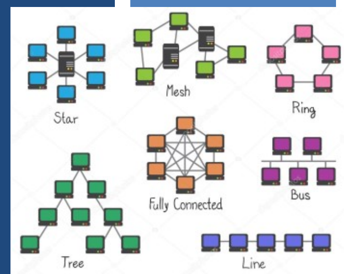
TCSS 558: APPLIED DISTRIBUTED COMPUTING



Chapter 4 – Communication–III

Chapter 6 - Coordination

Wes J. Lloyd
School of Engineering & Technology (SET)
University of Washington - Tacoma

1

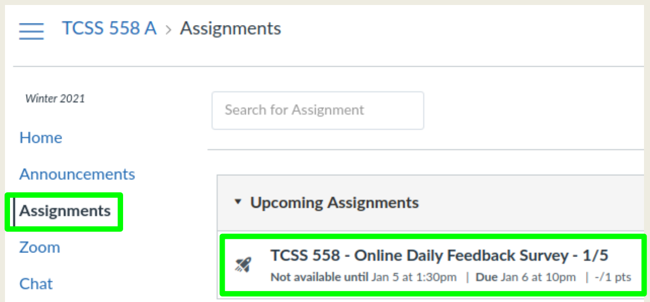
OBJECTIVES – 2/22

- Questions from 2/20
- Assignment 3: Replicated Key Value Store
- Chapter 4: Communication
  - Chapter 4.3: Message Oriented Communication
  - Chapter 4.4: Multicast Communication
- Chapter 6: Coordination
  - Chapter 6.1: Clock Synchronization

2

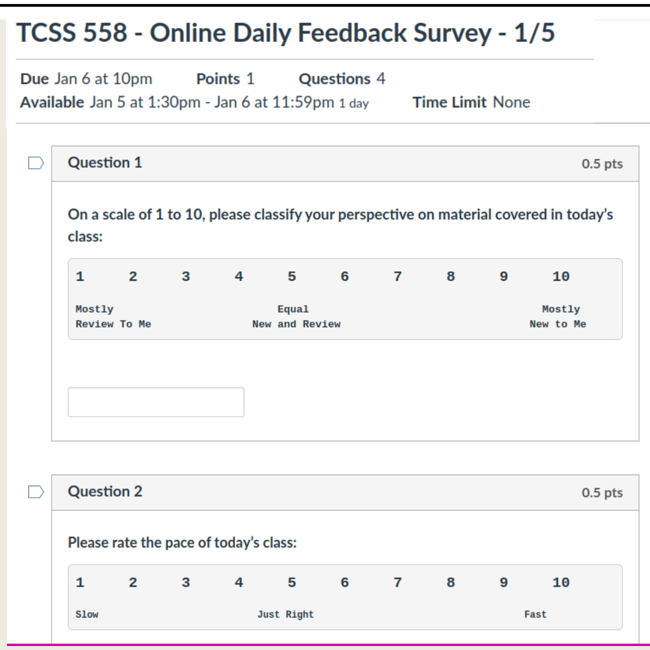
ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Available After Each Class
- Extra credit available for completing surveys ***ON TIME***
- Tuesday surveys: due by ~ Wed @ 10p
- Thursday surveys: due ~ Mon @ 10p



February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.3

3



February 22, 2024 — TCSS558: Applied Distributed Computing [Winter 2024]
School of Engineering and Technology, University of Washington - Tacoma — L14.4

4

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (25 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.60 (↑ - *previous 6.17*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.56 (↓ - *previous 5.75*)**

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.5

5

FEEDBACK FROM 2/20

- ***When considering an Overlay Network - where we have no information on the structure, we can consider the system as a "Random Graph" to support rationalization about the structure.***
- ***For the "Random Graph" rationalization, would we check each node for the unlikely event that it was assigned no neighbors and, if so, then assign it a neighbor?***
- NO. The Random Graphs here are not actual physical graphs. We are using graph theory to rationalize about the possible structure of the overlay network.
- It is worth noting, for a physical graph, a node with no neighbors is an orphan node, and is not a member of the network
- In a physical graph, each node should have at least one edge or else it is orphaned (not connected)
- ***Is there a check to make sure that all nodes are actually connected via some path?***
  - Our Random Graphs are not physical graphs, but rationalization to apply graph theory to compute probabilities about different properties

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.6

6

FEEDBACK - 2

- ***Does the number of edges in a network impact the probability of message spread(p_{flood})?***
- NO. It is the number of neighbors (n), not the number of edges that influences the probability of message spread (p_{flood}).
- For a network with 10,000 nodes, with a 10% probability of having an edge between every node, we calculated that there are nearly 5,000,000 edges that a message could be flooded on.
- With full message flooding, each node forwards the message m to each neighbor except the one from which it received m, where the node then tracks the messages it receives and forwards to not repeat sending→ Full flooding requires ~10,000,000 messages
- The idea with probabilistic flooding is to set a threshold to limit message spread. If we only flood on p_{flood}=.01 (1%) of the 5 million edges, then we only send 50,000 messages across 10,000 nodes but if a node say 'Q' has 298 neighbors, then it is 95% likely that Q will receive the message with p_{flood}=.01 !! (50-fold reduction)

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.7

7

FEEDBACK - 2

- Does the number of edges in a network impact the probability of message spread(p_{flood})?

