Bridges and LAN Switches

Hui Chen, Ph.D.
Dept. of Engineering & Computer Science
Virginia State University
Petersburg, VA 23806

Acknowledgements

- Some pictures used in this presentation were obtained from the Internet
- The instructor used the following references
  - Larry L. Peterson and Bruce S. Davie, Computer Networks: A Systems Approach, 5th Edition, Elsevier, 2011
  - Andrew S. Tanenbaum, Computer Networks, 5th Edition, Prentice-Hall, 2010
  - James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach, 5th Ed., Addison Wesley, 2009
  - Larry L. Peterson's (http://www.cs.princeton.edu/~llp/) Computer Networks class web site

Outline

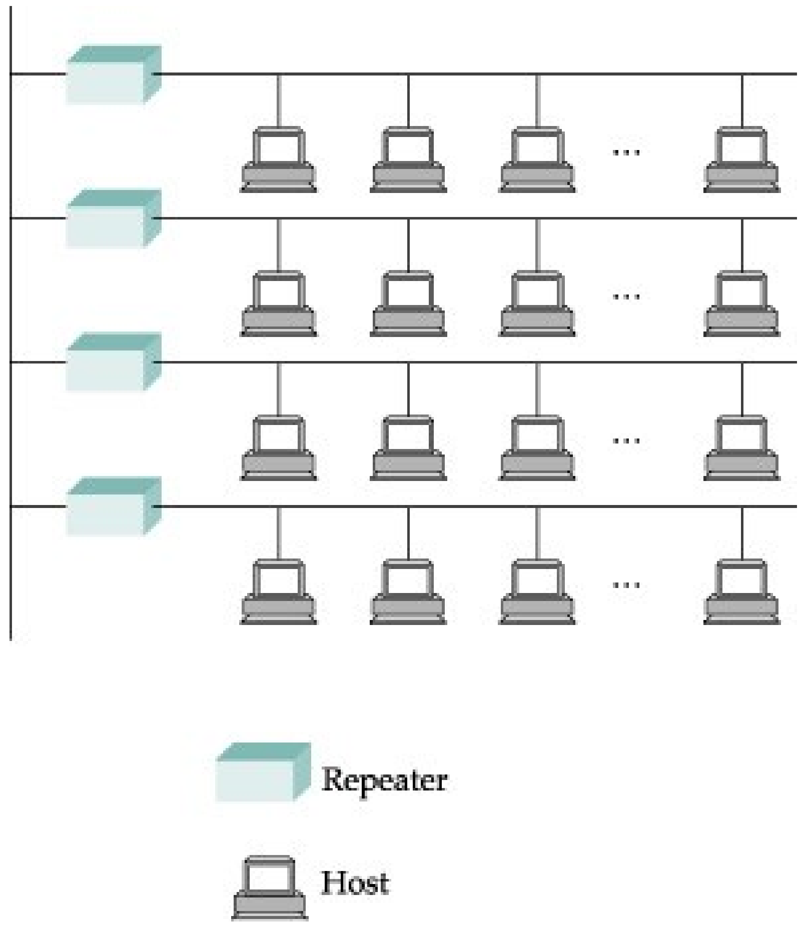
- Review
  - Discussed packet switching – **Expand networks**
    - Datagram switching
    - Virtual circuit switching
    - Source routing
- **Example of real networks**
  - **Ethernet**
  - **How to expand Ethernet?**

Ethernet LAN: How to Expand?

- Expand an Ethernet local area network (LAN)
  - Repeaters
  - Bridges
  - Switches

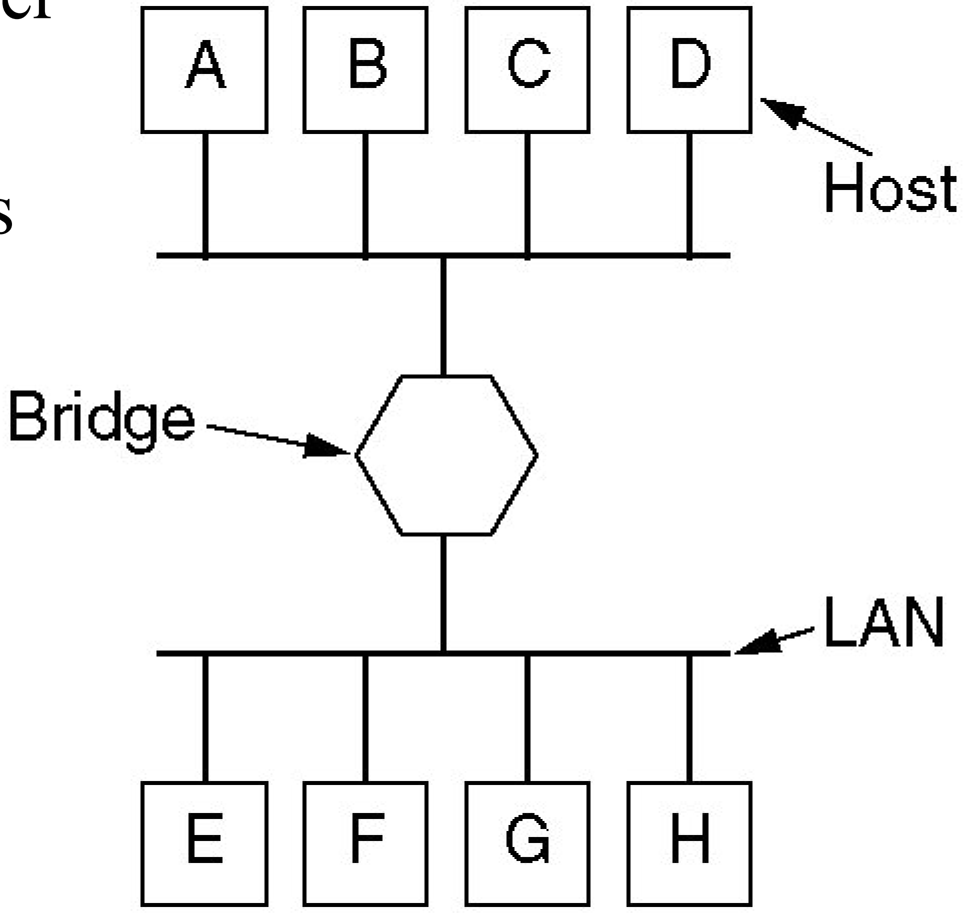
Repeaters

- Devices in physical layer
- Receive, amplify (regenerate), and retransmit signal in both directions.
- Example: Ethernet repeaters



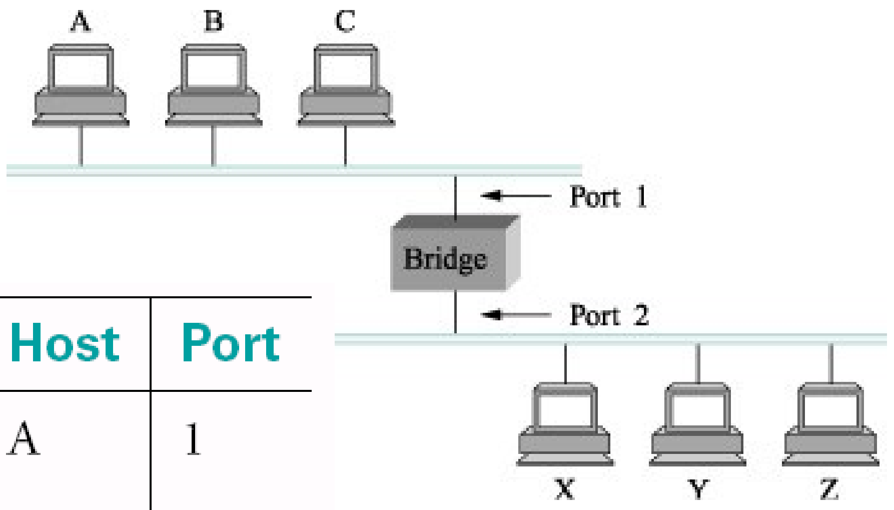
Bridges

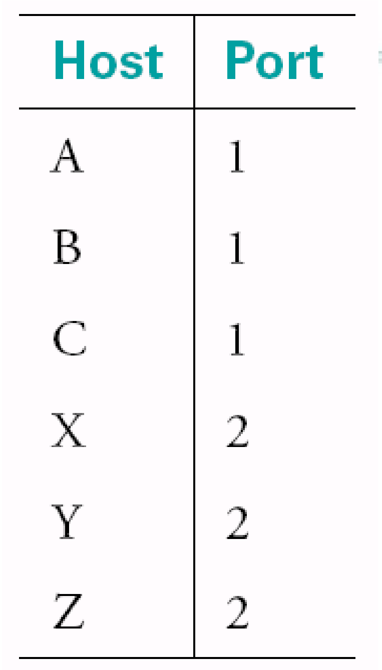
- Devices in data link layer
- In promiscuous mode
- Forward packets/frames to either connected networks
- Example: Ethernet bridges



Ethernet Switches: Learning Bridges

- Do you actually need to forward every frames when
  - A → B?
  - A → X?
- Improvement
  - Do not forward unnecessarily
  - Relying on a **forwarding table** at each switch

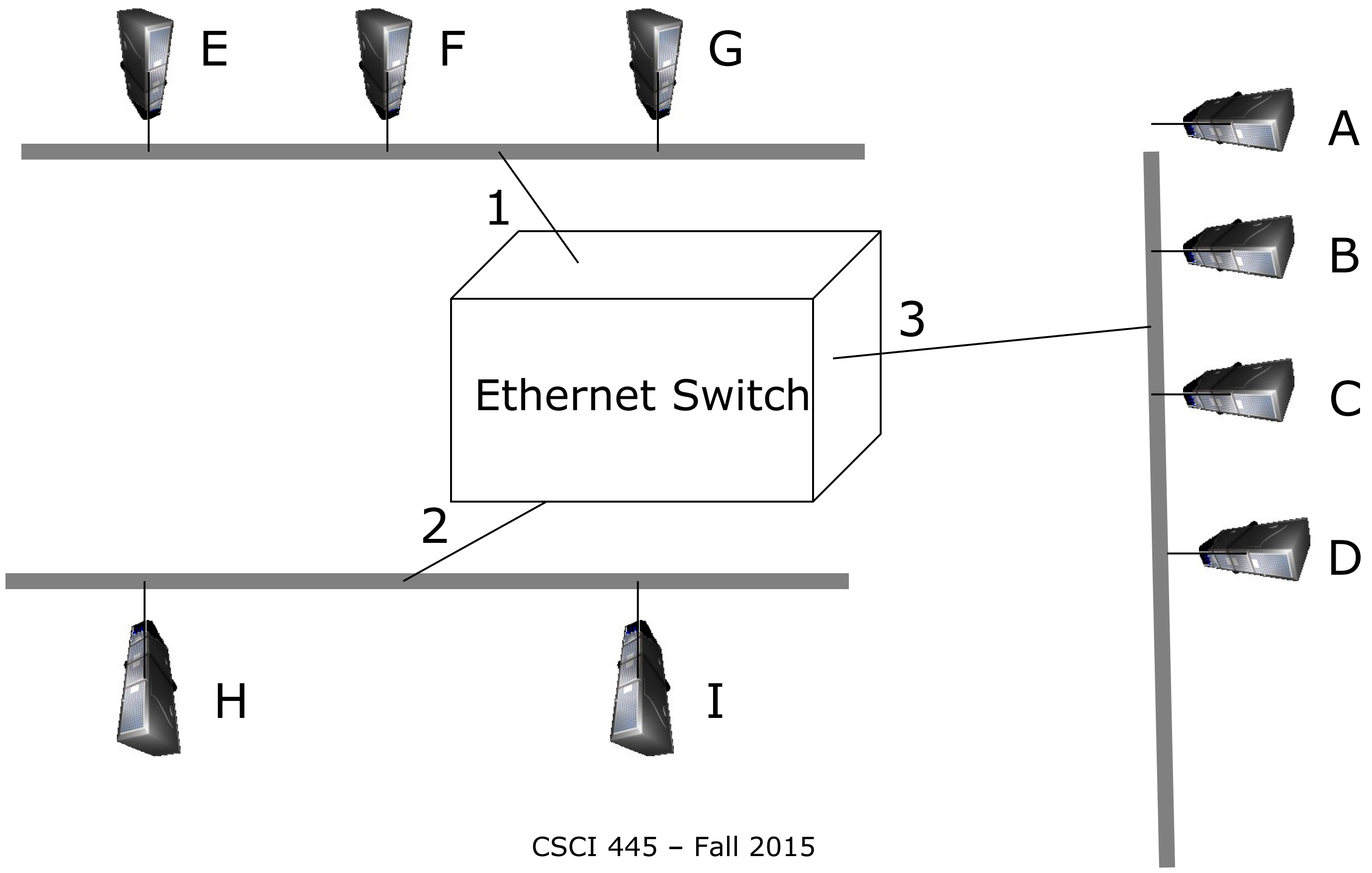




Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

Exercise L9-1

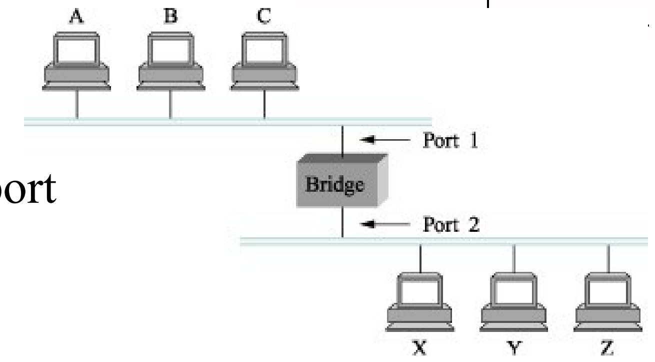
- Build a complete "forwarding" table for the bridge



