# Operating Systems and Networks

# Network Lecture 4: Link Layer (2)

Adrian Perrig
Network Security Group
ETH Zürich

## Pending Issues

- How to read the course textbook?
- How to prepare for the exam given that there is a huge amount of material?

2

## Where we are in the Course

- Finishing off the Link Layer!
  - Builds on the physical layer to transfer frames over connected links

| |
|---|
| Application |
| Transport |
| Network |
| Link |
| Physical |

3

## Topics

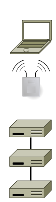
1. Framing
   - Delimiting start/end of frames
2. Error detection/correction
   - Handling errors

Done


4

## Topics (2)

3. Retransmissions
   - Handling loss
4. Multiple Access
   - Classic Ethernet, 802.11
5. Switching
   - Modern Ethernet



5

## Retransmissions (ARQ) (§3.3)

- Two strategies to handle errors:

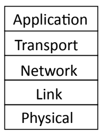
1. Detect errors and retransmit frame (Automatic Repeat reQuest, ARQ)
2. Correct errors with an error correcting code

Done this

6

## Context on Reliability

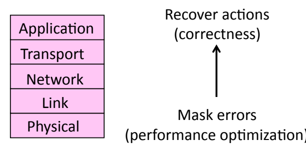
- Where in the stack should we place reliability functions?




7

## Context on Reliability (2)

- Everywhere! It is a key issue
  - Different layers contribute differently




8

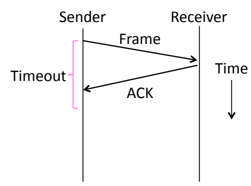
## ARQ (Automatic Repeat reQuest)

- ARQ often used when errors are common or must be corrected
  - E.g., WiFi, and TCP (later)
- Rules at sender and receiver:
  - Receiver automatically acknowledges correct frames with an ACK
  - Sender automatically resends after a timeout, until an ACK is received

9

## ARQ (2)

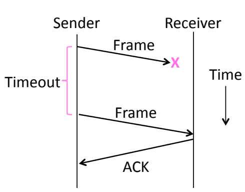
- Normal operation (no loss, no error)




10

## ARQ (3)

- Loss and retransmission




11

## So What's Tricky About ARQ?

- Two non-trivial issues:
  - How long to set the timeout?
  - How to avoid accepting duplicate frames as new frames
- Want performance in the common case and correctness always

12

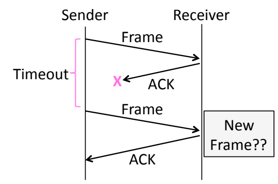
## Timeouts

- Timeout should be:
  - Not too big (link goes idle)
  - Not too small (spurious resend)
- Fairly easy on a LAN
  - Clear worst case, little variation
- Fairly difficult over the Internet
  - Much variation, no obvious bound
  - We'll revisit this with TCP (later)

13

## Duplicates

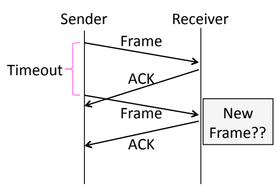
- What happens if an ACK is lost?




14

## Duplicates (2)

- Or the timeout is early?




15

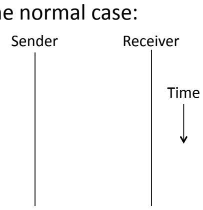
## Sequence Numbers

- Frames and ACKs must both carry sequence numbers for correctness
- To distinguish the current frame from the next one, a single bit (two numbers) is sufficient
  - Called Stop-and-Wait

16

## Stop-and-Wait

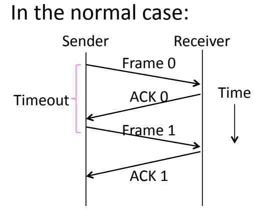
- In the normal case:




17

## Stop-and-Wait (2)

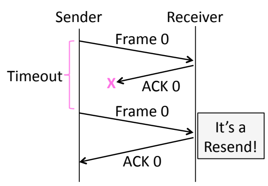
- In the normal case:




