CMSC417 Spring 2016. Lecture #4 2/8/2016

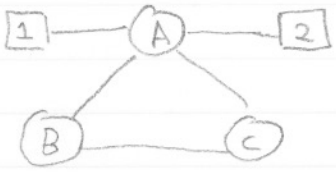
Agenda:

⇒ Project 0 grades (13 didn't turn it in, 22 got all 7)
⇒ Drop/Add period
⇒ Spanning Tree
⇒ What limitations do we have?
⇒ Staying safe
  - ports default to off?
  - turn them on only after becoming path to root?

⇒ traceroute

CMSC 417 Spring 2016 Lecture #4 2/8/2016

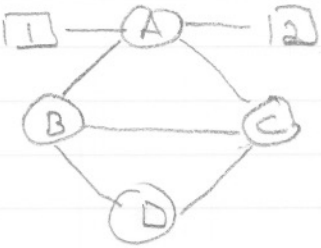
Spanning Tree Protocol



if A, B, C are learning bridges from last time, 1 sending a message to 2 will cause two frames to loop around until A learns 2's location [BAD :(]

if instead 1 sends a message to a non-existent 3, the frames will loop forever [WORSE :(]

if we have 2 loops: the frames will multiply [CATASTROPHIC :(]



Root Problem

loops are bad!

Solution

⇒ Do away with loops
- still need a connected graph
- finding a connected graph without loops is just the spanning tree problem
- look up your 351 notes

⇒ Two caveats
1. has to be distributed and everyone has to agree
2. can't even temporarily form loops!
   see example with two loops above

Spanning Tree Protocol

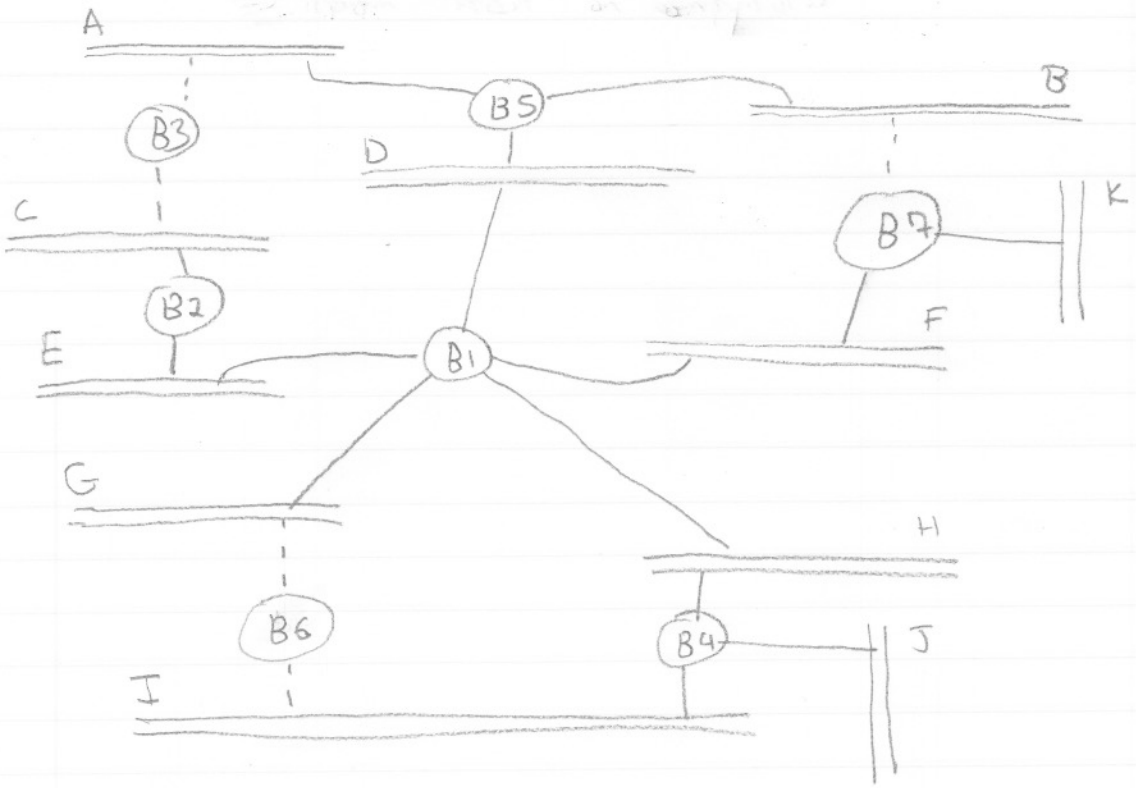
All messages are special, non-forwarded frames