Forwarding Algorithm

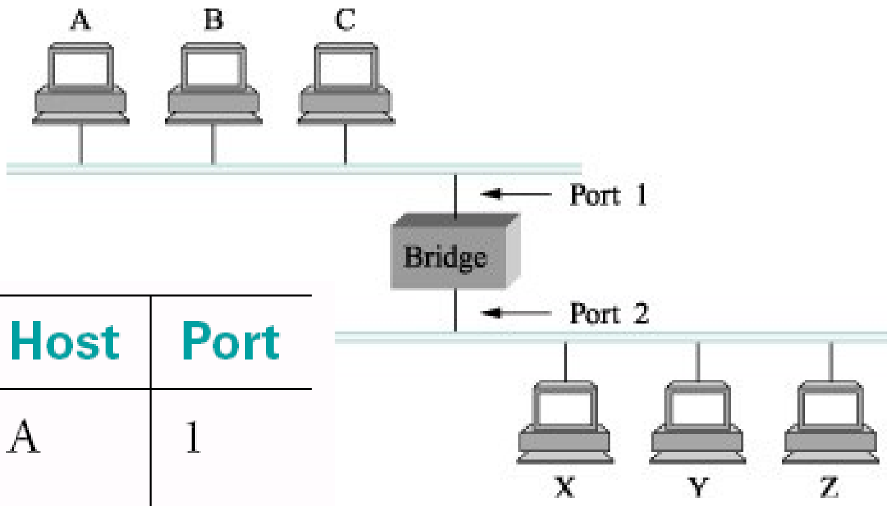
- Destination address in frame header indicates which host a frame is addressed to
- Source address in frame header indicates which host a frame is originated
- Each bridge maintains a "forwarding" table
- Algorithm
  - On receiving a frame *(src, dst, receiving port)*, look up *dst* in the forwarding table
    - If *dst* is found
      - If port in forwarding table = receiving port, do not forward;
      - otherwise, forward to the forwarding port
    - <u>Otherwise, forward to all ports</u>

Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2



Ethernet Switches: Learning Bridges

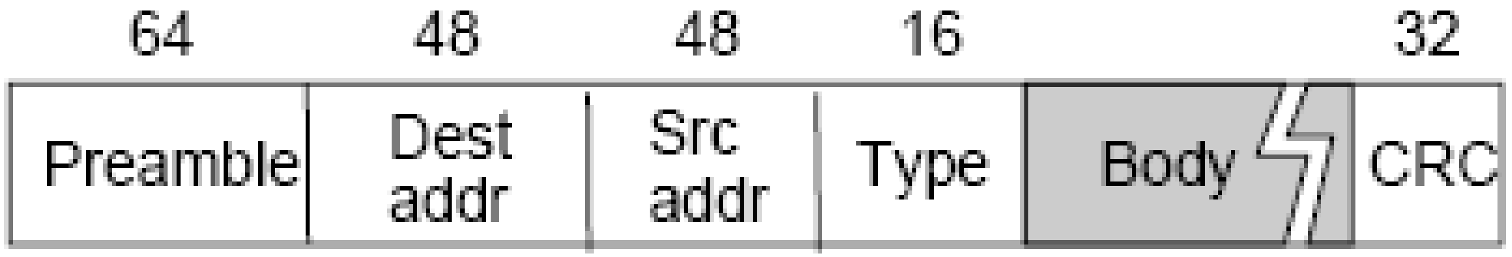
- Improvement
  - Forwarding table
  - Do not forward unnecessarily to receiving port.
- How to maintain the table?
  - Manually?
  - Automatically?
- Learning bridges
  - **automatically** maintain the table without human intervention → "learning" from received frames → Ethernet switches = <u>Learning</u> bridges
- How does it learn?



Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

Ethernet Switches: Learning from Received Frame

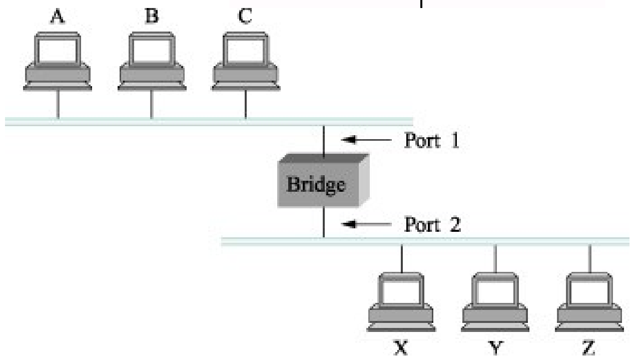
- From a received frame, an Ethernet switch knows
  - Destination address and source address
- Receiving port number on the switch



Learning Bridges: Learning Algorithm

- Destination address in frame header indicates which host a frame is addressed to
- Source address in frame header indicates which host a frame is originated
- Each bridge maintains a “forwarding” table, initially empty
- Algorithm
  - On receiving a frame *(src, dst, receiving port)*, look up *src* in the forwarding table
    - If *src* is not found
      - Insert (src, receiving port number) to the forwarding table

Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

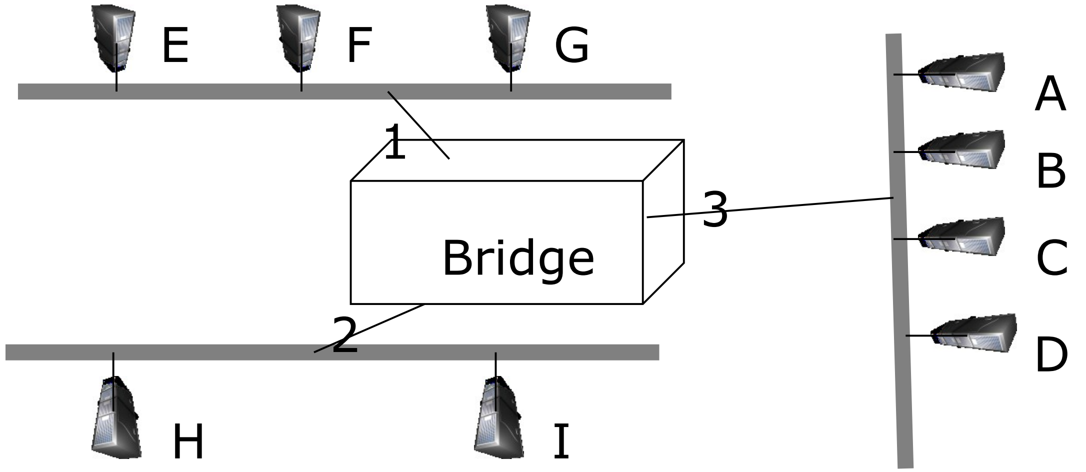


Example

- Describe the table built by the switch as the following frames arrives
  - Starting with the table as follows

Host	Port
B	3

  - The following frames (indicated by sending hosts) are received by the bridge as time goes
    - I, H, B, F (reads, first I, then H, then B, and then F)
    - Please draw four tables to show the resulting table after each frame is processed



Example: Answer

0. Initial table

Host	Port
B	3

1. Frame sent from Host I arrives

Host	Port
B	3
I	2

2. Frame sent from Host H arrives

Host	Port
B	3
I	2
H	2

3. Frame sent from Host B arrives

Host	Port
B	3
I	2
H	2

4. Frame sent from Host F arrives

Host	Port
B	3
I	2
H	2
F	1

Example: Question 1

0. Initial table

Host	Port
B	3

Q: which hosts will see the frame sent from Host I?

Example: Question 2

0. Initial table

Host	Port
B	3

Q: which hosts will see the frame sent from Host B?

1. Frame sent from Host I arrives

Host	Port
B	3
I	2

2. Frame sent from Host H arrives

Host	Port
B	3
I	2
H	2

Exercise L9-2

- Build a “forwarding” table for the Ethernet shown as the following transmissions happen
  - A sends to C; C sends to A; E sends to I; I sends to E; E sends to B

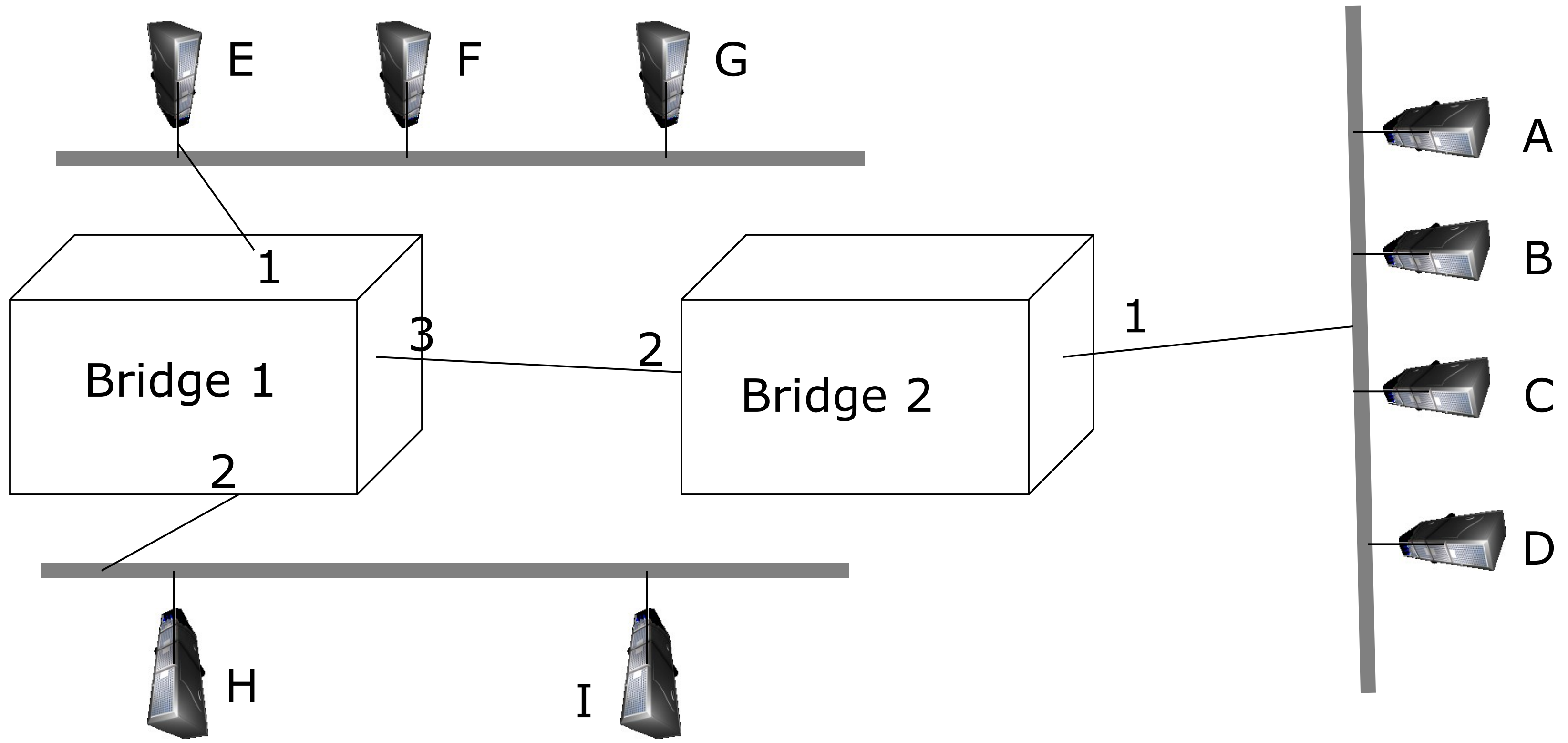


Table Maintenance Algorithm for Learning Bridges (1)