18

## Stop-and-Wait (3)

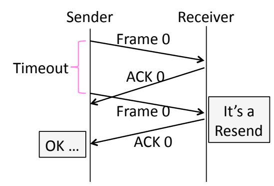
- With ACK loss:




19

## Stop-and-Wait (4)

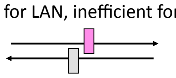
- With early timeout:




20

## Limitation of Stop-and-Wait

- It allows only a single frame to be outstanding from the sender:
  - Good for LAN, inefficient for high BD (bandwidth-delay product)

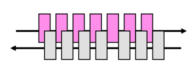


- Ex: R=1 Mbps, D = 50 ms
  - How many frames/sec? If R=10 Mbps?

21

## Sliding Window

- Generalization of stop-and-wait
  - Allows W frames to be outstanding
  - Can send W frames per RTT (=2D)



  - Various options for numbering frames/ACKs and handling loss
    - Will look at along with TCP (later)

22

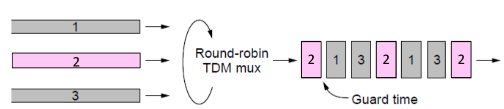
## Multiplexing (§2.5.3, 2.5.4)

- Multiplexing is the network word for the sharing of a resource
- Classic scenario is sharing a link among different users
  - Time Division Multiplexing (TDM)
  - Frequency Division Multiplexing (FDM)

23

## Time Division Multiplexing (TDM)

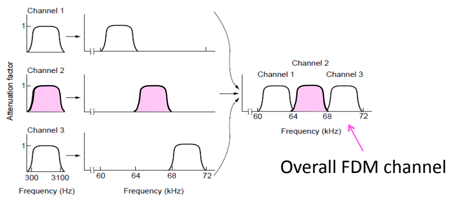
- Users take turns on a fixed schedule




24

## Frequency Division Multiplexing (FDM)

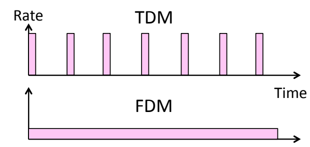
- Put different users on different frequency bands






## TDM versus FDM

- In TDM a user sends at a high rate a fraction of the time; in FDM, a user sends at a low rate all the time





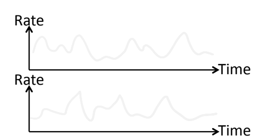

## TDM/FDM Usage

- Statically divide a resource
  - Suited for continuous traffic, fixed number of users
- Widely used in telecommunications
  - TV and radio stations (FDM)
  - GSM (2G cellular) allocates calls using TDM within FDM



## Multiplexing Network Traffic

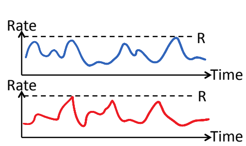
- Network traffic is <u>bursty</u>
  - ON/OFF sources
  - Load varies greatly over time






## Multiplexing Network Traffic (2)

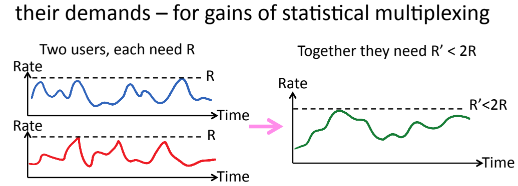
- Network traffic is <u>bursty</u>
  - Inefficient to always allocate user their ON needs with TDM/FDM






## Multiplexing Network Traffic (3)

- <u>Multiple access</u> schemes multiplex users according to their demands – for gains of statistical multiplexing



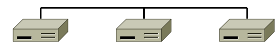
## Multiple Access

- We will look at two kinds of multiple access protocols
1. Randomized. Nodes randomize their resource access attempts
   - Good for low load situations
2. Contention-free. Nodes order their resource access attempts
   - Good for high load or guaranteed quality of service situations

31

## Randomized Multiple Access (§4.2.1-4.2.2, 4.3.1-4.3.3)

- How do nodes share a single link? Who sends when, e.g., in WiFI?
  - Explore with a simple model

