Communication Networks
Spring 2022

Laurent Vanbever
nsg.ee.ethz.ch

ETH Zürich (D-ITET)
14 March 2022

Materials inspired from Scott Shenker & Jennifer Rexford
Last week on
Communication Networks
Communication Networks
Part 2: Concepts

How do you guide IP packets from a source to destination?
## Forwarding vs Routing

### Summary

<table>
<thead>
<tr>
<th>Goal</th>
<th>Forwarding</th>
<th>Routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directing packet to</td>
<td>Directing packet to an outgoing link</td>
<td>Computing the paths packets will follow</td>
</tr>
<tr>
<td>an outgoing link</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scope</td>
<td>Local</td>
<td>Network-wide</td>
</tr>
<tr>
<td>Implement.</td>
<td>Hardware</td>
<td>Software</td>
</tr>
<tr>
<td></td>
<td>Usually</td>
<td>Usually</td>
</tr>
<tr>
<td>Timescale</td>
<td>Nanoseconds</td>
<td>Milliseconds</td>
</tr>
<tr>
<td></td>
<td>(hopefully)</td>
<td></td>
</tr>
</tbody>
</table>
Theorem

A global forwarding state is valid if and only if:

- There are no dead ends
  - No outgoing port defined in the table
- There are no loops
  - Packets going around the same set of nodes
Producing valid routing state is harder but doable.

- Prevent dead ends: easy
- Prevent loops: hard

This is the question you should focus on.
Essentially, there are three ways to compute valid routing state:

<table>
<thead>
<tr>
<th>Intuition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 Use tree-like topologies</td>
<td>Spanning-tree</td>
</tr>
<tr>
<td>#2 Rely on a global network view</td>
<td>Link-State</td>
</tr>
<tr>
<td>#3 Rely on distributed computation</td>
<td>Distance-Vector</td>
</tr>
</tbody>
</table>
This week on

Communication Networks
This week we'll start speaking about
How the Internet actually works
We’ll do that layer-by-layer, bottom-up, starting with the Link layer
How do local computers communicate?
Communication Networks
Part 2: The Link Layer

#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?
#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?
Link Communication medium and Network adapter

- Wifi
- Ethernet
- Fiber
Network adapters communicate together through the medium.

![Diagram showing communication between network adapters through packets and frames.](image-url)
Network adapters communicate together through the medium.

Sender:
- Encapsulate packets in a frame
- Add error checking bits, flow control, ...

Receiver:
- Look for errors, flow control, ...
- Extract packet and passes it to the network layer
The Link Layer provides a best-effort delivery service to the Network layer.
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** encapsulate packet into a frame, adding header and trailer
- **error detection** detects errors with checksum
- **error correction** optionally correct errors
- **flow control** pace sending and receiving node
As of March 2021, state-of-the-art Ethernet adapters clock at **200 Gbps**

215 million pkt/sec

sub 0.8 usec latency

PCIe Gen 4.0

source: [Mellanox ConnectX-6]
400 Gbps adapters are around the corner

source: [NVIDIA ConnectX-7 (InfiniBand)]
400 Gbps Ethernet switches are already on the market

Cisco Nexus 9364D-GX2A

64x400 GbE ports (QSFP-DD)

25.6 Tbps backplane capacity

source: [cisco]
Communication Networks

Part 2: The Link Layer

What is a link?

#2 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?
Medium Access Control addresses
MAC addresses...
MAC addresses...

identify the sender & receiver adapters used on 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

\texttt{ff:ff:ff:ff:ff:ff}

enables to send a frame to
\textit{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.
Adapters must be identified during bootstrap
need to talk to an adapter to give it an IP address
We need to solve two problems when we bootstrap an adapter

Who am I?  
MAC-to-IP binding

How do I acquire an IP address?

Who are you?  
IP-to-MAC binding

Given an IP address reachable on a link,  
How do I find out what MAC to use?
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who am I?</td>
<td>How do I acquire an IP address?</td>
</tr>
<tr>
<td>MAC-to-IP binding</td>
<td><strong>Dynamic Host Configuration Protocol</strong></td>
</tr>
<tr>
<td>Who are you?</td>
<td>Given an IP address reachable on a link, How do I find out what MAC to use?</td>
</tr>
<tr>
<td>IP-to-MAC binding</td>
<td><strong>Address Resolution Protocol</strong></td>
</tr>
</tbody>
</table>
Network adapters traditionally acquire an IP address using the Dynamic Host Configuration Protocol (DHCP)
Every connected device needs an IP address…

Newark Airport… source: http://i.imgur.com/m1SQA6W.jpg
34:36:3b:d2:8a:10
no ip :(

34:36:3b:d2:8a:89
192.168.1.1

34:36:3b:d2:8a:86
192.168.1.10
Host sends an “IP request” to everyone on the link using the broadcast address.