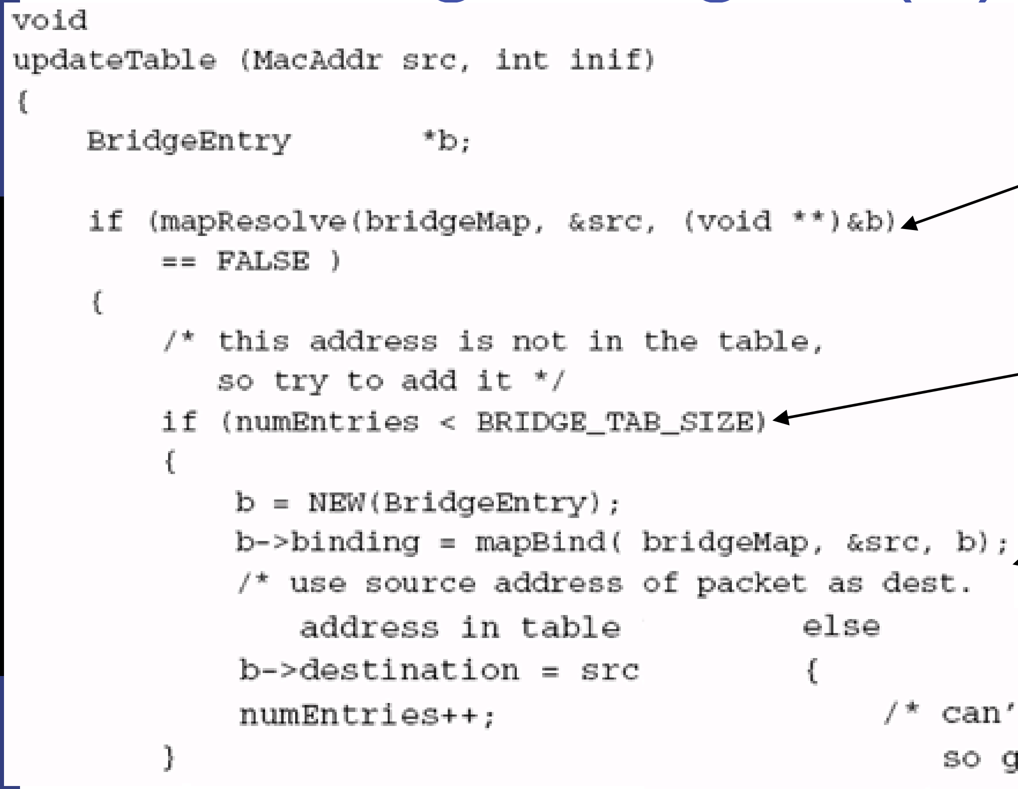
```
#define BRIDGE_TAB_SIZE 1024 /* max. size of bridging
                               table */
#define MAX_TTL         120  /* time (in seconds) before
                                   an entry is flushed */

typedef struct {
    MacAddr     destination;  /* MAC address of a node */
    int         ifnumber;     /* interface to reach it */
    u_short     TTL;          /* time to live */
    Binding     binding;      /* binding in the Map */
} BridgeEntry;

int     numEntries = 0;
Map     bridgeMap = mapCreate(BRIDGE_TAB_SIZE,
                              sizeof(BridgeEntry));
```

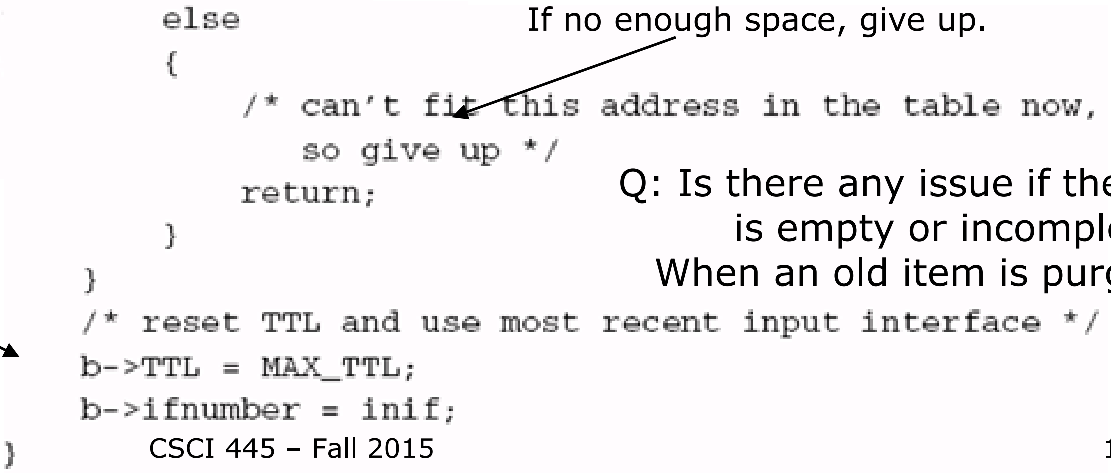
Table initialized: empty table that can hold up to BridgeEntry number of entries created

Table Maintenance Algorithm for Learning Bridges (2)



```
void
updateTable (MacAddr src, int inif)
{
    BridgeEntry       *b;

    if (mapResolve(bridgeMap, &src, (void **)&b)
        == FALSE )
    {
        /* this address is not in the table,
           so try to add it */
        if (numEntries < BRIDGE_TAB_SIZE)
        {
            b = NEW(BridgeEntry);
            b->binding = mapBind( bridgeMap, &src, b);
            /* use source address of packet as dest.
               address in table
            b->destination = src
            numEntries++;
        }
```



```
        else
        {
            /* can't fit this address in the table now,
               so give up */
            return;
        }
    }
    /* reset TTL and use most recent input interface */
    b->TTL = MAX_TTL;
    b->ifnumber = inif;
}
```

Can the source address of the frame be found in the table? Use src as an index to search the table. If entry not found, return a pointer/reference to a unused entry in variable *b*

No machine has infinite amount of resource. It is always good to check if you can store the entry

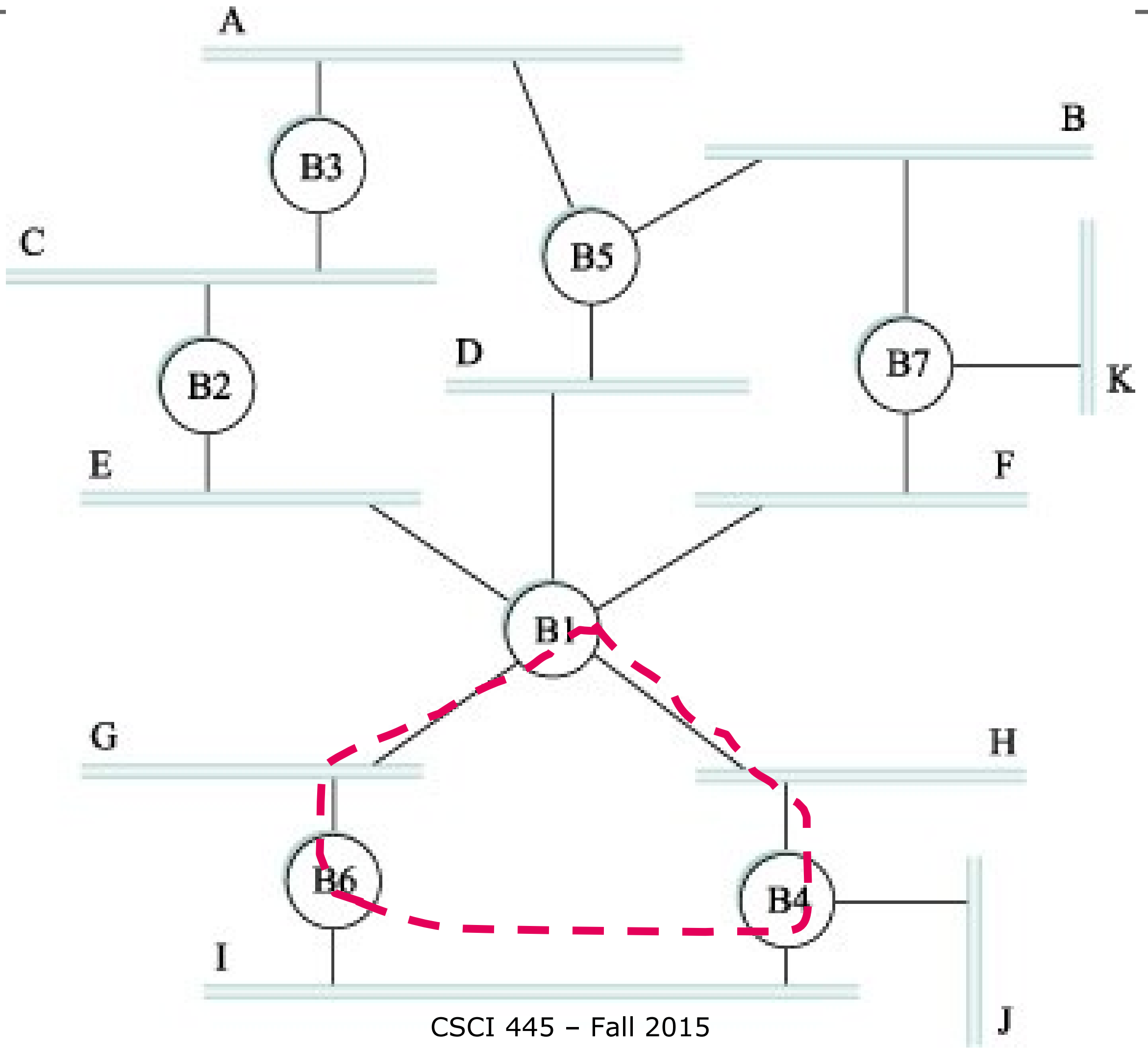
Insert the new entry into the table

If no enough space, give up.

Q: Is there any issue if the table is empty or incomplete? When an old item is purged?

If entry found, reset TTL (time-to-live)

Extended LAN Can Have Loops!