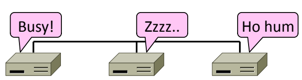


- Assume no-one is in charge; this is a distributed system

32

## Randomized Multiple Access (2)

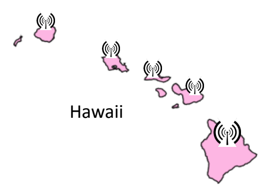
- We will explore random multiple access control or medium access control (MAC) protocols
  - This is the basis for classic Ethernet
  - Remember: data traffic is bursty




33

## ALOHA Network

- Seminal computer network connecting the Hawaiian islands in the late 1960s
  - When should nodes send?
  - A new protocol was devised by Norm Abramson …




34

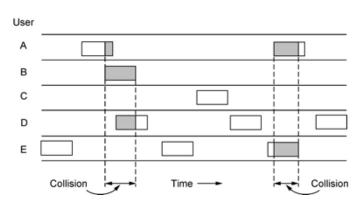
## ALOHA Protocol

- Simple idea:
  - Node just sends when it has traffic.
  - If there was a collision (no ACK received) then wait a random time and resend
- That's it!

35

## ALOHA Protocol (2)

- Some frames will be lost, but many may get through…
- Good idea?




36

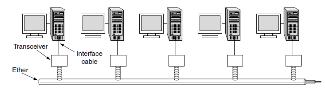
# ALOHA Protocol (3)

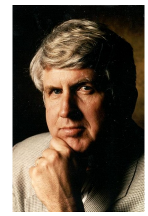
- Simple, decentralized protocol that works well under low load!
- Not efficient under high load
  - Analysis shows at most 18% efficiency
  - Improvement: divide time into slots and efficiency goes up to 36%
- We'll look at other improvements

37

# Classic Ethernet

- ALOHA inspired Bob Metcalfe to invent Ethernet for LANs in 1973
  - Nodes share 10 Mbps coaxial cable
  - Hugely popular in 1980s, 1990s



: © 2009 IEEE

38

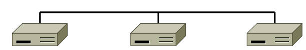
# CSMA (Carrier Sense Multiple Access)

- Improve ALOHA by listening for activity before we send (Doh!)
  - Can do easily with wires, not wireless
- So does this eliminate collisions?
  - Why or why not?

39

# CSMA (2)

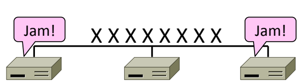
- Still possible to listen and hear nothing when another node is sending because of delay



40

# CSMA/CD (with Collision Detection)

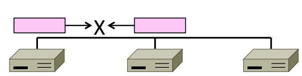
- Can reduce the cost of collisions by detecting them and aborting (Jam) the rest of the frame time
  - Again, we can do this with wires




41

# CSMA/CD Complications

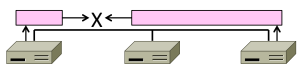
- Want everyone who collides to know that it happened
  - Time window in which a node may hear of a collision is 2D seconds




42

## CSMA/CD Complications (2)

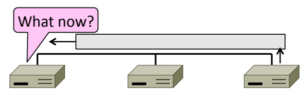
- Impose a minimum frame size that lasts for 2D seconds
  - So node can't finish before collision
  - Ethernet minimum frame is 64 bytes




43

## CSMA "Persistence"

- What should a node do if another node is sending?

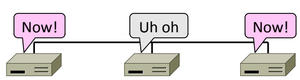



- Idea: Wait until it is done, and send

44

## CSMA "Persistence" (2)

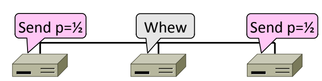
- Problem is that multiple waiting nodes will queue up then collide
  - More load, more of a problem




45

## CSMA "Persistence" (3)

- Intuition for a better solution
  - If there are N queued senders, we want each to send next with probability 1/N




46

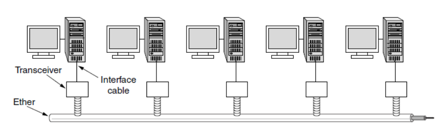
## Binary Exponential Backoff (BEB)