34:36:3b:d2:8a:10
no ip :(  
34:36:3b:d2:8a:89
192.168.1.1

34:36:3b:d2:8a:86
192.168.1.10
DHCP discovery

dstmac: ff:ff:ff:ff:ff:ff
payload: I want an IP

34:36:3b:d2:8a:10
no ip :(

34:36:3b:d2:8a:89
192.168.1.1

34:36:3b:d2:8a:86
192.168.1.10
DHCP server (if any) answers with an IP address

34:36:3b:d2:8a:10
no ip :(

34:36:3b:d2:8a:89
192.168.1.1

34:36:3b:d2:8a:86
192.168.1.10
DHCP offer

dstmac 34:36:3b:d2:8a:10
payload use 192.168.1.9

34:36:3b:d2:8a:10
no ip :(

34:36:3b:d2:8a:86
192.168.1.10

34:36:3b:d2:8a:89
192.168.1.1
The Address Resolution Protocol (ARP) enables a host to discover the MAC associated to an IP
I want to send an IP packet to 192.168.1.10? What destination MAC do I use?!
Who has 192.168.1.10? Tell 192.168.1.9

ARP request

dstmac ff:ff:ff:ff:ff:ff
payload Who has 192.168.1.10? Tell 192.168.1.9

34:36:3b:d2:8a:10
192.168.1.9

34:36:3b:d2:8a:10
192.168.1.9

34:36:3b:d2:8a:89
192.168.1.1

34:36:3b:d2:8a:86
192.168.1.10
ARP table

```
192.168.1.10  34:36:3b:d2:8a:86
...           ...
```

```
34:36:3b:d2:8a:10  192.168.1.9
```

```
34:36:3b:d2:8a:89  192.168.1.1
```

```
34:36:3b:d2:8a:86  192.168.1.10
```
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
Some medium are **multi-access:**
>1 host can communicate at the same time

- Wireless networks
- Satellite networks
- Original Ethernet networks
- Cellular networks
Some medium are **multi-access:**
> 1 host can communicate at the same time

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>collisions lead to garbled data</td>
<td>distributed algorithm for sharing the channel</td>
</tr>
</tbody>
</table>

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
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?
Ethernet...

*was* a broadcast technology
each packet was received by all attached hosts

is *the* dominant wired LAN technology
by far the most widely used

kept up with the speed race
from 10 Mbps to 400 Gbps (next: 800 Gbps and 1.6 Tbps!)
<table>
<thead>
<tr>
<th>unreliable</th>
<th>Receiving adapter does not acknowledge anything</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Packets passed to the network layer can have gaps</td>
</tr>
<tr>
<td></td>
<td>which can be filled by the transport protocol (TCP)</td>
</tr>
<tr>
<td>connectionless</td>
<td>No handshaking between the send and receive adapter</td>
</tr>
</tbody>
</table>
“Traditional” Ethernet relies on CSMA/CD
CSMA/CD imposes limits on the network length

Suppose $A$ sends a packet at time $t$

$B$ sees an idle line just before $t+d$ and sends a packet

Effect

$B$ would detect a collision and sends a jamming signal

$A$ can detect the collision only after $t+2d$
For this reason, Ethernet imposes a minimum packet size (512 bits)

This imposes restriction on the length of the network

\[
\text{Network length} = \frac{\text{min_frame_size} \times \text{speed of light}}{2 \times \text{bandwidth}} \quad \text{[m]}
\]

= 768 meters \quad \text{for 100 Mbps}

What about for 1 Gbps, 10 Gbps, 100 Gbps?
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 important concepts to know in practice.
The Ethernet header is simple, composed of 6 fields only:

- **Preamble**: used for synchronization
- **Destination address**
- **Source address**
- **Type**: usually, IPv4 (0x0800)
- **Data**
- **CRC**: Cyclic Redundant Check
<table>
<thead>
<tr>
<th>Ethernet efficiency (payload/tot. frame size):</th>
<th>~97.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum throughput for 100 Mbps:</td>
<td>~97.50 Mbps</td>
</tr>
</tbody>
</table>
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?

#5 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.
Hubs are now OBSOLETE

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
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.

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

unicast traffic between A and C is \textit{not} seen by F.
Unlike with hubs, switches supports concurrent communication

B and F can talk to each other, while A and C are talking
The advantages of switches are numerous

- advantages
  - only forward frames where needed
  - avoids unnecessary load on segments

- join segment using different technologies

- improved privacy
  - host can just snoop traffic traversing their segment

- wider-geographic span
  - separates segments allow longer distance
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
Switches are plug-and-play devices, they build their forwarding table on their own.

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

Switch learns how to reach A
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

*when in doubt, shout!*
While flooding enables automatic discovery of hosts, it also creates problems when the networks have loops.