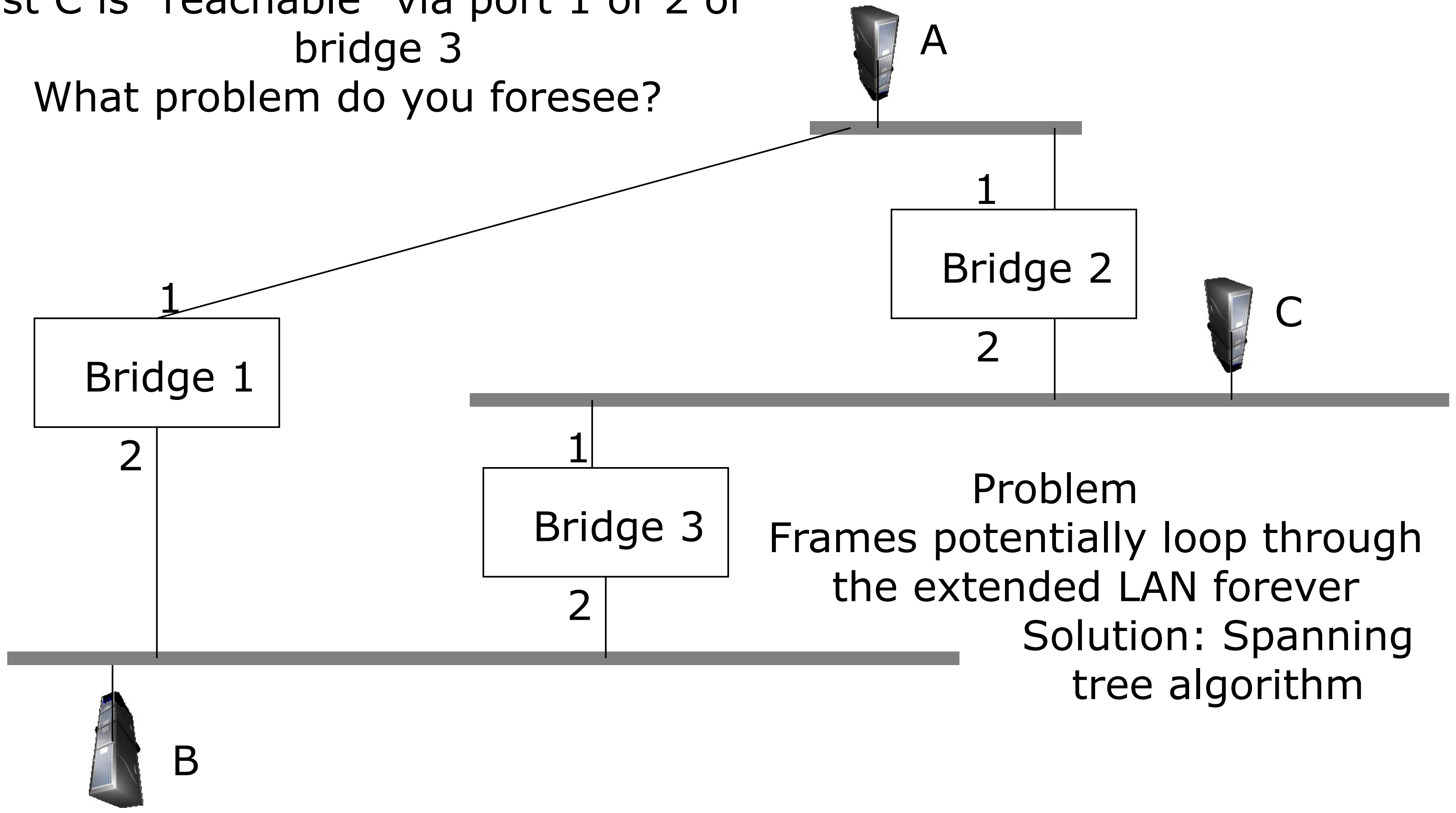


Learning Bridges: Why Are There Loops in Extended LANs?

- Created unintentionally
  - No one knows the entire topology of the network …
  - Not everyone has knowledge of what is NOT supposed to be done
- Created intentionally
  - To provide redundancy in case of failure

Loop in an Extended LAN

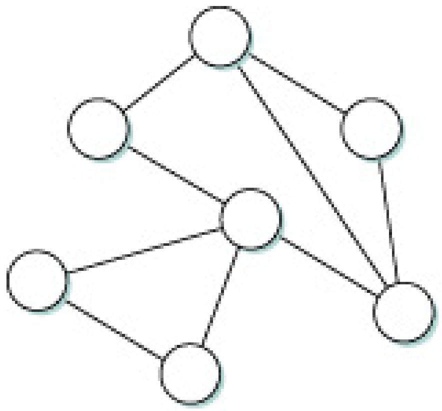
host C is "reachable" via port 1 or 2 of bridge 3
What problem do you foresee?

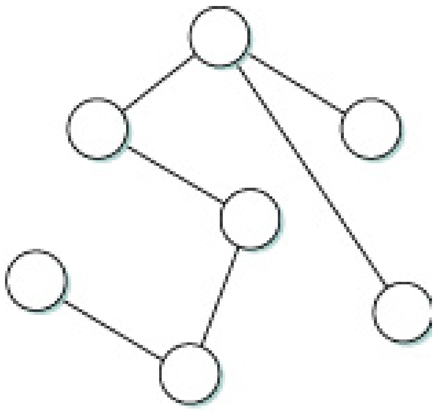


Problem
Frames potentially loop through the extended LAN forever
Solution: Spanning tree algorithm

Spanning Tree

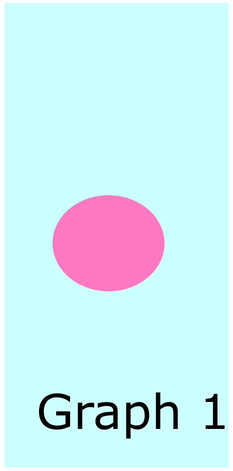
- Spanning tree: A spanning tree of an undirected graph of n nodes is a subgraph of a set of $n - 1$ edges that connects all nodes.
  - A tree is a simple, undirected, connected, acyclic graph
  - A connected graph with n nodes and n-1 edges is a tree.
  - A graph is connected if there is a path from any point to any other point in the graph.
  - A graph G is a pair (V,E), where V is a set of vertices, and E is a set of edges between the vertices $E \subseteq \{\{u,v\} \mid u, v \in V\}$.

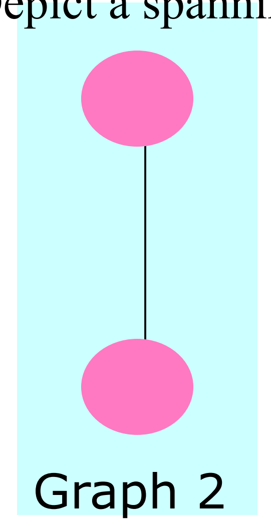

A Graph

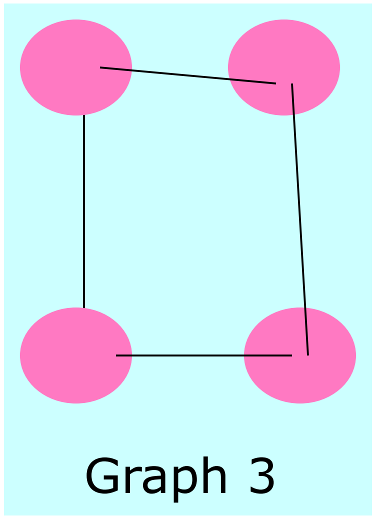

A spanning tree of the Graph

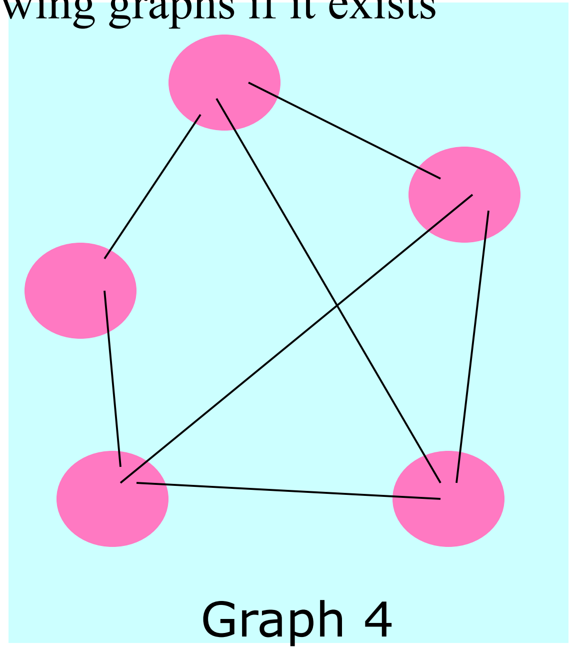
Exercise L9-3

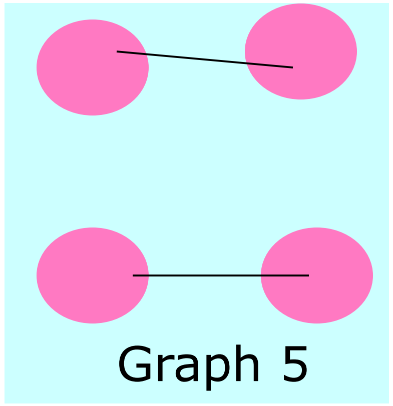
- Question
  - Indicate whether a spanning tree exists for each graph below
  - Depict a spanning tree for each of the following graphs if it exists


Graph 1


Graph 2


Graph 3

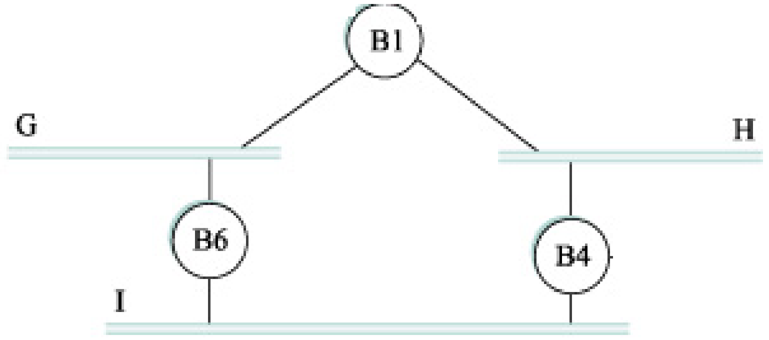

Graph 4

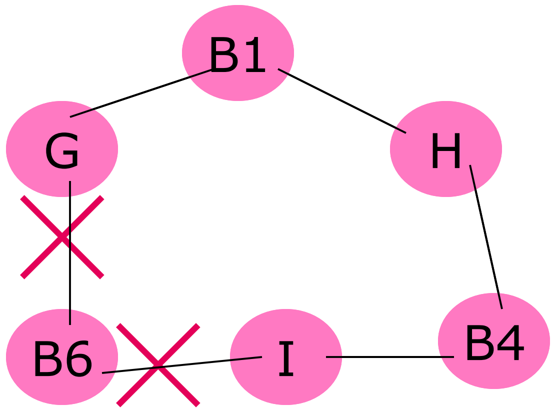

Graph 5

Please turn in your work!

An Extended LAN with Loops

- 3 bridges: B1, B4, and B6
- 3 LANs: G, H, and I
- Q: Can you draw its corresponding graph? (bridges/LANs as nodes, links as edges)?
  - Turn your work in





Q: How to break up the loops? → find "spanning tree" (pp. 194 -: bridges could be disconnected)
Q: What bridges should do then? → stop forwarding to corresponding ports

Spanning Tree Algorithm

- Breaking loops in an extended LAN

Radia Perlman. 1985. An algorithm for distributed computation of a spanning tree in an extended LAN. In Proceedings of the ninth symposium on Data communications (SIGCOMM '85). ACM, New York, NY, USA, 44-53. DOI=10.1145/319056.319004 http://doi.acm.org/10.1145/319056.319004

Spanning Tree Algorithm: Breaking Loops

- Spanning tree: A spanning tree of an undirected graph of n nodes is a graph of a set of $n - 1$ edges that connects all nodes.
- Bridges and LANs as nodes, and ports as edges
- Bridges have no knowledge of network topology
- Overview
  - Each bridge has unique id (e.g., B1, B2, B3)
  - Select bridge with smallest id as <u>root</u>
  - Select bridge on each LAN closest to root as <u>designated bridge</u> (use id to break ties)
  - Each bridge forwards frames over each LAN for which it is the <u>designated bridge</u>
- Challenge: Every bridge is equal. No centralized control! No bridge knows the entire topology!
  - Switches need to elect among each other who forward frames and who won't via exchanging messages!