- Cleverly estimates the probability
  - 1st collision, wait 0 or 1 frame times
  - 2nd collision, wait from 0 to 3 times
  - 3rd collision, wait from 0 to 7 times …
- BEB doubles interval for each successive collision
  - Quickly gets large enough to work
  - Very efficient in practice

47

## Classic Ethernet, or IEEE 802.3

- Most popular LAN of the 1980s, 1990s
  - 10 Mbps over shared coaxial cable, with baseband signals
  - Multiple access with "1-persistent CSMA/CD with BEB"




48

# Ethernet Frame Format

- Has addresses to identify the sender and receiver
- CRC-32 for error detection; no ACKs or retransmission
- Start of frame identified with physical layer preamble

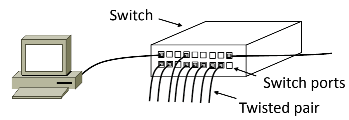
Packet from Network layer (IP)

| | Preamble | Destination address | Source address | Type | Data | Pad | Check-sum |
|---|---|---|---|---|---|---|---|
| Bytes | 8 | 6 | 6 | 2 | 0-1500 | 0-46 | 4 |

49

# Modern Ethernet

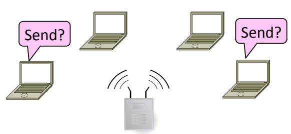
- Based on switches, not multiple access, but still called Ethernet
  - We'll get to it later




50

# Wireless Multiple Access (§4.2.5, 4.4)

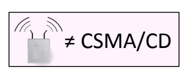
- How do wireless nodes share a single link? (Yes, this is WiFi!)
  - Build on our simple, wired model




51

# Wireless Complications

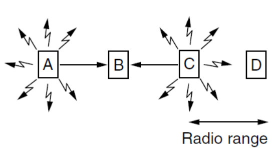
- Wireless is more complicated than the wired case (Surprise!)
  1. Nodes may have different areas of coverage – doesn't fit Carrier Sense
  2. Nodes can't hear while sending – can't Collision Detect




52

# Different Coverage Areas

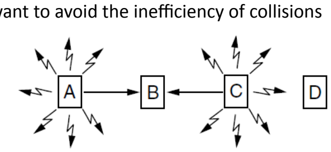
- Wireless signal is broadcast and received nearby, where there is sufficient SNR




53

# Hidden Terminals

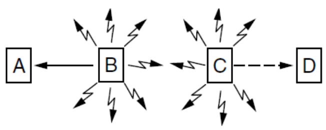
- Nodes A and C are <u>hidden terminals</u> when sending to B
  - Can't hear each other (to coordinate) yet collide at B
  - We want to avoid the inefficiency of collisions




54

## Exposed Terminals

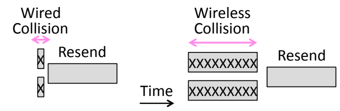
- B and C are <u>exposed terminals</u> when sending to A and D
  – Can hear each other yet don't collide at receivers A and D
  – We want to send concurrently to increase performance






## Nodes Can't Hear While Sending

- With wires, detecting collisions (and aborting) lowers their cost
- More wasted time with wireless






## Possible Solution: MACA

- MACA uses a short handshake instead of CSMA (Karn, 1990)
  – 802.11 uses a refinement of MACA (later)
- Protocol rules:
  1. A sender node transmits a RTS (Request-To-Send, with frame length)
  2. The receiver replies with a CTS (Clear-To-Send, with frame length)
  3. Sender transmits the frame while nodes hearing the CTS stay silent
  – Collisions on the RTS/CTS are still possible, but less likely



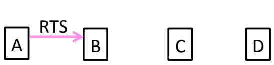
## MACA – Hidden Terminals

- A→B with hidden terminal C
  1. A sends RTS, to B




## MACA – Hidden Terminals (2)

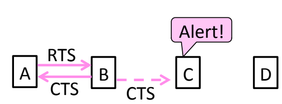
- A→B with hidden terminal C
  2. B sends CTS, to A, and C too