What does it mean to have p_{flood} =.01? (1%)

If a node Q has n neighbors, the probability that all neighbors don't forward the message to Q is $p=(1-p_{flood})^n$

if n=10, $p=(1-.01)^{10}=.904$ (pretty likely)
if n=100, $p=(1-.01)^{100}=.366$ (less likely)
if n=298, $p=(1-.01)^{298}=$**.05 (unlikely)**

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.8

8

FEEDBACK - 3

- ***In Multi-Cast Tree, is it the case that cost of sending message over overlay network is always greater than cost of sending message over physical network?***
- The cost will always be equal to or higher than when using the physical network
- ***If so what is the purpose of the overlay network?***
- The purpose is for **Application level multi-casting (broadcast)**
  - Nodes organize into an overlay network
  - KEY→ Network routers ***not*** involved in group membership
  - KEY → Group membership is managed at the **application level** (A2)
- The disadvantage:
  - Application-level routing likely less efficient than network-level
  - Necessary tradeoff until we have better multicasting protocols defined at lower layers in the OSI model

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.9

9

FEEDBACK - 4

- ***What is an example of a realistic message where it would be okay to send to 95% of nodes in a network (probabilistic flooding).***
- p_{flood}=.95 is actually **very high!!**
- Nodes will need very few neighbors to ensure message delivery (saturation)
- PROBLEM:
  - p_{flood}=.95 is very close to p_{flood}=1.00
  - For a network with 5,000,000 edges, p_{flood}=1.00 is 10 million msgs
  - p_{flood}=.95 will be about 4.75 million msgs, which is about half the number of total messages…
- ***Thinking about it I guess it could be a message like "flush to disk" or "check in with heartbeat server" or something like that. I just haven't heard of something like this used before.***
  - There are a variety of broadcast messages possible used to notify nodes about various events and state changes across the system

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.10

10

CSS TENURE TRACK FACULTY CANDIDATE RESEARCH SEMINARS – EXTRA CREDIT

- Week 8:
  - Thursday February 22 – 12:30pm –MLG 110
  - Friday February 23 – 12:30pm –MLG 301
- Week 9:
  - Monday February 26 – 12:30pm – MLG 110
  - Wednesday February 28 – 1:30pm – JOY 117
  - Thursday February 29 – 1:30pm – MLG 110
  - Friday March 1 – 1:30pm – MLG 301
- Week 10:
  - Monday March 4 – 1:30pm - MLG 110
  - Tuesday March 5 – 1:30pm - CP 324
  - Thursday March 7 – 12:30pm – MLG 110
- Earn up to 30 buffer points added to the Final Exam score
- Earn 3 points for each seminar attended
- Buffer points replace missed points on the Final Exam
- Once the Final Exam score = 100%, additional points do not push the Final Exam score above 100%
- Buffer points will not impact the course curve for the Final Exam
- Any course curve will be applied before buffer points

February 20, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L13.11

11

ASSIGNMENT 2 → DUE FEB 24

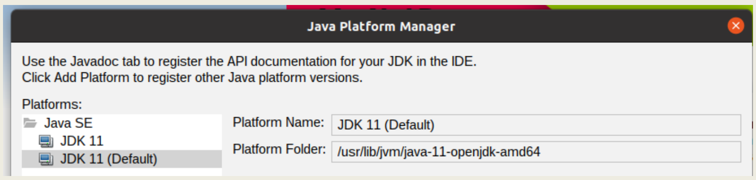
- Find Teammates: signup posted on Canvas under 'People'
- GenericNode.tar.gz includes Dockerfile examples
- GenericNode.tar.gz assumes Java 11
- TCP/UDP/RMI Key Value Store
- Implement a "GenericNode" project which assumes the role of a client or server for a Key/Value Store
- Recommended in Java 11 LTS
- Client node program interacts with server node to put, get, delete, or list items in a key/value store

February 20, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L13.12

12

USING JAVA 11 IN NETBEANS

- In Netbeans IDE, under Tools menu, 'Java Platforms', be sure to install and select JDK 11



- On left-hand Project menu, right-click on '**GenericNode**' project
- Select Properties
- Under Build | Compile, be sure Java Platform is JDK 11
- Under Sources, be sure Source/Binary Format is 11