Spanning Tree Algorithm: Breaking Loops

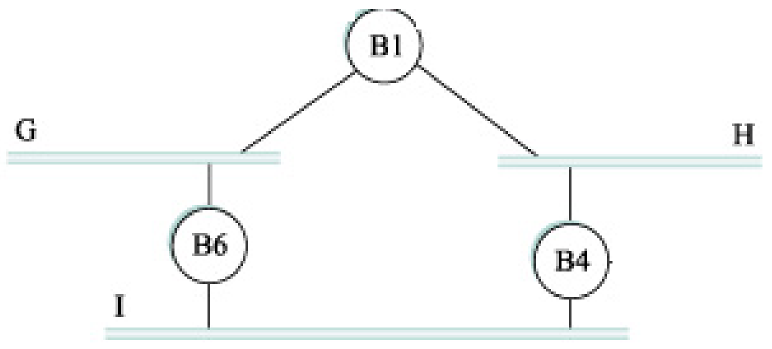
- Initially, each bridge treats itself as the root
- Bridges broadcast <u>configuration messages</u>
  - A message contains
    - id for bridge sending the message
    - id for what the sending bridge believes to be root bridge
    - distance (hops) from sending bridge to root bridge
- Each bridge records current best configuration message for each port
  - When a configuration message arrives, the bridge checks if the message is better than current best configuration message
  - A message is considered better, if any of these holds
    - The bridge identifies a root with a smaller ID
    - The bridge identifies a root with an equal ID but with shorter distance (# of hops) to the root
    - The root ID and distance are equal, but the sending bridge has a smaller ID

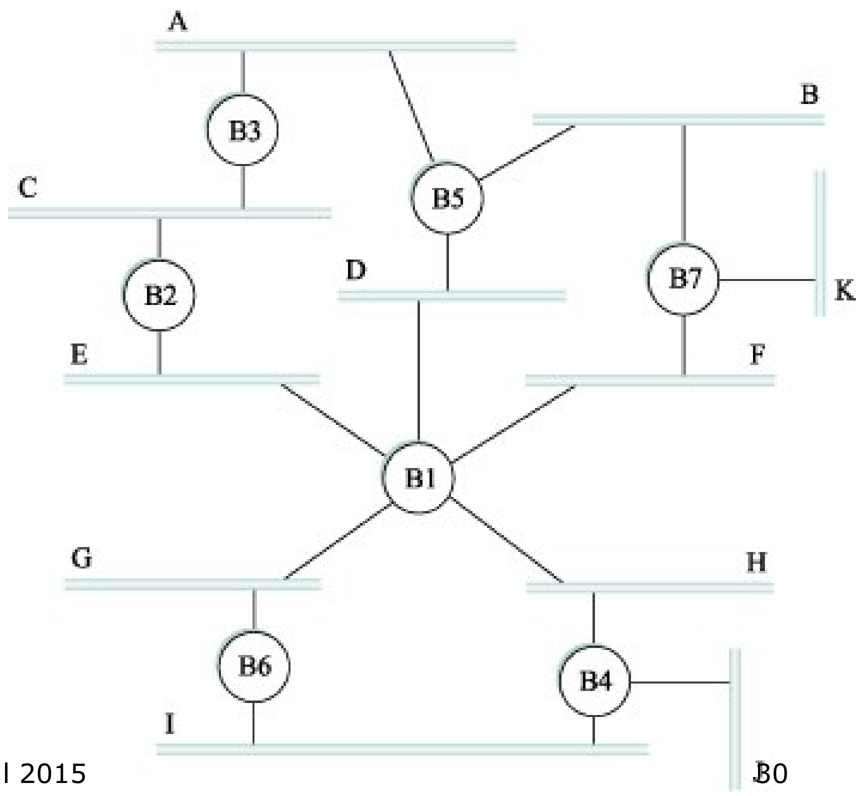
Spanning Tree Algorithm: Breaking Loops

- When a bridge learns it is not the root
  - It stops generating configuration messages
  - It only forwards configuration messages it receives (after adding 1 to the hop distance to the root)
  - In steady state, only root generates configuration messages
- When a bridge learns it is not the designated bridge for a LAN
  - It stops forwarding configuration messages to the LAN
  - in steady state, only designated bridges forward configuration messages the corresponding LAN
- Root continues to periodically send configuration messages
- Reconstruction in case of failure
  - If any bridge does not receive configuration message after a period of time, it starts generating configuration messages claiming to be the root

Example

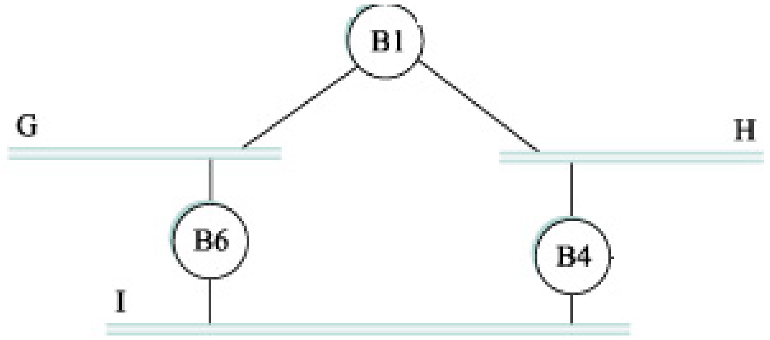
- Use the algorithm outlined to find the spanning trees for the following extended LANs





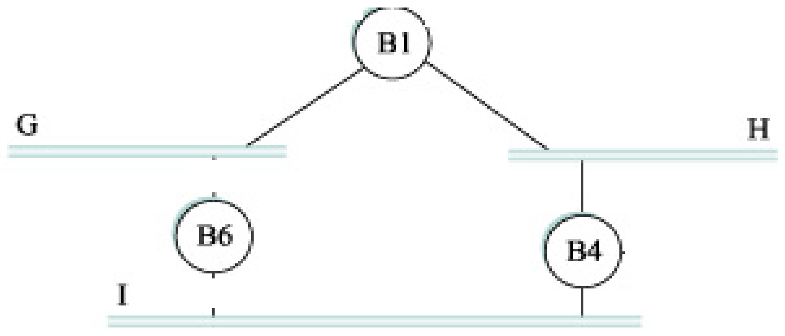
Test Run of Distributed Spanning Tree Algorithm (1)

- An extended LAN with three bridges that connects to three LANs



Test Run of Distributed Spanning Tree Algorithm (2)

1. B1, B4, and B6 broadcast configuration messages
   a) B1: (B1, 0, B1)
   b) B4: (B4, 0, B4)
   c) B6: (B6, 0, B6)
2. What happens next?
   a) B1 receives (B4, 0, B4), since 1 < 4, reject 4 as root
   b) B1 receives (B6, 0, B6), Since 1 < 6, reject 6 as root
   c) B4 receives (B1, 0, B1), since 1 < 4, make 1 as root, add 1 to the distance in the received message, and send (B1, 1, B4) to B6
   d) B4 receives (B6, 0, B6), since 4 < 6, reject 6 as root, stop forwarding this message
   e) B6 receives (B4, 0, B4), since 4 < 6, accepts 4 as root, send (B4, 1, B6) to B1
   f) B6 receives (B1, 0, B1), since 1 < 4 (current root) < 6, accepts 1 as root, send (B1, 1, B6) to B4

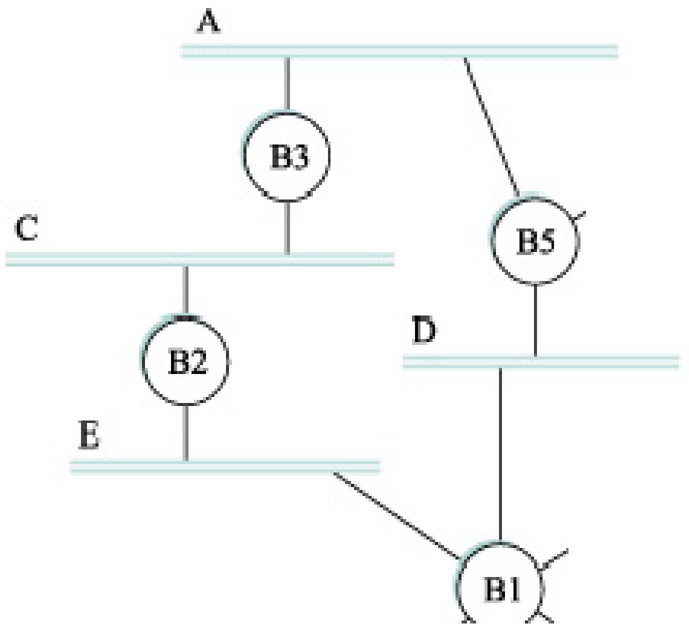


3. B4 and B6 are non-roots, stop generating configuration messages
4. B6 receives (B1, 1, B4) from B4, B6 knows it is of the same distance away from 1 as B4, however, its ID is great than B4's ID, it stops forwarding on both its interfaces
5. B4 receives (B1, 1, B6) from B6, B4 knows it is of the same distance away from 1 as B6, however, its ID is less than B6's ID, it keeps forwarding

➔ "Spanning tree" formed

Exercise L9-4

- Running/tracing distributed spanning tree algorithm
  - 4 bridges
  - 4 LANs
  - Construct the "spanning tree"



Extended LANs: Broadcasting and Multicasting

- Current practice
  - Forward all broadcast/multicast frames
- Potential improvement
  - Learn when no group members downstream
    - Accomplished by having each member of group G send a frame to bridge multicast address with G in source field

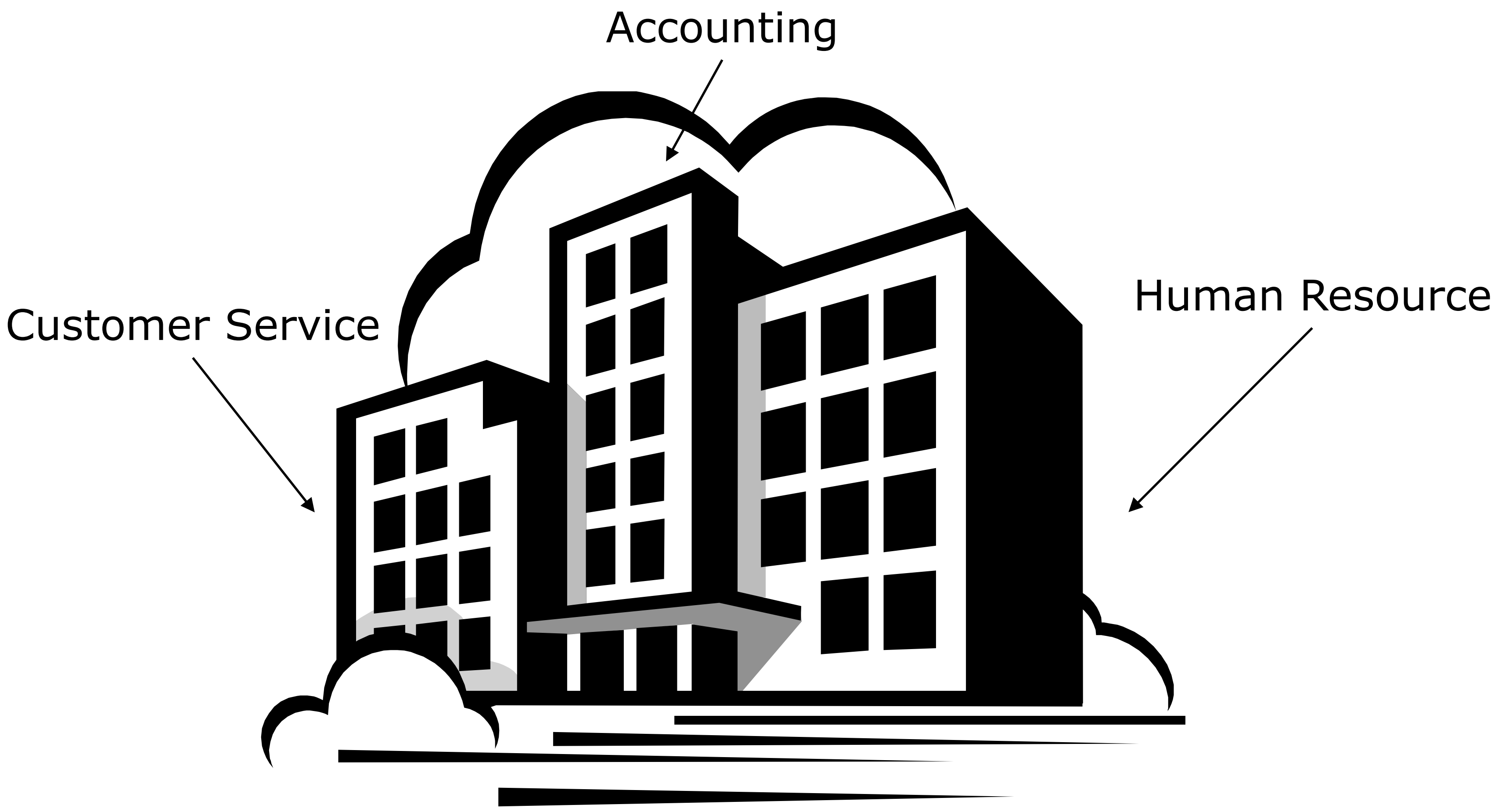
Limitations of Bridges

- Potential scalability problem
  - spanning tree algorithm scales only linearly → does not scale up well
  - no one needs receive messages from every one → broadcast does not scale
- Switches/bridges run on data link layer → rely on frames header → supports only the same types of networks → do not accommodate heterogeneity
- Advantage
  - Runs on data link layer → multiple LANs connected transparently → end hosts do not need run additional protocols
- Caution: beware of transparency

