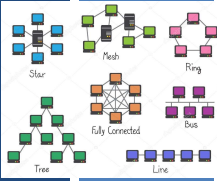



TCSS 558: APPLIED DISTRIBUTED COMPUTING

Communication & Coordination

Wes J. Lloyd
 Institute of Technology
 University of Washington - Tacoma

OBJECTIVES

- Assignment #1 Questions
- Assignment #2
- Feedback from 11/14

- Ch. 4 – Communications
 - Multicast communication

- Ch. 6 – Coordination
 - Physical clocks
 - Clock synchronization
 - Logical clocks, Lamport clocks

November 16, 2017
TCSS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma
L14.2

CHAPTER 4 - COMMUNICATION

- 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

November 16, 2017
TCSS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma
L14.3

FEEDBACK FROM 11/14

- How multicasting is effected by overlay networks?
- Application-level multicasting leverages overlay networks
- Create adhoc overlay network
- Support communication between nodes involved in the application
- Multicasting (e.g. zeromq, UDP) can leverage the network overlay to propagate application level traffic to the nodes


November 16, 2017
TCSS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma
L14.4

FEEDBACK - 2

- Intermediate concurrent hash table in assignment #2:
 - Chapter 8.5 Distributed Commit
 - Where should the intermediate concurrent hash table be deployed?
 - What is our aim to add this data structure?

November 16, 2017
TCSS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma
L14.5

CH. 4.4: MULTICAST COMMUNICATION



one to many
X = subscriber

Apache ActiveMQ

November 16, 2017
TCSS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma
L14.6

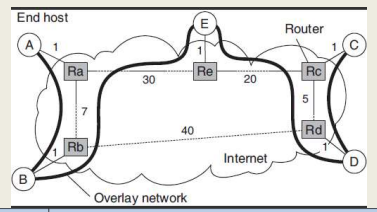
MULTICAST COMMUNICATION

- Sending data to multiple receivers
- Many **failed** proposals for network-level / transport-level protocols to support multicast communication
- **Problem:** How to set up communication paths for information dissemination?
- **Solutions:** require huge management effort, human invention
- Focus shifted more recently to **peer-to-peer** networks
 - Structured overlay networks can be setup easily and provide efficient communication paths
 - Application-level multicasting techniques more successful
 - Gossip-based dissemination: unstructured p2p networks

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.7

NETWORK STRUCTURE

- **Overlay network**
 - Virtual network implemented on top of an actual physical network
- **Underlying network**
 - The actual physical network that implements the overlay



November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.8

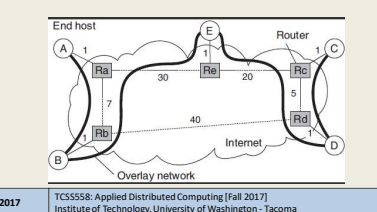
APPLICATION LEVEL TREE-BASED MULTICASTING

- Application level multi-casting
 - Nodes organize into an overlay network
 - Network routers not involved in group membership
 - Group membership is managed at the application level (A2)
- **Downside:**
 - Application-level routing likely less efficient than network-level
 - Necessary tradeoff until having better multicasting protocols at lower layers
- **Overlay topologies**
 - **TREE:** top-down, unique paths between nodes
 - **MESH:** nodes have multiple neighbors; multiple paths between nodes

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.9

MULTICAST TREE METRICS

- Measure quality of application-level multicast tree
- **Link stress:** is defined per link, counts how often a packet crosses same link (**Ideally not more than 1**)
- **Stretch:** ratio in delay between two nodes in the **overlay** vs. the **underlying** networks



November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.10

MULTICAST TREE METRICS - 2

- **Stretch (Relative Delay Penalty RDP)** for B to C routes:
- **Overlay:** B → Rb → Ra → Re → E → Re → Rc → Rd → D → Rd → Rc → C = 73
- **Underlying:** B → Rb → Rd → Rc → C = 47
- $73 / 47 = 1.55$
- **Tree cost:** Overall cost of the overlay network
- Ideally would like to minimize network costs
- Find a minimal spanning tree which minimizes total time for disseminating information

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.11

FLOOD-BASED MULTICASTING

- **Broadcasting:** every node in overlay receives message
- Key design issue: minimize the use of intermediate nodes for which the message is not intended
- **Tree:** if only the leaf nodes are to receive the multicast message, many intermediate nodes are involved
- **Solution:** construct an overlay network for each multicast group
- **Flooding:** each node simply forwards a message to each of its neighbors, except to the message originator