February 20, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L13.13

13

OBJECTIVES – 2/22

- Questions from 2/20
- **Assignment 3: Replicated Key Value Store**
- Chapter 4: Communication
  - Chapter 4.3: Message Oriented Communication
  - Chapter 4.4: Multicast Communication
- Chapter 6: Coordination
  - Chapter 6.1: Clock Synchronization

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.14

14

ASSIGNMENT 2

- **Sunday March 10th**
- Goal: Replicated Key Value Store
- Team signup to be posted on Canvas under 'People'
- Build off of Assignment 2 GenericNode
- Focus on TCP client/server w/ replication
- How to track membership for data replication?
  - Can implement multiple types of membership tracking for extra credit
- REQUIREMENT: 'store' command needs to output 1 key-value pair per line using ASCII text (no binary)

15

SHORT-HAND-CODES FOR MEMBERSHIP TRACKING APPROACHES

- Include readme.txt or doc file with instructions in submission
- Must document membership tracking method

>> please indicate which types to test <<

ID	Description
F	Static file membership tracking – file is not reread
FD	Static file membership tracking DYNAMIC - file is periodically reread to refresh membership list
T	TCP membership tracking – servers are configured to refer to central membership server
U	UDP membership tracking - automatically discovers nodes with no configuration

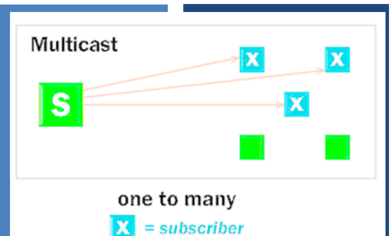
16

OBJECTIVES – 2/22

- Questions from 2/20
- Assignment 3: Replicated Key Value Store
- Chapter 4: Communication
  - Chapter 4.3: Message Oriented Communication
  - **Chapter 4.4: Multicast Communication**
- Chapter 6: Coordination
  - Chapter 6.1: Clock Synchronization

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.17

17



Apache ActiveMQ

CH. 4.4: MULTICAST COMMUNICATION

L14.18

18

CHAPTER 4

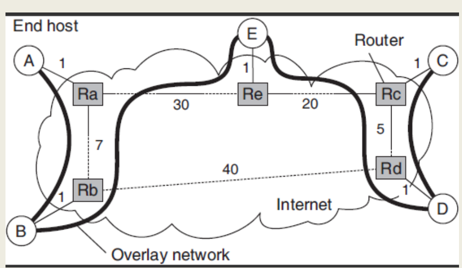
- 4.1 Foundations
  - Protocols
  - Types of communication
- 4.2 Remote procedure call
- 4.3 Message-oriented communication
  - Socket communication
  - Messaging libraries
  - Message-Passing Interface (MPI)
  - Message-queueing systems
  - Examples
- 4.4 Multicast communication
  - Flooding-based multicasting
  - Gossip-based data dissemination

**These sections feature many details,
Our focus is on the "big picture"**

19

FLOOD-BASED MULTICASTING

- Broadcasting: every node in overlay network receives message



- How many nodes are in the overlay network?
- How many nodes are in the underlying network?

20

FLOOD-BASED MULTICASTING

- Broadcasting: every node in overlay network receives message
- Key design issue: minimize the use of intermediate nodes for which the message is not intended
- If only leaf nodes are to receive the multicast message, many intermediate nodes are involved in ***storing*** and ***forwarding*** the message *not meant for them*
- Solution: construct an overlay network for each multicast group
  - Sending a message to the group, becomes the same as broadcasting to the multicast group (*group of nodes that listen and receive traffic for a shared IP address*)
- **Flooding**: each node simply forwards a message to each of its neighbors, except to the message originator

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.21

21

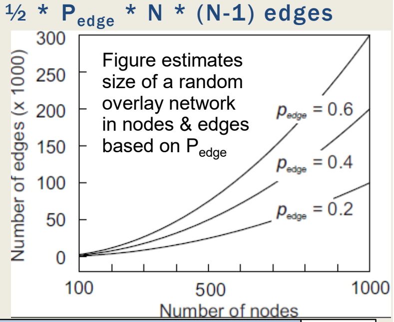
RANDOM GRAPHS

- Used when no information on the structure of the overlay network
- Assume network can be represented as a **Random graph**
- Random graphs are described by a probability distribution:

1. Given a probability P_{edge} that two nodes are joined
2. Size of a random overlay network is: $\frac{1}{2} * P_{edge} * N * (N-1)$ edges

Random graphs allow us to assume some structure (# of nodes, # of edges) regarding the network by scaling the P_{edge} probability

Assumptions may help then to reason or rationalize about the network…



February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.22

22

PROBABILISTIC FLOODING


Washington state in winter?