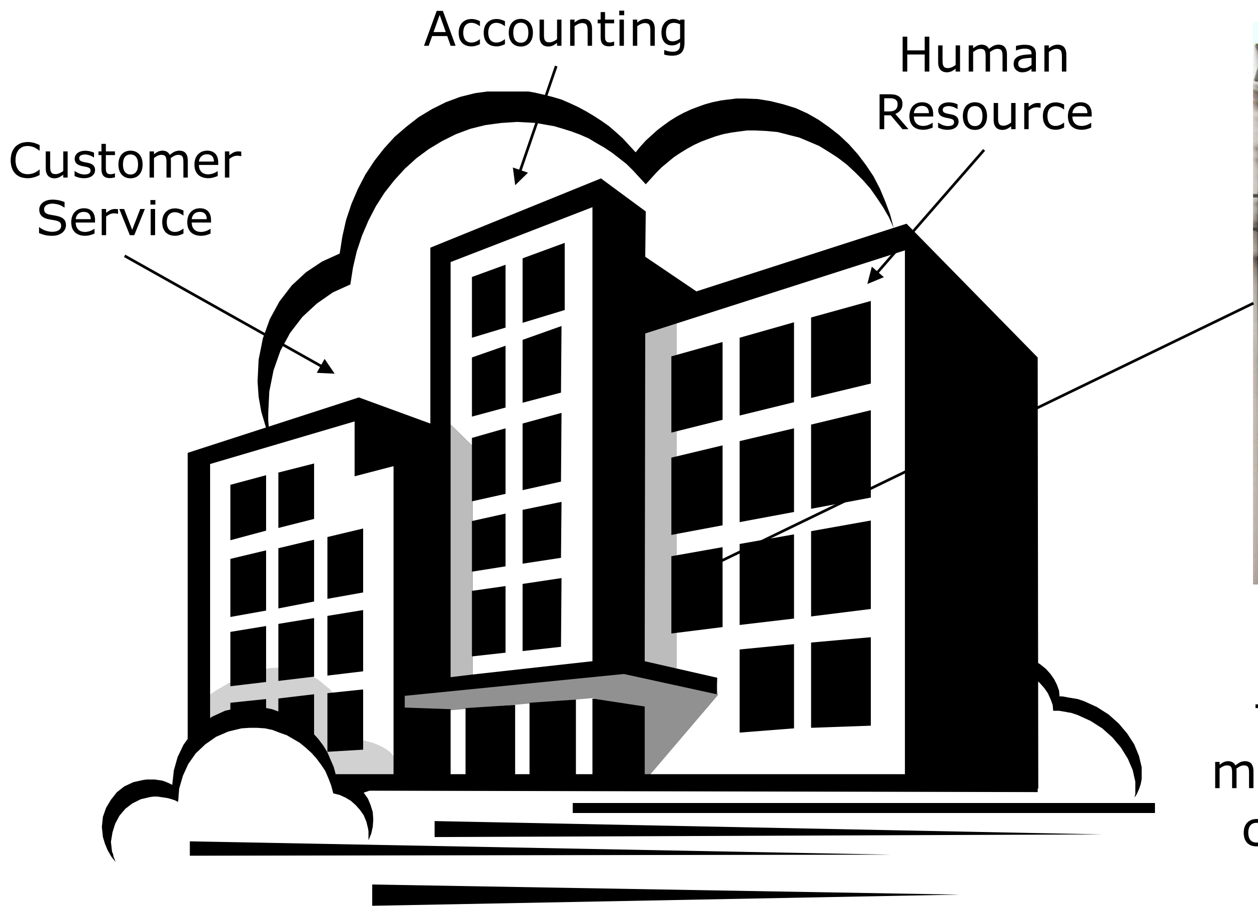
VLAN

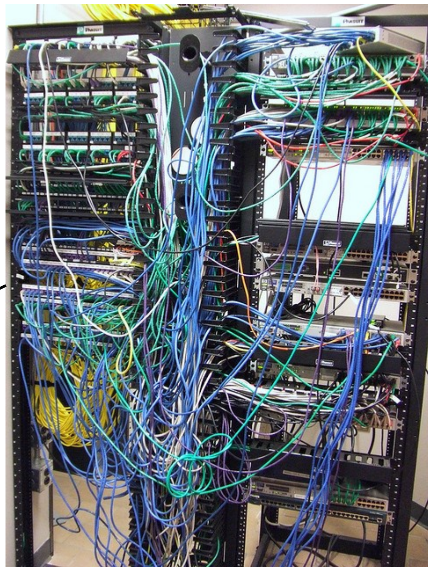
- Virtual LANs (VLANs)
- Extended LAN grows, depth of spanning tree increases, lower performance (longer latency and more frame forwarding)
- VLAN
  - Partition a single extended LAN into several <u>seemingly/logically separated segments</u>
  - Advantage
    - Each VLAN is a smaller LAN (depth of spanning tree?)
    - Logical seperation

Logical Separation



Logical Separation

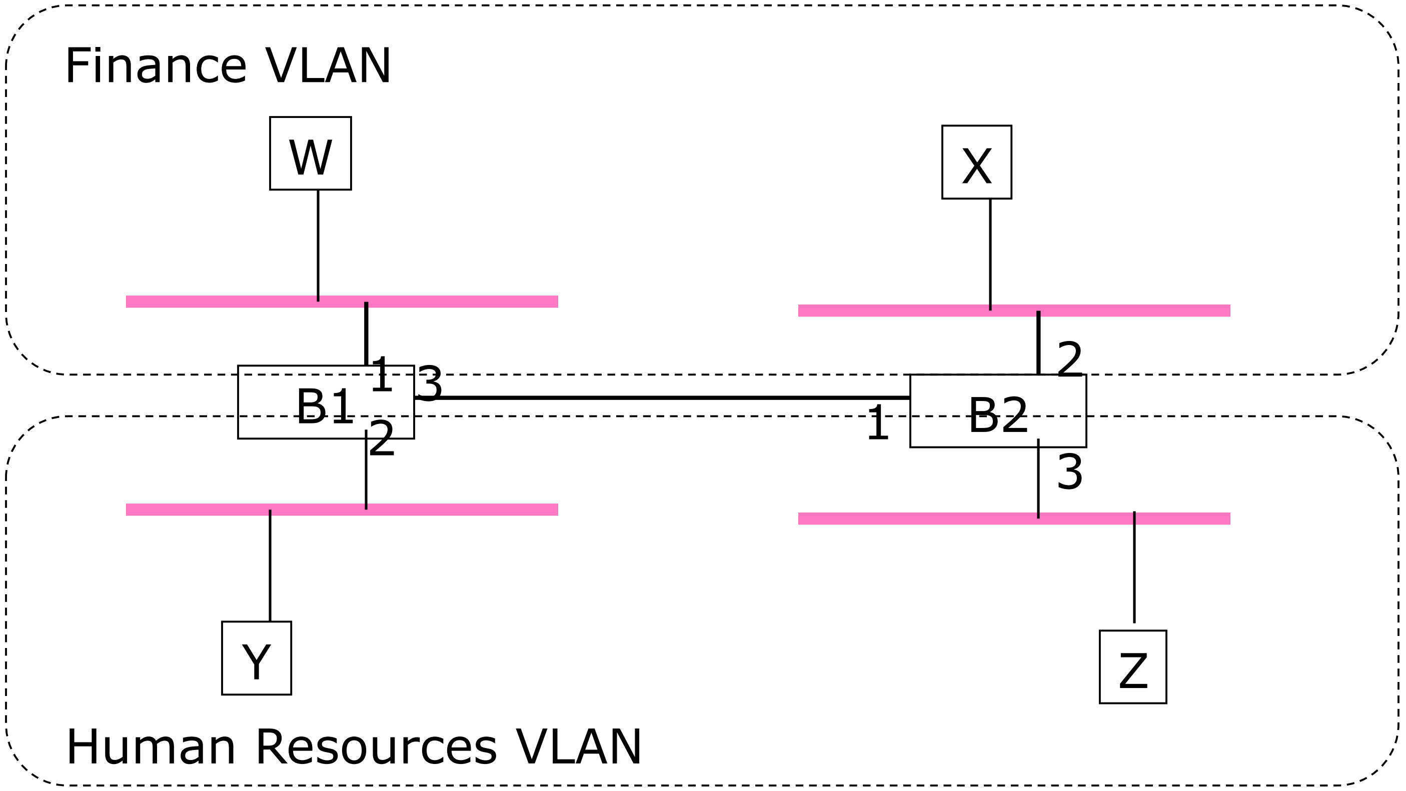




Change logical topology without moving any wires or change addresses

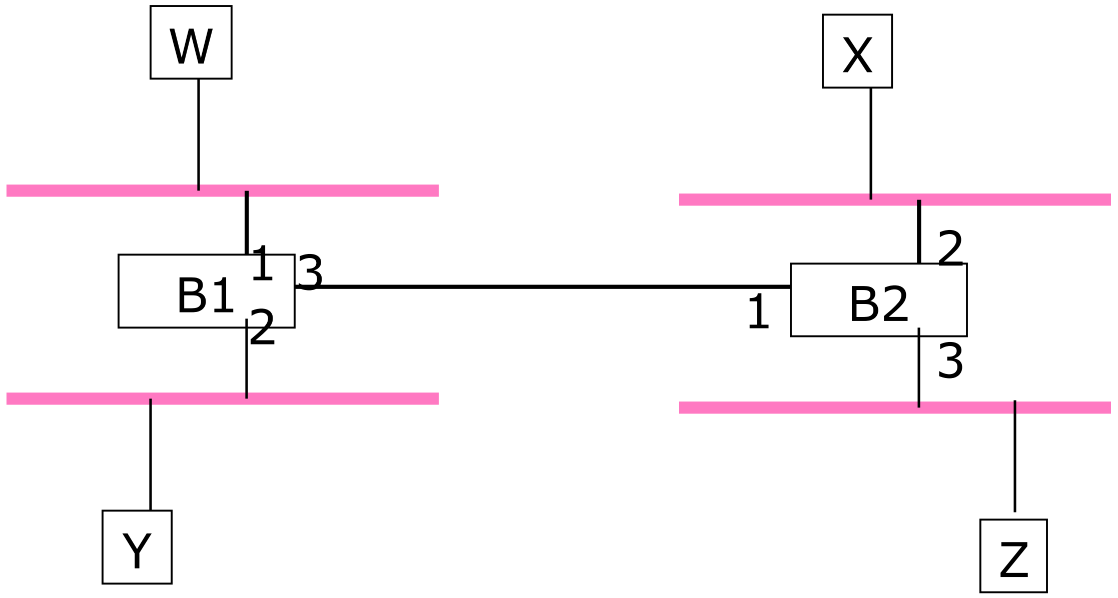
Logical Separation

A frame can travel only within a segment (VLAN)