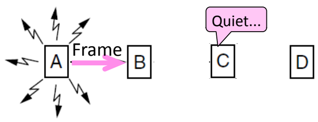

## MACA – Hidden Terminals (3)

- A→B with hidden terminal C
  2. B sends CTS, to A, and C too



## MACA – Hidden Terminals (4)

- A→B with hidden terminal C
  3. A sends frame while C defers





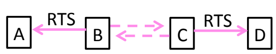

## MACA – Exposed Terminals

- B→A, C→D as exposed terminals
  – B and C send RTS to A and D




## MACA – Exposed Terminals (2)

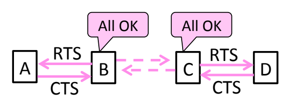
- B→A, C→D as exposed terminals
  – A and D send CTS to B and C






## MACA – Exposed Terminals (3)

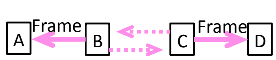
- B→A, C→D as exposed terminals
  – A and D send CTS to B and C






## MACA – Exposed Terminals (4)

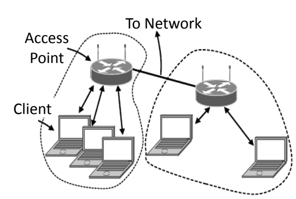
- B→A, C→D as exposed terminals
  – A and D send CTS to B and C






## 802.11, or WiFi

- Very popular wireless LAN started in the 1990s
- Clients get connectivity from a (wired) AP (Access Point)
- It's a multi-access problem ☺
- Various flavors have been developed over time
  – Faster, more features



## 802.11 Physical Layer

- Uses 20/40 MHz channels on ISM bands
  - 802.11b/g/n on 2.4 GHz
  - 802.11 a/n on 5 GHz
- OFDM modulation (except legacy 802.11b)
  - Different amplitudes/phases for varying SNRs
  - Rates from 6 to 54 Mbps plus error correction
  - 802.11n uses multiple antennas; see "802.11 with Multiple Antennas for Dummies"

67

## 802.11 Link Layer

- Multiple access uses CSMA/CA (next); RTS/CTS optional
- Frames are ACKed and retransmitted with ARQ
- Funky addressing (three addresses!) due to AP
- Errors are detected with a 32-bit CRC
- Many, many features (e.g., encryption, power save)

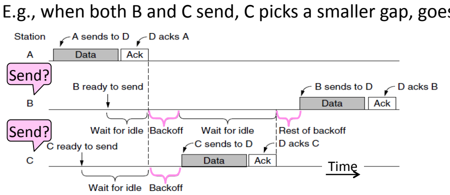
Packet from Network layer (IP)

| | Frame control | Duration | Address 1 (recipient) | Address 2 (transmitter) | Address 3 | Sequence | Data | Check sequence |
|---|---|---|---|---|---|---|---|---|
| Bytes | 2 | 2 | 6 | 6 | 6 | 2 | 0–2312 | 4 |

68

## 802.11 CSMA/CA for Multiple Access

- Sender avoids collisions by inserting small random gaps
  - E.g., when both B and C send, C picks a smaller gap, goes first




69

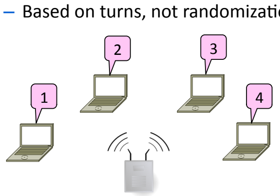
## The Future of 802.11 (Guess)

- Likely ubiquitous for Internet connectivity
  - Greater diversity, from low- to high-end devices
- Innovation in physical layer drives speed
  - And power-efficient operation too
- More seamless integration of connectivity
  - Too manual now, and limited (e.g., device-to-device)

70

## Contention-Free Multiple Access (§4.2.3)

- A new approach to multiple access
  - Based on turns, not randomization




71

## Issues with Random Multiple Access

- CSMA is good under low load:
  - Grants immediate access
  - Little overhead (few collisions)
- But not so good under high load:
  - High overhead (expect collisions)
  - Access time varies (lucky/unlucky)
- We want to do better under load!