- When a node is flooding a message m: (p_{flood}) is the probability that the message is spread to a specific neighbor =(p_{flood})
- Throttle message flooding based on a probability
- Implementation needs to considers # of neighbors to achieve various p_{flood} scores
- With lower p_{flood} messages may not reach all nodes

- **Efficiency of probabilistic broadcasting:** For random network with 10,000 nodes
- With p_{edge} = 0.1 and p_{flood} =.01
- Achieves 50-fold reduction in messages vs. full flooding

23

PROBABILISTIC FLOODING


Washington state in winter?

- When a node is flooding a message m: (p_{flood}) is the probability that the message is spread to a specific neighbor =(p_{flood})
- Thrott
- Impler ... to achiev
- With l ... s

How many edges does network with 10,000 nodes have with p_{edge}=0.1?

- **Efficiency of probabilistic broadcasting:** For random network with 10,000 nodes
- With p_{edge} = 0.1 and p_{flood} =.01
- Achieves 50-fold reduction in messages vs. full flooding

24

PROBABILISTIC FLOODING


Washington state in winter?

- When a node is flooding a message m: (p_{flood}) is the probability that the message is spread to a specific neighbor $=(p_{flood})$
- Thrott
- Impler ... to achiev
- With l ... s
- **Efficie** ... m netwo
- With p_{edge} = 0.1 and p_{flood} =.01
- Achieves 50-fold reduction in messages vs. full flooding

How many edges does network with 10,000 nodes have with p_{edge}=0.1?

Edges = $\frac{1}{2} * P_{edge} * N * (N-1)$

25

PROBABILISTIC FLOODING


Washington state in winter?

- When a node is flooding a message m: (p_{flood}) is the probability that the message is spread to a speci
- Throt
- Imple ... to achie
- With
- **Effici** ... n network with 10,000 nodes
- With p_{edge} = 0.1 and p_{flood} =.01
- Achieves 50-fold reduction in messages vs. full flooding

How many edges does network with 10,000 nodes have with p_{edge}=0.1?

Edges = $\frac{1}{2} * P_{edge} * N * (N-1)$
$\frac{1}{2} * (.1) * (10000) * (9999)$

26

PROBABILISTIC FLOODING

Washington state in winter?

- When a node is flooding a message m:
  (p_{flood}) is the probability that the message is spread to a speci…
- Throt…
- Imple… to achie…
- With …
- Effici… network with 10,000 nodes
- With p_{edge} = 0.1 and p_{flood} =.01
- Achieves 50-fold reduction in messages vs. full flooding

How many edges does network with 10,000 nodes have with p_{edge}=0.1?

Edges = ½ * P_{edge} * N * (N-1)
½ * (.1) * (10000) * (9999)
4,999,500 edges

February 22, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L14.27

27

PROBABILISTIC FLOODING

Washington state in winter?

- When a node is flooding a message m:
  (p_{flood}) is the probability that the message is spread to a specific neighbor =(p_{flood})
- Throt…
- Imple… achie…
- With …
- Effici… netw…
- With p_{edge} = 0.1 and p_{flood} =.01
- Achieves 50-fold reduction in messages vs. full flooding

What does it mean to have p_{flood} =.01?

With lower p_{flood} messages may not reach all nodes

If there are 10,000 nodes and ~5 million edges, how many messages is full flooding?
> every node sends to all neighbors across all edges
> will be ~10,000,000 messages

February 22, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L14.28

28

PROBABILISTIC FLOODING


Washington state in winter?

- When a node is flooding a message m:
  (p_{flood}) is the probability that the message is spread to a specific neighbor =(p_{flood})

What does it mean to have p_{flood} =.01?

node Q has n neighbors
Probability that ***all*** neighbors don't forward message to Q is $p=(1-p_{flood})^n$

network with 10,000 nodes

- With p_{edge} = 0.1 and p_{flood} =.01
- Achieves 50-fold reduction in messages vs. full flooding

29

PROBABILISTIC FLOODING


Washington state in winter?

What does it mean to have p_{flood} =.01?

node Q has n neighbors
Probability that <u>***no*** neighbors of Q forward the message to Q:</u>
$p=(1-p_{flood})^n$ ← *probability of Q not getting the message*

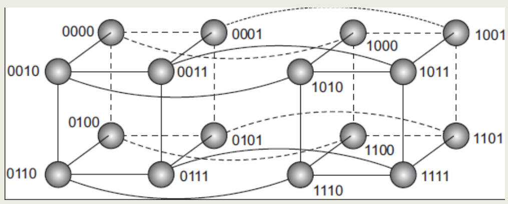
10 nodes: if n=10, $p=(1-.01)^{10}$=.904 (small network, few edges, likely Q doesn't get msg)
100 nodes: if n=100, $p=(1-.01)^{100}$=.366 (less likely Q doesn't get msg)
298 nodes: if n=298, $p=(1-.01)^{298}$=.05 (Q probably gets msg)