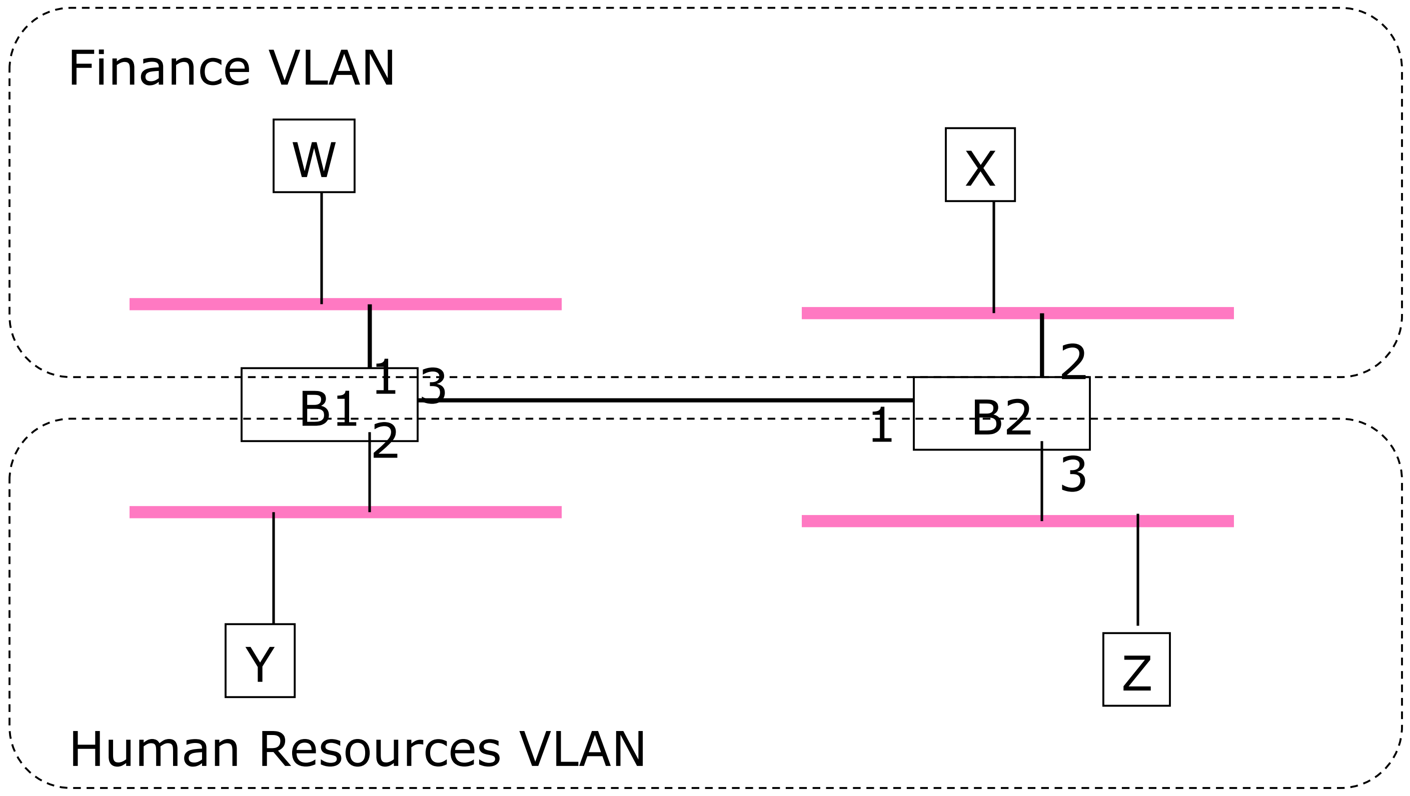
Without VLAN

A frame can travel anywhere within the extended LAN



Logical Separation with VLAN

A frame can travel only within a segment (VLAN)

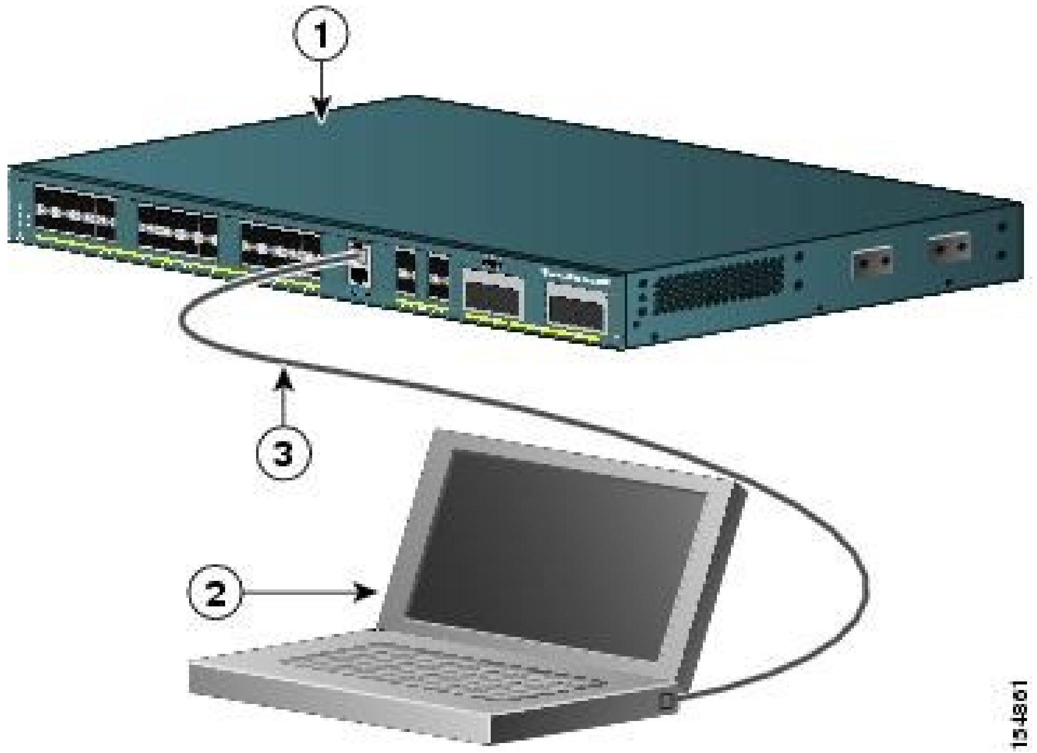


Virtual LAN

- Each VLAN is assigned an ID (or color)
- A frame can travel only within a segment (VLAN) with the same identifier → controlled by forwarding algorithms in switches
  - Configure switches B1 and B2
    - Assign B1.1, B1.3, B2.1, B2.1 to VLAN 100
    - Assign B1.2, B1.3, B2.1, B2.3 to VLAN 200

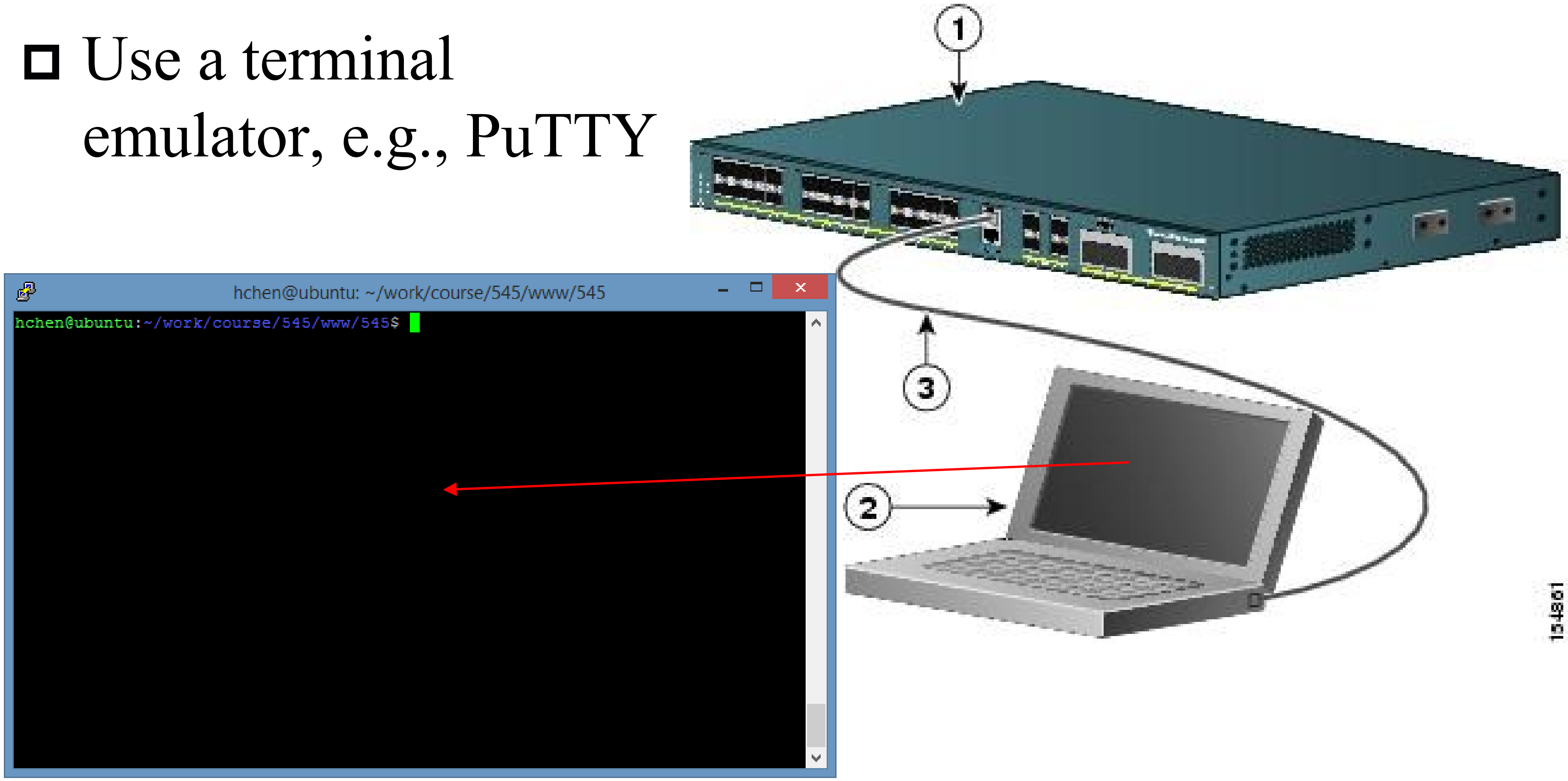
Virtual LAN: Configuration Example

1. Cisco Switch
2. Computer
3. RJ-45-to-DB-9 adapter cable



Virtual LAN: Configuration Example

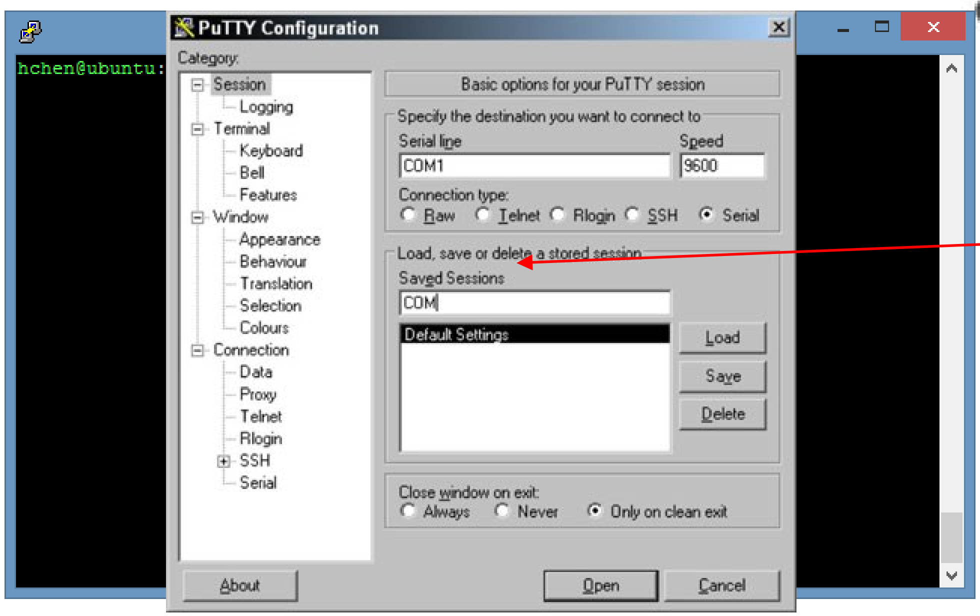
- Use a terminal emulator, e.g., PuTTY



```
hchen@ubuntu:~/work/course/545/www/545$
```