72

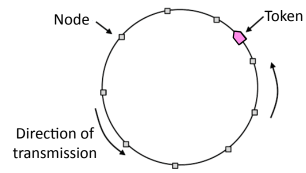
## Turn-Taking Multiple Access Protocols

- They define an order in which nodes get a chance to send
  - Or pass, if no traffic at present
- We just need some ordering …
  - E.g., Token Ring
  - E.g., node addresses

73

## Token Ring

- Arrange nodes in a ring; token rotates "permission to send" to each node in turn




74

## Turn-Taking Advantages

- Fixed overhead with no collisions
  - More efficient under load
- Regular chance to send with no unlucky nodes
  - Predictable service, easily extended to guaranteed quality of service

75

## Turn-Taking Disadvantages

- Complexity
  - More things that can go wrong than random access protocols!
    - E.g., what if the token is lost?
  - Higher overhead at low load

76

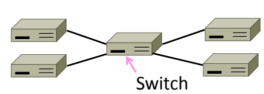
## Turn-Taking in Practice

- Regularly tried as an improvement offering better service
  - E.g., qualities of service
- But random multiple access is hard to beat
  - Simple, and usually good enough
  - Scales from few to many nodes

77

## LAN Switches (§4.3.4, 4.8.1-4.8.2, 4.8.4)

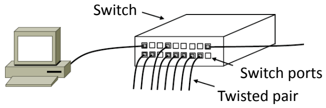
- How do we connect nodes with a switch instead of multiple access
  - Uses multiple links/wires
  - Basis of modern (switched) Ethernet




78

## Switched Ethernet

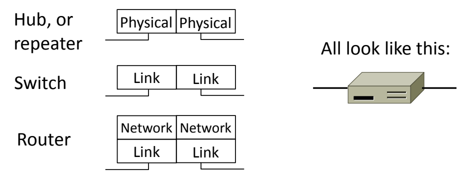
- Hosts are wired to Ethernet switches with twisted pair
  - Switch serves to connect the hosts
  - Wires usually run to a closet






## What's in the box?

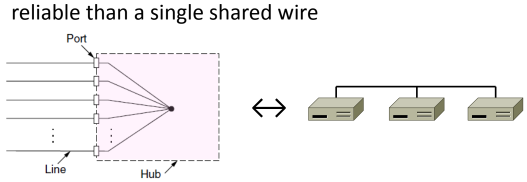
- Remember from protocol layers:






## Inside a Hub

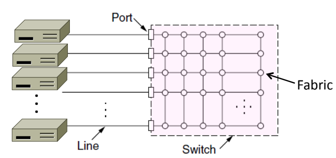
- All ports are wired together; more convenient and reliable than a single shared wire






## Inside a Switch

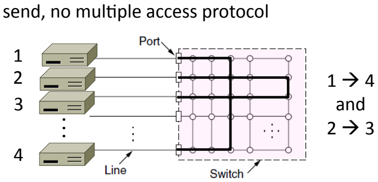
- Uses frame addresses to connect input port to the right output port; multiple frames may be switched in parallel






## Inside a Switch (2)

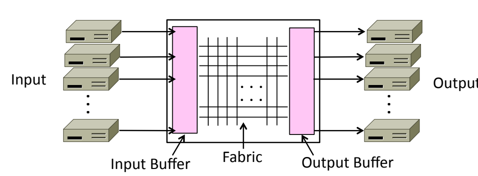
- Port may be used for both input and output (full-duplex)
  - Just send, no multiple access protocol





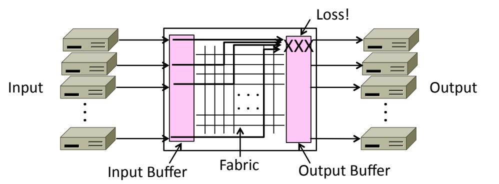

## Inside a Switch (3)

- Need buffers for multiple inputs to send to one output



## Inside a Switch (4)

- Sustained overload will fill buffer and lead to frame loss





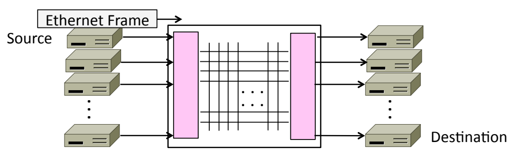

## Advantages of Switches

- Switches and hubs have replaced the shared cable of classic Ethernet
  - Convenient to run wires to one location
  - More reliable; wire cut is not a single point of failure that is hard to find
- Switches offer scalable performance
  - E.g., 100 Mbps per port instead of 100 Mbps for all nodes of shared cable / hub