- Achieves 50-fold reduction in messages vs. full flooding

30

MESSAGE FLOODING

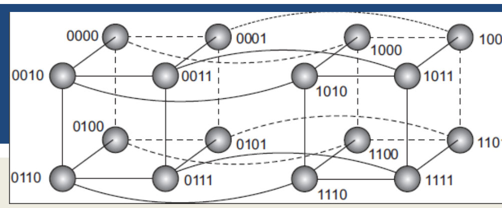
- For deterministic topologies (such as hypercube), design of efficient flooding scheme is much simpler
- If the overlay network is structured, this gives us a deterministic topology
- Schlosser et al [2002] – offer simple and efficient broadcasting scheme that relies on keeping track of neighbors per dimension



February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.31

31

MESSAGE FLOODING - 2



- <u>**Hypercube Broadcast**</u>
- N(1001) starts the network broadcast
- N(1001) neighbors {0001,1000,1011,1101}
- N(1001) Sends message to all neighbors
- <u>**>>Edge Labels *(which bit is changed?, 1st, 2nd, 3rd, 4th…)***</u>
- Edge to 0001 – labeled 1 – change the 1st bit
- Edge to 1000 – labeled 4 – change the 4th bit
- Edge to 1011 – labeled 3 – change the 3rd bit
- Edge to 1101 – labeled 2 – change the 2nd bit

- <u>**RULE: nodes only forward along edges with a higher dimension**</u>
- Node 1101 receives message on edge labeled 2
- Broadcast msg is only forwarded on <u>***higher***</u> valued edges (>2)

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.32

32

MESSAGE FLOODING - 3

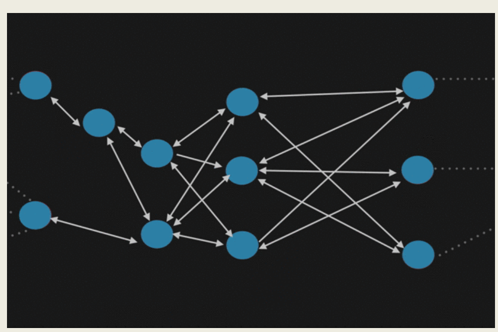
- **Hypercube:** forward msg along edges with higher dimension
- Node(1101)–neighbors {0101,1100,1001,1111}
- Node (1101) - incoming broadcast edge = 2
- Label Edges:
- Edge to 0101 – labeled 1 – change the 1st bit
- Edge to 1100 – labeled 4 – change the 4th bit ***<FORWARD>***
- Edge to 1001 – labeled 2 – change the 2nd bit
- Edge to 1111 – labeled 3 – change the 3rd bit ***<FORWARD>***
- N(1101) broadcast – forward only to N(1100) and N(1111)
- (1100) and (1111) are the ***higher dimension edges***
- Broadcast requires just: N-1 messages, where nodes $N=2^n$, n=dimensions of hypercube

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.33

33

GOSSIP BASED DATA DISSEMINATION

- When structured peer-to-peer topologies are not available
- Gossip based approaches support multicast communication over unstructured peer-to-peer networks
- General approach is to leverage how gossip spreads across a group
- This is also called "epidemic behavior"…
- Data updates for a specific item begin at a specific node



February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.34

34

INFORMATION DISSEMINATION

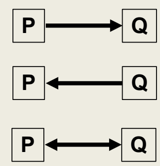
- **Epidemic algorithms**: algorithms for large-scale distributed systems that spread information
- Goal: "infect" all nodes with new information as fast as possible
- **Infected**: node with data that can spread to other nodes
- **Susceptible**: node without data
- **Removed**: node with data that is unable to spread data

EPIDEMIC PROTOCOLS

- **Gossiping**
- Nodes are randomly selected
- One node, randomly selects any other node in the network to propagate the network
- Complete set of nodes is known to each member

ANTI ENTROPY DISSEMINATION MODEL FOR GOSSIPING

- **Anti-entropy:** Propagation model where node P picks node Q at random and exchanges message updates
- Akin to random walk
- **Types of message exchange:**
- **PUSH:** P only **pushes** its own updates to Q
- **PULL:** P only **pulls** in new updates from Q
- **TWO-WAY:** P and Q send updates to each other (i.e. a push-pull approach)



