort delivery
- L1: Physical
  - Physical transfer of bits

#### The Link Layer provides a best-effort delivery service to the Network layer, composed of 5 sub-services

- Encoding: represents the 0s and the 1s
- Framing: encapsulates a packet into a frame
- Error detection: detects errors with checksum
- Error correction: optionally corrects errors
- Flow control: pace sending and receiving node

---

**Medium Access Control addresses**
MAC addresses... are uniquely assigned hard-coded into the adapter when built identify the sender & receiver adapters used within a link are uniquely assigned hard-coded into the adapter when built use a flat space of 48 bits allocated hierarchically

MAC addresses are hierarchically allocated

34:36:3b:d2:8a:86

The first 24 bits blocks are assigned to network adapter vendor by the IEEE

34:36:3b:d2:8a:86
Apple, Inc.
1 Infinite Loop
Cupertino CA 95014
US
see http://standards-oui.ieee.org/oui/oui.txt

The second 24 bits block is assigned by the vendor to each network adapter

34:36:3b:d2:8a:86
assigned by Apple to my adapter

The address with all bits set to 1 identifies the broadcast address

ff:ff:ff:ff:ff:ff
enables to send a frame to all adapters on the link

By default, adapters only decapsulates frames addressed to the local MAC or the broadcast address

The promiscuous mode enables to decapsulate everything, independently of the destination MAC
Why don’t we simply use IP addresses?

Links can support any protocol (not just IP)
different addresses on different kind of links

Adapters may move to different locations
cannot assign static IP address, it has to change

Adapters must be identified during bootstrap
need to talk to an adapter to give it an IP address

You need to solve two problems
when you bootstrap an adapter

Who am I?
MAC-to-IP binding

Who are you?
IP-to-MAC binding

How do I acquire an IP address?
Given an IP address reachable on a link,
How do I find out what MAC to use?

You need to solve two problems
when you bootstrap an adapter

Who am I?
MAC-to-IP binding

Who are you?
IP-to-MAC binding

How do I acquire an IP address?
Dynamic Host Configuration Protocol

How do I find out what MAC to use?
Address Resolution Protocol

Network adapters traditionally acquire an IP address
using the Dynamic Host Configuration Protocol (DHCP)

Every connected device needs an IP address...

Host sends an "IP request" to everyone on the link
using the broadcast address
The Address Resolution Protocol (ARP) enables a host to discover the MAC associated to an IP address. To send an IP packet to 192.168.1.10, one must use the MAC address learned from the ARP request.
Communication Networks
Part 2: The Link Layer

What is a link?
How do we identify link adapters?
How do we share a network medium?
What is Ethernet?
How do we interconnect segments at the link layer?

Some medium are multi-access:
>1 host can communicate at the same time

Problem
distributed algorithm for sharing the channel
Solution
When can each node transmit?

Essentially, there are three techniques
to deal with Multiple Access Control (MAC)

Divide the channel into pieces
either in time or in frequency
Take turns
pass a token for the right to transmit
Random access
allow collisions, detect them and then recover

Now, it’s your turn
…to design a Random Access Protocol
instructions given in class
Ethernet offers an unreliable, connectionless service

- Unreliable: Receiving adapter does not acknowledge anything.
- Packets passed to the network layer can have gaps which can be filled by the transport protocol (TCP).

“Traditional” Ethernet relies on CSMA/CD

For this reason, Ethernet imposes a minimum packet size (512 bits)

\[
\text{Network length} = \frac{\min\_frame\_size \times \text{speed of light}}{2 \times \text{bandwidth}}
\]

Effect:
- B would detect a collision and send a jamming signal.
- A can detect the collision only after \( t + 2d \).

Modern Ethernet links interconnects exactly two hosts, in full-duplex, rendering collisions impossible!

- CSMA/CD is only needed for half-duplex communications.
- 10 Gbps Ethernet does not even allow half-duplex anymore.
- This means the 64 bytes restriction is not strictly needed but IEEE chose to keep it.
- Multiple Access Protocols are still important for Wireless.

The Ethernet header is simple, composed of 6 fields only

- preamble
- destination address
- source address
- type
- data
- CRC

-used for synchronization
- usually, IPv4 (0x0800)
- Cyclic Redundant Check
Communication Networks
Part 2: The Link Layer

What is a link?
How do we identify link adapters?
How do we share a network medium?
What is Ethernet?
How do we interconnect segments at the link layer?