## Switch Forwarding

- Switch needs to find the right output port for the destination address in the Ethernet frame. How?
  - Want to let hosts be moved around readily; don't look at IP





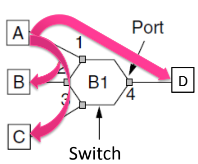

## Backward Learning

- Switch forwards frames with a port/address table as follows:
  1. To fill the table, it looks at the source address of input frames
  2. To forward, it sends to the port, or else broadcasts to all ports



## Backward Learning (2)

- 1: A sends to D

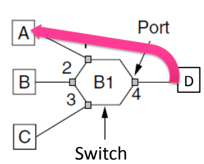



| Address | Port |
|---|---|
| A | |
| B | |
| C | |
| D | |



## Backward Learning (3)

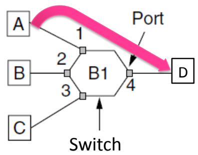
- 2: D sends to A




| Address | Port |
|---|---|
| A | 1 |
| B | |
| C | |
| D | |

## Backward Learning (4)

- 3: A sends to D

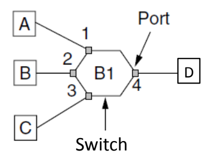



| Address | Port |
|---|---|
| A | 1 |
| B | |
| C | |
| D | 4 |

91

## Backward Learning (5)

- 3: A sends to D

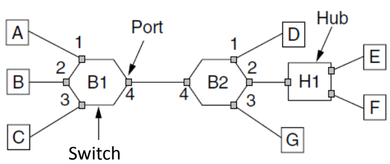



| Address | Port |
|---|---|
| A | 1 |
| B | |
| C | |
| D | 4 |

92

## Learning with Multiple Switches

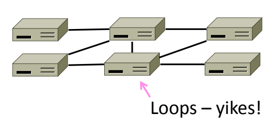
- Just works with multiple switches and a mix of hubs *assuming no loo*ps, e.g., A sends to D then D sends to A




93

## Switch Spanning Tree (§4.8.3)

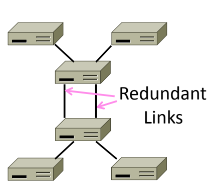
- How can we connect switches in any topology so they just work?




94

## Problem – Forwarding Loops

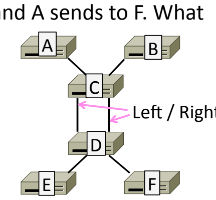
- May have a loop in the topology
  - Redundancy in case of failures
  - Or a simple mistake
- Want LAN switches to "just work"
  - Plug-and-play, no changes to hosts
  - But loops cause a problem …




95

## Forwarding Loops (2)

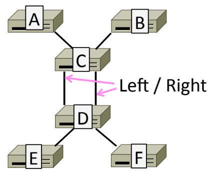
- Suppose the network is started and A sends to F. What happens?




96

## Forwarding Loops (3)

- Suppose the network is started and A sends to F. What happens?
  - A → C → B, D-left, D-right
  - D-left → C-right, E, F
  - D-right → C-left, E, F
  - C-right → D-left, A, B
  - C-left → D-right, A, B
  - D-left → …
  - D-right → …



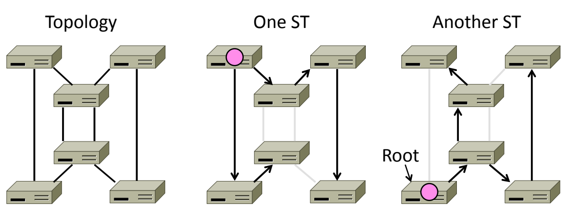

97

## Spanning Tree Solution

- Switches collectively find a spanning tree for the topology
  - A subset of links that is a tree (no loops) and reaches all switches
  - Switches forward as normal but only on spanning tree
  - Broadcasts will go up to the root of the tree and down all the branches