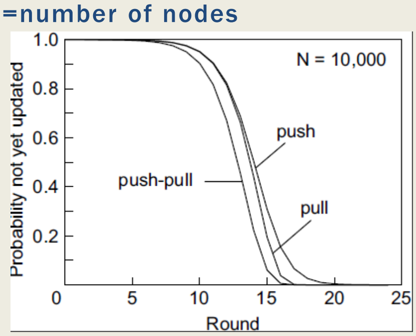
- Push only: hard to propagate updates to last few hidden susceptible nodes
- Pull: better because susceptible nodes can pull updates from infected nodes
- Push-pull is better still

37

ANTI ENTROPY EFFECTIVENESS

- **Round**: span of time during which every node takes initiative to exchange updates with a randomly chosen node
- The number of rounds to propagate a single update to all nodes requires O(log(N)), where N=number of nodes
- Let p_i denote probability that node P has not received msg m after the i^{th} round.
- For pull, push, and push-pull based approaches:

10,000 nodes →



38

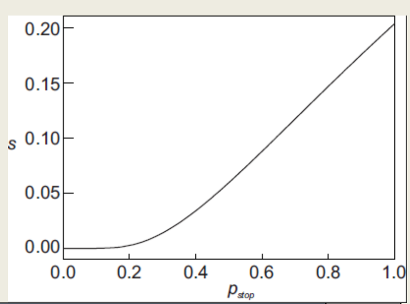
RUMOR SPREADING

- Variant of epidemic protocols
- Provides an approach to **"stop"** message spreading
- Mimics "gossiping" in real life

- **Rumor spreading:**
- **Node P** receives new data **item X**
- Contacts an arbitrary **node Q** to push update
- **Node Q** reports already receiving **item X** from another node
- **Node P** may loose interest in spreading the rumor with probability = p_{stop}, let's say 20% . . . (or 0.20)

RUMOR SPREADING - 2

- p_{stop}, is the probability node will stop spreading once contacting a node that already has the message
- Does not guarantee all nodes will be updated
- The fraction of nodes s, that remain susceptible grows relative to the probability that node P stops propagating when finding a node already having the message
- Fraction of nodes not updated remains < 0.20 with high p_{stop}
- Susceptible nodes (s) vs. probability of stopping →



REMOVING DATA

- Gossiping is good for spreading data
- **But how can data be removed from the system?**
- Idea is to issue ***"death certificates"***
- Act like data records, which are spread like data
- When death certificate is received, data is deleted
- Certificate is held to prevent data element from reinitializing from gossip from other nodes
- Death certificates time-out after expected time required for data element to clear out of entire system
- A few nodes maintain death certificates forever

DEATH CERTIFICATE EXAMPLE

- **For example:**
- **Node P** keeps death certificates forever
- **Item X** is removed from the system
- **Node P** receives an update request for **Item X**, but ***also*** holds the death certificate for **Item X**
- **Node P** will recirculate the death certificate across the network for **Item X**

WE WILL RETURN AT 4:55 PM

43

OBJECTIVES – 2/22

- Questions from 2/20
- Assignment 3: Replicated Key Value Store
- Chapter 4: Communication
  - Chapter 4.3: Message Oriented Communication
  - Chapter 4.4: Multicast Communication
- **Chapter 6: Coordination**
  - Chapter 6.1: Clock Synchronization

44

CHAPTER 6 - COORDINATION

- 6.1 Clock Synchronization
  - Physical clocks
  - Clock synchronization algorithms
- 6.2 Logical clocks
  - Lamport clocks
  - Vector clocks
- 6.3 Mutual exclusion
- 6.4 Election algorithms
- 6.6 Distributed event matching *(light)*
- 6.7 Gossip-based coordination *(light)*

CHAPTER 6 - COORDINATION

- How can processes synchronize and coordinate data?
- Process synchronization
  - Coordinate cooperation to grant individual processes temporary access to shared resources (e.g. a file)
- Data synchronization
  - Ensure two sets of data are the same (data replication)
- Coordination
  - Goal is to manage interactions and dependencies between activities in the distributed system
  - Encapsulates synchronization

COORDINATION - 2

- Synchronization challenges begin with **time**:
  - How can we synchronize computers, so they all agree on the time?
  - How do we measure and coordinate when things happen?
- Fortunately, for synchronization in distributed systems, it is often sufficient to only agree on a relative ordering of events
  - E.g. not actual time

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.47

47

COORDINATION - 3

- Groups of processes often appoint a **coordinator**
- **Election algorithms** can help elect a leader
- Synchronizing access to a shared resource is achieved with **distributed mutual exclusion** algorithms
- Also in chapter 6:
  - Matching subscriptions to publications in publish-subscribe systems
  - Gossip-based coordination problems:
    - Aggregation
    - Peer sampling
    - Overlay construction

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.48

48

OBJECTIVES – 2/22

- Questions from 2/20
- Assignment 3: Replicated Key Value Store
- Chapter 4: Communication
  - Chapter 4.3: Message Oriented Communication
  - Chapter 4.4: Multicast Communication
- Chapter 6: Coordination
  - **Chapter 6.1: Clock Synchronization**