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.12

RANDOM GRAPHS

- When no information on the structure of the overlay network
- Assume network can be represented as a **Random graph**
- Probability Pedge that two nodes are joined
- Overlay will have: $\frac{1}{2} * P_{edge} * N * (N-1)$ edges

Number of nodes	Edges (x 1000) at p _{edge} = 0.6	Edges (x 1000) at p _{edge} = 0.4	Edges (x 1000) at p _{edge} = 0.2
100	~10	~5	~2
500	~150	~75	~35
1000	~3000	~1500	~700

November 16, 2017 TCS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma L14.13

PROBABILISTIC FLOODING

- Washington state in winter?
- When a node is flooding a message, concept is to enforce a probability of message spread (p_{flood})
- Throttles message flooding based on a probability
- Implementation needs to consider # of neighbors to achieve various p_{flood} scores
- With lower p_{flood} messages may not reach all nodes
- USEFULNESS:** 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

November 16, 2017 TCS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma L14.14

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
- Hypercube:** nodes forward only to higher dimension nodes
- $N(1001)$ broadcast will only go to $N(1011)$ and $N(1000)$
- Broadcast requires just: $N-1$ messages, where nodes $N=2^n$, $n=$ dimensions of hypercube

November 16, 2017 TCS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma L14.15

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

November 16, 2017 TCS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma L14.16

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

November 16, 2017 TCS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma L14.17

ANTI ENTROPY DISSEMINATION MODEL

- Anti-entropy:** Propagation model where node P picks node Q at random and exchanges messages updates
- Akin to random walk
- PULL:** P only pulls in new updates from Q
- PUSH:** P only pushes its own updates to 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

November 16, 2017 TCS558: Applied Distributed Computing [Fall 2017]
 Institute of Technology, University of Washington - Tacoma L14.18

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:

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.19

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)

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.20

RUMOR SPREADING - 2

- Does not guarantee all nodes will be updated
- The fraction of nodes s , that remain susceptible is 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 \rightarrow

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.21

DIRECTIONAL GOSSIPING

- Taking network topology into account can help
- When gossiping, nodes connected to only a few other nodes are more likely to be contacted
- **Epidemic protocols assume:**
- For gossiping nodes are randomly selected
- One node, can randomly select any other node in the network
- Complete set of nodes is known to each member

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.22

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

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.23

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**

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.24

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*)

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma L14.25

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

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma L14.26

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

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma L14.27

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 coordinate problems:
 - Aggregation
 - Peer sampling
 - Overlay construction

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma L14.28



CH. 6.1: CLOCK SYNCHRONIZATION

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma L14.29

CLOCK SYNCHORNIZATION

- 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 that 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

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma L14.30

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

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma L14.31

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.
- 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

1960s ERA radio crystal →

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma L14.32

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

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma L14.33

UNIVERSAL COORDINATED TIME

- 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

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma L14.34

COMPUTING: CLOCK CHALLENGES

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

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma L14.35

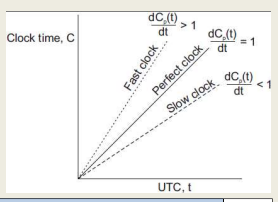
CLOCK SYNCHRONIZATION

- 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

November 16, 2017 TCCS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma L14.36

CLOCK SYNCHRONIZATION - 2

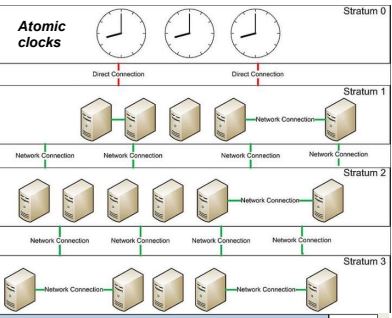
- If two clocks drift from UTC in opposite directions, after time Δt after synchronization, they may be 2ρ apart.
- Clocks must be resynchronized every $\pi/2\rho$ seconds
- **Network time protocol**
- Provide coordination of time for servers
- Leverage distributed network of time servers



November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.37

NETWORK TIME PROTOCOL

- Servers organized into stratum
- 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

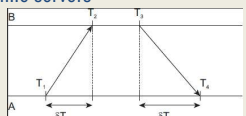


November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.38

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

November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.39

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)


November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.40

QUESTIONS



November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.41

EXTRA SLIDES



November 16, 2017
TCCS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma
L14.42