Each frame leads to the creation of \textit{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 *distributed* Spanning-Tree Protocol (STP)
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.

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
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.

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.
Constructing a Spanning Tree in a nutshell

Switches...

elect a root switch
the one with the smallest identifier

determine if each interface is
on the shortest-path from the root
disable it if not
For this switches exchange
Bridge Protocol Data Unit (BDPU) messages

Each switch $X$ iteratively sends

$$\text{BPDU} \ (Y, d, X)$$

to each neighboring switch

the switch ID it considers as root

the # hops to reach it
Initially, each switch proposes itself as root. It sends \((X,0,X)\) on all its interfaces.

Upon receiving \((Y, d, X)\), if \(Y\) is a better root, the switch considers \(Y\) as the new root and floods the updated message.

Switches compute their distance to the root, for each port, by simply adding 1 to the distance received. If the shorter path is found, the switch floods the information.

Switches disable interfaces not on the shortest path.
Upon receiving non-neighbor BPDUs from non-neighbor switches with equal cost:
Pick the BPDU with the lower switch sender ID.

Upon receiving non-neighbor BPDUs from a neighboring switch:
Pick the BPDU with the lowest port ID (e.g. port 2 < port 3).
Apply the algorithm starting with switch 4
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
The Local Area Networks we have considered so far define single broadcast domains

If one user broadcast a frame, every other user receives it
As the network scales, network operators like to segment their LANs

Why?

Improves security
smaller attack surface (visibility & injection)

Improves performance
limit the overhead of broadcast traffic (e.g. ARP)

Improves logistics
separates traffic by role (e.g. staff, students, visitors)
Organizational changes are too frequent to segment networks purely *physically*—rewiring is a major pain.

What about doing this in software though?
Enters “Virtual Local Area Networks” (VLANs)

Definition

A VLAN logically identifies a set of ports attached to one (or more) Ethernet switches, forming one broadcast domain.
A VLAN identifies a set of ports attached to one or more Ethernet switches.
Switches need configuration tables telling them which VLANs are accessible via which interfaces
Switches need configuration tables telling them which VLANs are accessible via which interfaces.

VLAN Staff: port A, port C
VLAN Student: port B
Consider that A sends a broadcast frame
say, an ARP request

src_mac: A
dst_mac: ff:ff:ff:ff:ff:ff
payload: Who has 10.0.1.1?
That frame should be received by all staff members: i.e. C and F, and *only* them

```
src_mac: A
dst_mac: ff:ff:ff:ff:ff:ff
payload: Who has 10.0.1.1?
```
src_mac: A
dst_mac: ff:ff:ff:ff:ff:ff
payload: Who has 10.0.1.1?

How does this switch know where to flood it next?
e.g. pointless to send it right
How does this switch know where to flood it next?

Here, only to F

src_mac: A
dst_mac: ff:ff:ff:ff:ff:ff
payload: Who has 10.0.1.1?
To identify VLAN, switches add new header when forwarding traffic to another switch.

**Without VLAN**

<table>
<thead>
<tr>
<th>preamble</th>
<th>dest address</th>
<th>src address</th>
<th>type</th>
<th>CRC</th>
</tr>
</thead>
</table>

**With VLAN**

<table>
<thead>
<tr>
<th>preamble</th>
<th>dest address</th>
<th>src address</th>
<th>TPID</th>
<th>VID</th>
<th>type</th>
<th>data</th>
<th>CRC</th>
</tr>
</thead>
</table>

802.1q Header (4 bytes)
(4 bits missing)
With VLANs, Ethernet links are divided in two sets: access and trunks (inter switches) links.
Access links belong to one VLAN
they do not carry 802.1q headers
Trunk links carry traffic for more than one VLAN and as such carry 801.1q tagged frames.
Each switch runs one MAC learning algorithm for each VLAN

When a switch receives a frame with an unknown or a broadcast destination,
it forwards it over all the ports that belong to the same VLAN

When a switch learns a source address on a port
it associates it to the VLAN of this port and only uses it when forwarding frames on this VLAN
src_mac: A
dst_mac: ff:ff:ff:ff:ff:ff
VID: 
payload: Who has 10.0.1.1?
Switches can also compute per-VLAN spanning-tree allowing a distinct SPT for each VLAN. This allows the operators to use more of their links.
A diagram showing a network topology with seven switches labeled as follows:

- Switch 1
- Switch 2
- Switch 3
- Switch 4
- Switch 5
- Switch 6
- Switch 7
Root switch for all VLANs

deactivated link for all VLANs
Any communication between the red hosts on switch 5 and 6 need to go via switch 1…
Root switch for VLAN

Deactivated link for...
Now any communication between the red hosts on switch 5 and 6 go via the direct link.