49

CH. 6.1: CLOCK SYNCHRONIZATION

50

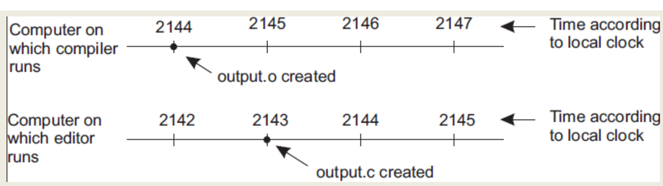
CLOCK SYNCHRONIZATION

- Example:
- "make" is used to compile source files into binary object and executable files
- As an optimization, make only compiles files when the "last modified time" of source files is more recent than object and executables
- Consider if files are on a shared disk of a distributed system where there is no agreement on time
- Consider if the program has 1,000 source files

February 22, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L14.51

51

TIME SYNCHRONIZATION PROBLEM FOR DISTRIBUTED SYSTEMS

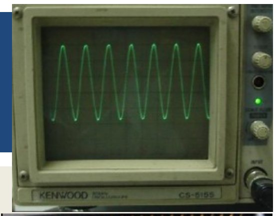


- Updates from different machines, may have clocks set to different times
- Make becomes confused with which files to recompile

February 22, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L14.52

52

PHYSICAL CLOCKS



- **Computer timers:** precisely machined quartz crystals
- When under tension, they oscillate at a well defined frequency
- In analog electronics/communications crystals once used to set the frequency of two-way radio transceivers for
- Today, crystals are associated with a counter and holding register on a digital computer.

1960s ERA radio crystal →

- Each oscillation decrements a counter by one
- When counter gets to zero, an interrupt fires
- Can program timer to generate interrupt, let's say 60 times a second, or another frequency to track time

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.53

53

COMPUTER CLOCKS

- Digital clock on computer sets base time
- Crystal clock tracks forward progress of time
  - Translation of wave "ticks" to clock pulses
- CMOS battery on motherboard maintains clock on power loss
- **Clock skew**: physical clock crystals are not exactly the same
- Some run at slightly different rates
- Time differences accumulate as clocks drift forward or backward slightly
- In an automobile, where there is no clock synchronization, clock skew may become noticeable over months, years

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.54

54

UNIVERSAL COORDINATED TIME

```
ubuntu@ip-172-31-58-89:~$ date
Thu Nov 16 10:13:39 UTC 2017
```

- **Universal Coordinated Time (UTC)**
  - Worldwide standard for time keeping
  - Equivalent to Greenwich Mean Time (United Kingdom)
  - 40 shortwave radio stations around the world broadcast a short pulse at the start of each second (WWV)
  - World wide "atomic" clocks powered by constant transitions of the non-radioactive caesium-133 atom
    - 9,162,631,770 transitions per second
- Computers track time using UTC as a base
  - Avoid thinking in local time, which can lead to coordination issues
  - Operating systems may translate to show local time

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.55

55

COMPUTING: CLOCK CHALLENGES

- How do we synchronize computer clocks with real-world clocks?
- How do we synchronize computer clocks with each other?

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.56

56

CLOCK SYNCHRONIZATION

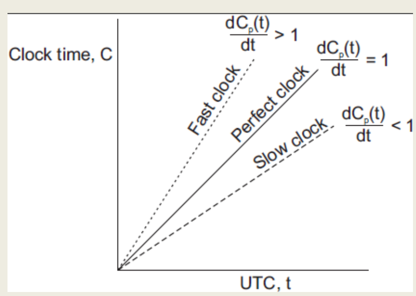
- **UTC services**: use radio and satellite signals to provide time accuracy to 50ns
- **Time servers**: Server computers with UTC receivers that provide accurate time
- **Precision** (π): how close together a set of clocks may be
- **Accuracy**: how correct to actual time clocks may be
- **Internal synchronization**: Sync local computer clocks
- **External synchronization**: Sync to UTC clocks
- **Clock drift**: clocks on different machines gradually become out of sync due to crystal imperfections, temperature differences, etc.
- **Clock drift rate**: typical is 31.5s per year
- **Maximum clock drift rate** (ρ): clock specifications include one

February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.57

57

CLOCK SYNCHRONIZATION - 2

- If two clocks drift from UTC in opposite directions, after time Δt after synchronization, they may be 2ρ apart.
  - ρ - clock drift rate, π - clock precision (max 50ns)
- Clocks must be resynchronized every $\pi/2\rho$ seconds
- **Network time protocol**
- Provide coordination of time for servers
- Leverage distributed network of time servers