0) when a bridge comes up turn all links off

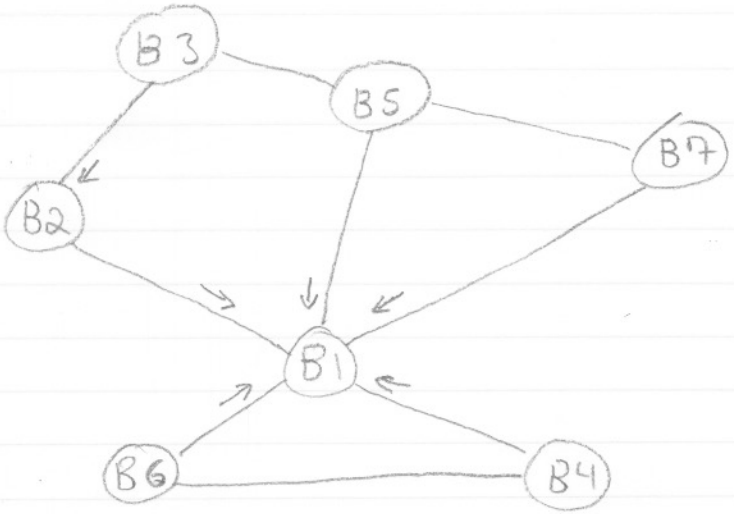
1) periodically each bridge sends a message
   (myID, distance-to-root, believed-root)
   starts with (myID, 0, myID)

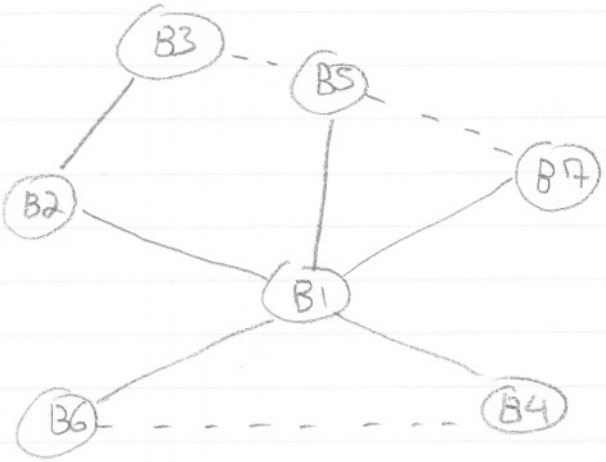
2) on recieving a messag update if the other message is "better"
   X is better than Y iff
   X.believed-root < Y.believed-root OR
   X.distance < Y.distance OR
   X.myID < Y.myID

3) on updating it's state, a node sends the message with the new state and distance+1



Spanning Tree Protocol





Spanning Tree Protocol

1) B3 recieves (B2, 0, B2) from B2
   ⇒ accepts B2 as the root
   ⇒ sends (B3, 1, B2) to B5
2) B5 recieves (B3, 1, B2) from B3
   ⇒ accepts B2 as the root
3) B3 receives (B2, 1, B1) from B
   ⇒ accepts B1 as the root
   ⇒ sends (B3, 2, B1) to B5
4) B5 recieves (B3, 2, B1) fom B3
   ⇒ accepts B1 as the root
5) B5 recieves (B1, 0, B1) from B1
   ⇒ accepts B1 as the root (via B1)
   ⇒ send (B5, 1, B1) to B3
6) B3 keeps B1 as root via B3 b/c B3