Virtual LAN: Configuration Example

- Use PuTTY
- Choose Serial Connection



hchen@ubuntu:

PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM1

Speed: 9600

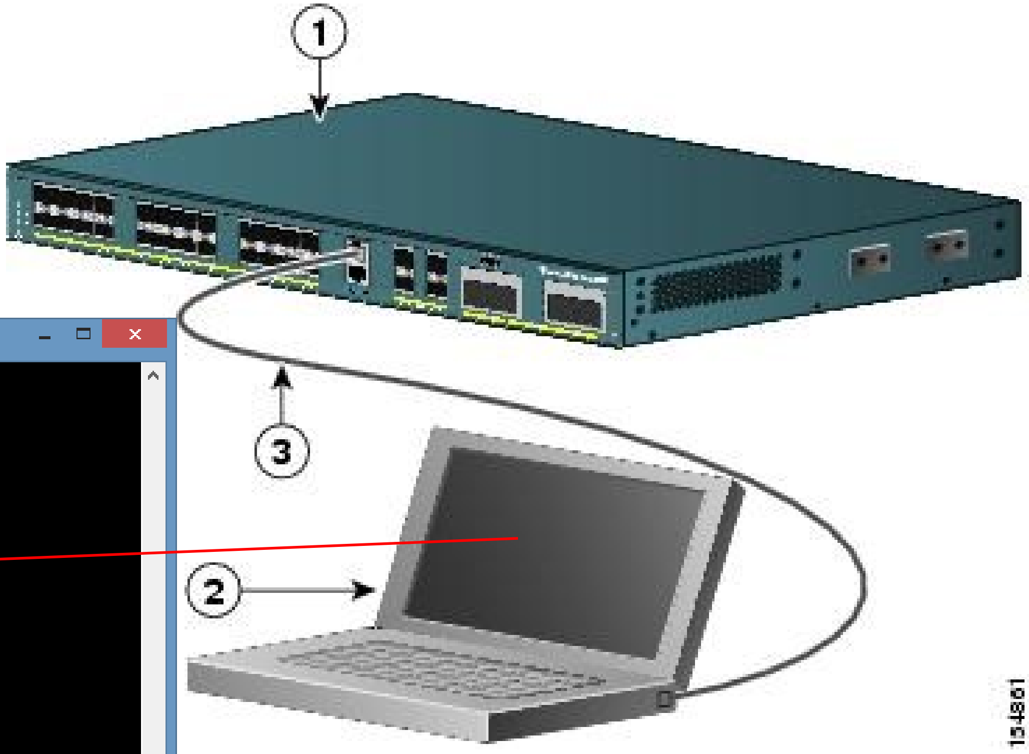
Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Load, save or delete a stored session

Saved Sessions: COM

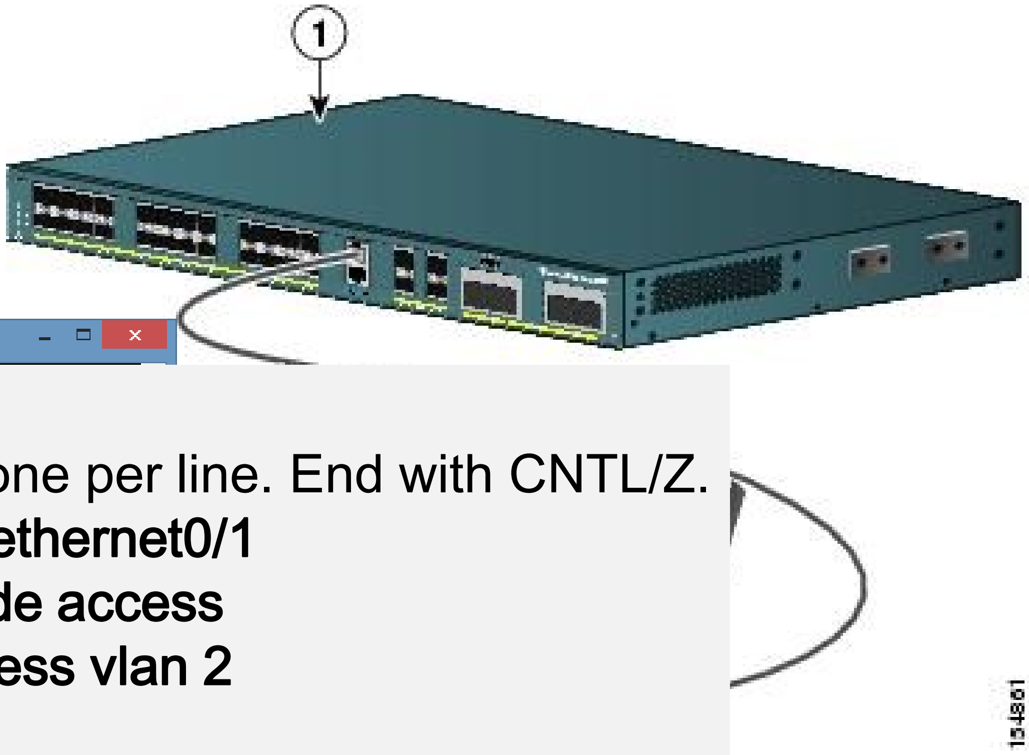
Default Settings

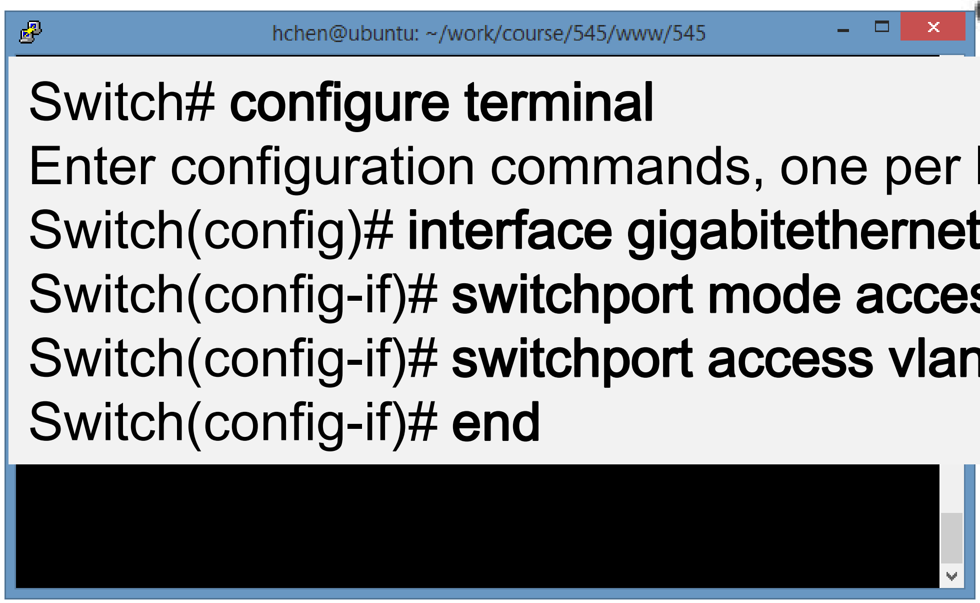
Close window on exit: Always Never [✓ Only on clean exit]



Virtual LAN: Configuration Example

- Use PuTTY
- Choose Serial Connection

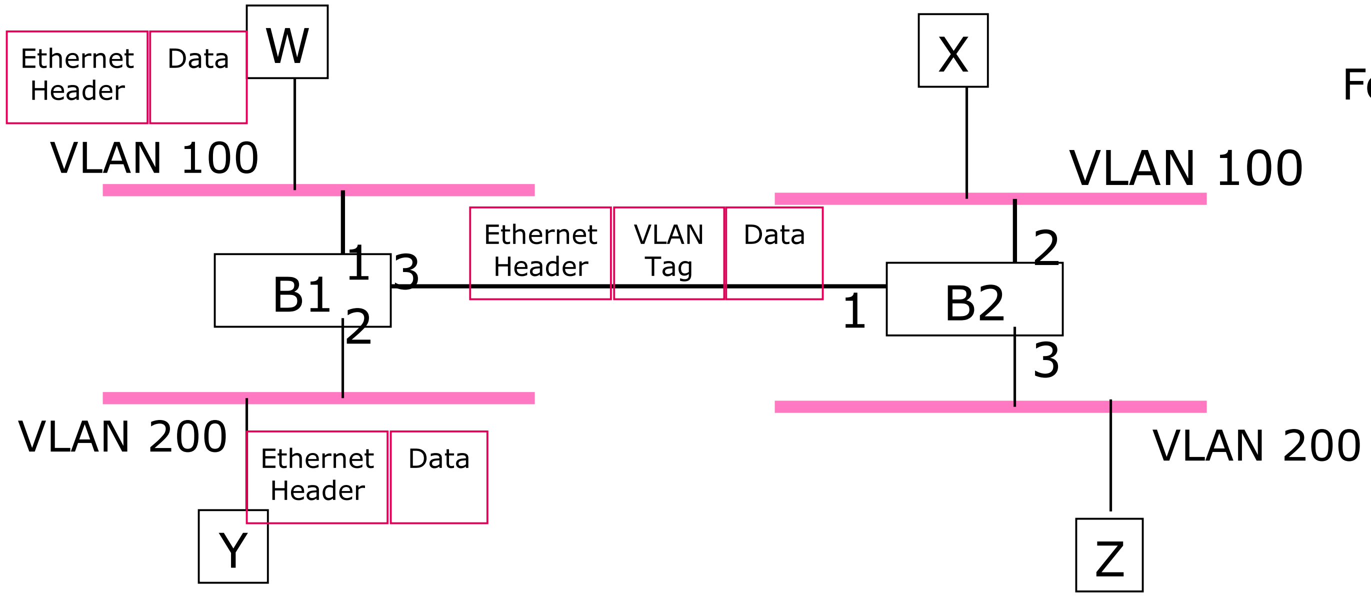




```
Switch# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)# interface gigabitethernet0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 2
Switch(config-if)# end
```

Virtual LAN

- Each VLAN is assigned an ID (or color)
- A frame can travel only within a segment (VLAN) with the same identifier → controlled by forwarding algorithms in switches
- IEEE 802.1Q:
  - VLAN Tag (4 bytes = 32 bits): 0x8100 + … (4bits) … + VLAN ID (12 bits)



Forwarding rule:
Frame may not be sent out an interface that is not part of VLAN indicated by the VLAN ID

VLAN Management In Practice

- Configuring VLAN and VLAN Trunk on Cisco Catalyst Switches
  - Cisco’s proprietary VLAN Trunk Protocol
    - For propagating VLAN information among multiple VLANs with a VTP domain
  - Creating VLAN and VLAN trunks
    - Define VTP domains
    - Create VLANs
    - Add and remove ports to a VLAN
    - Troubleshooting

Managing Spanning Tree Protocol in Practice

- Configure Spanning Tree Protocol (STP) on Cisco Catalyst Switches
  - Show current STP configuration
  - Change STP configuration
  - Disable and enable STP for a VLAN

Summary

- Switches → scalable networks
- Packet switching
  - Datagram switching
  - Virtual circuit switching
  - Source routing
- Datagram switching in practice
  - Ethernet
    - Bridges as LAN Switches
    - How to apply datagram switching to extended LANs?
    - Learning bridges: forward or not to forward?
    - Spanning Tree Algorithm: break loops
- *Virtual Local Area Networks (VLANs)*
  - *We will address it in late classes*
- *Q: Different networks are built, extended LANs do not scale well, how to expand networks?*
  - *Inter-networking*