Historically, people connected Ethernet segments together at the physical level using Ethernet hubs

Hubs work by repeating bits from one port to all the other ones

Local Area Networks are now almost exclusively composed of Ethernet switches

Switches connect two or more LANs together at the Link layer, acting as L2 gateways

Unlike with hubs, switches enable each LAN segment to carry its own traffic

<table>
<thead>
<tr>
<th>8 bytes</th>
<th>12 bytes</th>
<th>2 bytes</th>
<th>46–1500 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>preamble</td>
<td>dest address</td>
<td>src address</td>
<td>data</td>
<td>CRC</td>
</tr>
</tbody>
</table>

Ethernet efficiency (payload/tot. frame size): ~97.5%
Maximum throughput for 100 Mbps: ~97.50 Mbps

advantages
simple, cheap

disadvantages
inefficient, each bit is sent everywhere
limits the aggregates throughput
limited to one LAN technology
can’t interconnect different rates/formats
limited number of nodes and distances
cannot go beyond 2500m on Ethernet

Switches are “store-and-forward” devices, they
- extract the destination MAC from the frame
- look up the MAC in a table (using exact match)
- forward the frame on the appropriate interface

Switches are similar to IP routers, except that they operate one layer below

unicast traffic between A and C is not seen by F
Unlike with hubs, switches support concurrent communication. A and F can talk to each other, while A and C are talking.

The advantages of switches are numerous:
- only forward frames where needed
- avoids unnecessary load on segments
- join segments using different technologies
- improved privacy
- host can just snoop traffic traversing their segment
- wider geographic span

Switches are plug-and-play devices, they build their forwarding table on their own.

Switches are "store-and-forward" devices, they:
- extract the destination MAC from the frame
- look up the MAC in a table (using exact match)
- forward the frame on the appropriate interface

When a frame arrives:
- inspect the source MAC address
- associate the address with the port
- store the mapping in the switch table
- launch a timer to eventually forget the mapping

In cases of misses, switches simply floods the frames.

When a frame arrives with an unknown destination:
- forward the frame out of all interfaces except for the one where the frame arrived

Hopefully, this is an unlikely event.

While flooding enables automatic discovery of hosts, it also creates problems when the networks has loops.

Each frame leads to the creation of at least two new frames! exponential increase, with no TTL to remove looping frames...

While loops create major problems, networks need redundancy for tolerating failures!

solution
Reduce the network to one logical spanning tree

Upon failure, automatically rebuild a spanning tree.
In practice, switches run a \textit{distributed} Spanning-Tree Protocol (STP)

A tree that must be sure to span
So packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.
Least-cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
Then bridges find a spanning tree.

For this switches exchange
\textbf{Bridge Protocol Data Unit (BDPU) messages}

Each switch \texttt{X}iteratively sends

\hspace{1cm} \texttt{BPDU (Y, d, X)}
\hspace{1cm} the switch ID
\hspace{1cm} if it considers as root
\hspace{1cm} the # hops to reach it

to each neighboring switch

Initially

Each switch proposes itself as root
\hspace{1cm} sends (X,0,X) on all its interfaces

Upon receiving (Y, d, X), checks if Y is a better root
\hspace{1cm} if so, considers Y as the new root, flood updated message

Switches compute their distance to the root, for each port
\hspace{1cm} simply add 1 to the distance received, if shorter, flood

Switches disable interfaces not on shortest-path

Algorhyme

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree that must be sure to span
So packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.
Least-cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
Then bridges find a spanning tree.

— Radia Perlman

Constructing a Spanning Tree in a nutshell

Switches...

elect a root switch
\hspace{1cm} the one with the smallest identifier

determine if each interface is
\hspace{1cm} on the shortest-path from the root
and disable it if not

Apply the algorithm starting with switch 4

\begin{itemize}
\item For this switches exchange
\item Bridge Protocol Data Unit (BDPU) messages
\end{itemize}
Apply the algorithm starting with switch 4

To be robust, STP must react to failures

Any switch, link or port can fail including the root switch.

Root switch continuously sends messages announcing itself as the root (1,0,1); others forward it.

Failures is detected through timeout (soft state) if no word from root in $X$, times out and claims to be the root.