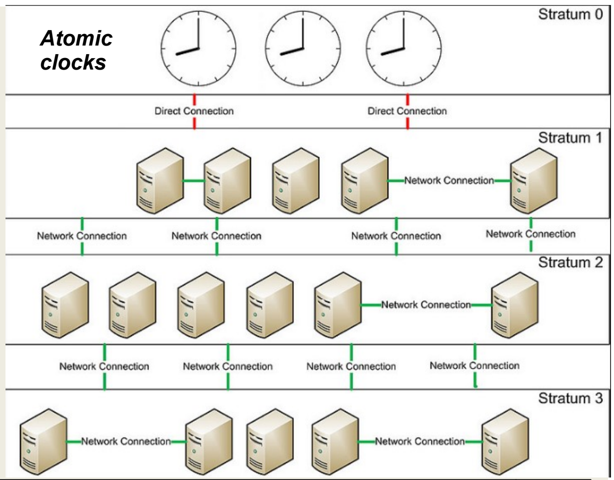


February 22, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L14.58

58

NETWORK TIME PROTOCOL

- Servers organized into stratums
- Stratum-1 servers have UTC receivers and are sync'd with atomic clocks
- Servers connect with closest NTP server for time synchronization
- Servers assume role as NTP server at stratum+1



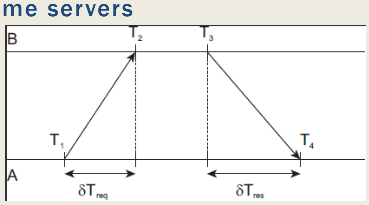
59

NTP - 2

- Must estimate network delays when synchronizing with remote UTC receiver clocks / time servers

Time server B

Client A



1. A sends message to B, with timestamp T1
2. B records time of receipt T2 (from local clock)
3. B returns response with send time T3, and receipt time T2
4. A records arrival of T4

- Assuming propagation delay of A→B→A is the same
- Estimate propagation delay: $\theta = T_3 + \frac{(T_2 - T_1) + (T_4 - T_3)}{2} - T_4 = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$
- Add delay to time

60

NTP - 3

- Cannot set clocks backwards (recall "make" file example)
- Instead, temporarily slow the progress of time to allow fast clock to align with actual time
- Change rate of clock interrupt routine
- Slow progress of time until synchronized
- NTP accuracy is within 1-50ms
- In Ubuntu Linux, to quickly synchronize time:
  `$apt install ntp ntpdate`
- Specify local timeservers in /etc/ntp.conf
  ```
  server time.u.washington.edu iburst
  server bigben.cac.washington.edu iburst
  ```
- Shutdown service (sudo service ntp stop)
- Run ntpdate: (sudo ntpdate time.u.washington.edu)
- Startup service (sudo service ntp start)

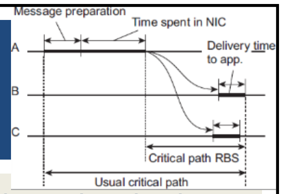
61

BERKELEY ALGORITHM

- Berkeley time daemon server actively polls network to determine average time across servers
- Suitable when no machine has a UTC receiver
- Time daemon instructs servers how much to adjust clocks to achieve precision
- Accuracy can not be guaranteed
- Berkeley is an internal clock synchronization algorithm

62

CLOCK SYNCHRONIZATION IN WIRELESS NETWORKS



- Sensor networks bring unique challenges for clock synchronization
  - ***Address resource constraints***: limited power, multihop routing slow
- Reference broadcast synchronization (RBS)
- Provides precision of time, not accuracy as in Berkeley
- No UTC clock available
- RBS sender broadcasts a reference message to allow receivers to adjust clocks
- No multi-hop routing
- Time to propagate a signal to nodes is roughly constant
- Message propagation time does not consider time spent waiting in NIC for message to send
  - Wireless network resource contention may force wait before message even ***can*** be sent

REFERENCE BROADCAST SYNCHRONIZATION (RBS)

- Node broadcasts reference message m
- Each node p records time Tp,m when m is received
- Tp,m is read from node p's clock
- Two nodes p and q can exchange delivery times to estimate mutual relative offset
- Then calculate relative average offset for the network:

$$Offset[p,q] = \frac{\sum_{k=1}^{M}(T_{p,k} - T_{q,k})}{M}$$

- Where M is the total number of reference messages sent
- Nodes can simply store offsets instead of frequently synchronizing clocks to save energy

REFERENCE BROADCAST SYNCHRONIZATION (RBS) - 2

- Cloud skew: over time clocks drift apart
- Averages become less precise
- Elson et al. propose using standard linear regression to predict offsets, rather than calculating them
- IDEA: Use node's history of message times in a simple linear regression to continuously refine a formula with coefficients to predict time offsets:

$$Offset[p,q](t) = \alpha t + \beta$$

65

QUESTIONS

66