98

## Spanning Tree (2)




99

## Radia Perlman (1951–)

- Key early work on routing protocols
  - Routing in the ARPANET
  - Spanning Tree for switches (next)
  - Link-state routing (later)
- Now focused on network security

100

## Spanning Tree Algorithm

- Rules of the distributed game:
  - All switches run the same algorithm
  - They start with no information
  - Operate in parallel and send messages
  - Always search for the best solution
- Ensures a highly robust solution
  - Any topology, with no configuration
  - Adapts to link/switch failures, …

101

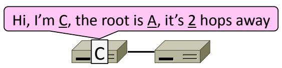
## Spanning Tree Algorithm (2)

- Outline:
  1. Elect a root node of the tree (switch with the lowest address)
  2. Grow tree as shortest distances from the root (using lowest address to break distance ties)
  3. Turn off ports for forwarding if they are not on the spanning tree

102

## Spanning Tree Algorithm (3)

- Details:
  - Each switch initially believes it is the root of the tree
  - Each switch sends periodic updates to neighbors with:
    - Its address, address of the root, and distance (in hops) to root
  - Switches favors ports with shorter distances to lowest root
    - Uses lowest address as a tie for distances

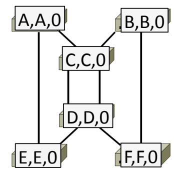



or (C, A, 2)

103

## Spanning Tree Example

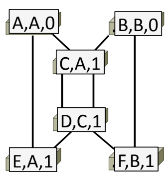
- 1st round, sending:
  - A sends (A, A, 0) to say it is root
  - B, C, D, E, and F do likewise
- 1st round, receiving:
  - A still thinks is it (A, A, 0)
  - B still thinks (B, B, 0)
  - C updates to (C, A, 1)
  - D updates to (D, C, 1)
  - E updates to (E, A, 1)
  - F updates to (F, B, 1)




104

## Spanning Tree Example (2)

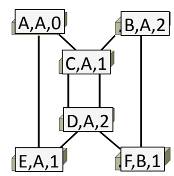
- 2nd round, sending
  - Nodes send their updated state
- 2nd round receiving:
  - A remains (A, A, 0)
  - B updates to (B, A, 2) via C
  - C remains (C, A, 1)
  - D updates to (D, A, 2) via C
  - E remains (E, A, 1)
  - F remains (F, B, 1)




105

## Spanning Tree Example (3)

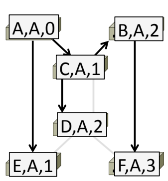
- 3rd round, sending
  - Nodes send their updated state
- 3rd round receiving:
  - A remains (A, A, 0)
  - B remains (B, A, 2) via C
  - C remains (C, A, 1)
  - D remains (D, A, 2) via C-left
  - E remains (E, A, 1)
  - F updates to (F, A, 3) via B




106

## Spanning Tree Example (4)

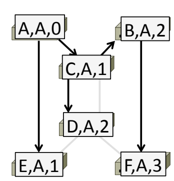
- 4th round
  - Steady-state has been reached
  - Nodes turn off forwarding that is not on the spanning tree
- Algorithm continues to run
  - Adapts by timing out information
  - E.g., if A fails, other nodes forget it, and B will become the new root




107

## Spanning Tree Example (5)

- Forwarding proceeds as usual on the ST
- Initially D sends to F:
- And F sends back to D:

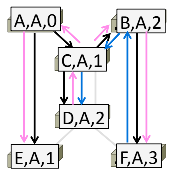



108

# Spanning Tree Example (6)

- Forwarding proceeds as usual on the ST
- Initially D sends to F:
  - D → C-left
  - C → A, B
  - A → E
  - B → F
- And F sends back to D:
  - F → B
  - B → C
  - C → D

  (hm, not such a great route)




109