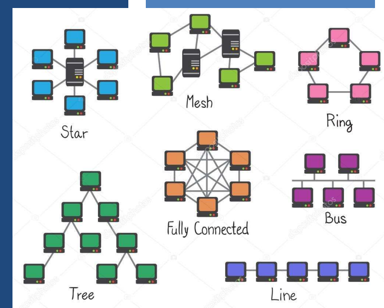
TCSS 558: APPLIED DISTRIBUTED COMPUTING

Distributed Systems Architectures

Wes J. Lloyd
School of Engineering and Technology
University of Washington - Tacoma



OBJECTIVES

- Homework 0 Questions
- Homework 1
- Feedback
- Chapter 2: System architectures
  - (X) Centralized: Single client, multi-tier
  - Decentralized peer-to-peer: structured, unstructured, hierarchical
  - Hybrid
- Chapter 3 Processes
  - 3.1 Threads
  - 3.2 Virtualization
  - 3.3 Clients
  - 3.4 Servers

FEEDBACK – 1/28

- **UDP can save network bandwidth, is it because UDP sends message w/o sequencing?**

TCP Segment Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset / Res	Flags	Window Size	
128	Header and Data Checksum		Urgent Pointer	
160…	Options			

UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

January 30, 2019 | TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma | L7.3

FEEDBACK – 1/28

- **UDP can save network bandwidth, is it because UDP sends message w/o sequencing?**

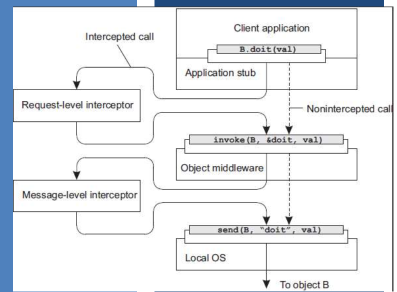
TCP Segment Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset / Res	Flags	Window Size	
128				
160…				

UDP header size is 1/3 the size: UDP (64-bytes) vs TCP (192-bytes)

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

January 30, 2019 | TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma | L7.4



SYSTEM ARCHITECTURES

TYPES OF SYSTEM ARCHITECTURES

- Centralized system architectures
  - Client-server
  - Multitiered
- Decentralized peer-to-peer architectures
  - Structured
  - Unstructured
  - Hierarchically organized
- Hybrid architectures

DECENTRALIZED PEER-TO-PEER ARCHITECTURES

- Client/server:
  - Nodes have specific roles
- Peer-to-peer:
  - Nodes are seen as ***all equal…***
- **How should nodes be organized for communication?**

STRUCTURED PEER-TO-PEER

- Nodes organized using specific *topology* (e.g. ring, binary-tree, grid, etc.)
  - Organization (structure) assists in data lookups
- Data indexed using "semantic-free" indexing
  - Key / value storage systems
  - Key used to look-up data
- Nodes store data associated with a subset of keys

DISTRIBUTED HASH TABLE (DHT)

- Distributed hash table (DHT) *(ch. 5)*
- Hash function

`key(data item) = hash(data item's value)`

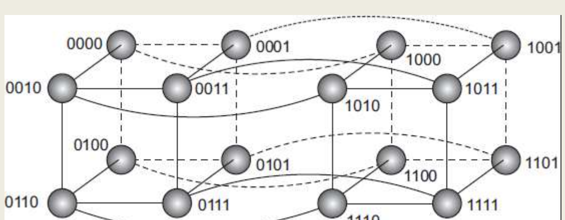
- Hash function "generates" a unique key based on the data
- No two data elements will have the same key (hash)
- System supports data lookup via key
- ***Any*** node can receive/resolve requests with the hash function
- Lookup function determines which node stores the key

`existing node = lookup(key)`

- Node forwards request to node with the data
- ***DOES this approach provide distribution transparency to clients?***

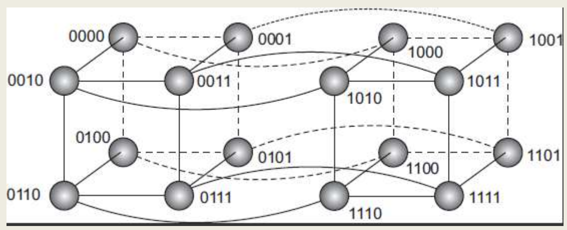
FIXED HYPERCUBE EXAMPLE

- Example where topology helps ***route*** data lookup request
- Statically sized 4-D hypercube, every node has 4 connectors
- 2 x 3-D cubes, 8 vertices, 12 edges
- Node IDs represented as 4-bit code (0000 to 1111)
- Hash data items to 4-bit key (1 of 16 slots)
- Distance (number of hops) determined by identifying number of varying bits between neighboring nodes and destination



FIXED HYPERCUBE EXAMPLE - 2

- **Example:** *fixed hypercube*
  node 0111 (7) retrieves data from node 1110 (14)
- Node 1110 is not a neighbor to 0111
- **Which connector leads to the shortest path?**



January 30, 2019 | TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma | L7.11

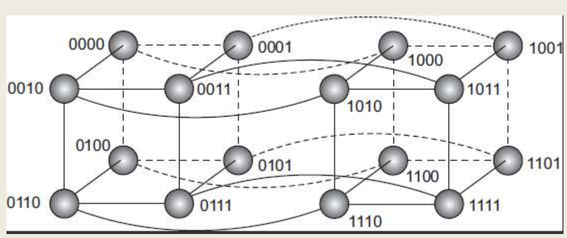
WHICH CONNECTOR LEADS TO THE SHORTEST PATH?

- **Example:** node 0111 (7) retrieves data from node 1110 (14)
- Node 1110 is not a neighbor to 0111

[0111] Neighbors:

1111 (1 bit different than 1110) 0011 (3 bits different– bad path)

0110 (1 bit different than 1110) 0101 (3 bits different– bad path)



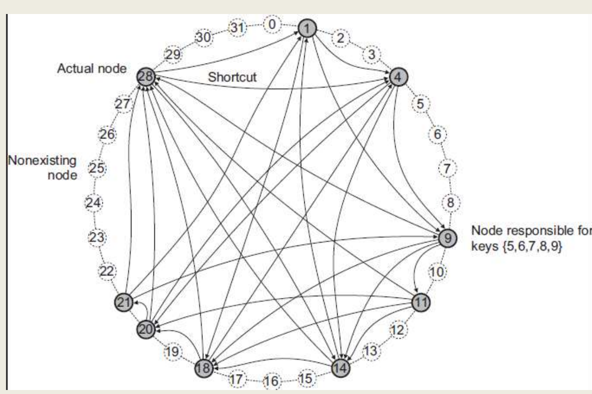
January 30, 2019 | TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma | L7.12

DYNAMIC TOPOLOGY

- Fixed hypercube requires <u>static topology</u>
  - Nodes cannot join or leave → *what if 1 node short of perfect cube?*
- Relies on symmetry of number of nodes
- Can force the DHT to a certain size
- Chord system – DHT (in ch.5)
  - Dynamic topology
  - Nodes organized in ring
  - Every node has unique ID
  - Each node connected with other nodes (shortcuts)
  - Shortest path between any pair of nodes is ~ order O(log N)
  - N is the total number of nodes

CHORD SYSTEM

- Data items have m-bit key
- Data item is stored at closest "successor" node with ID ≥ key k
- Each node maintains finger table of successor nodes
- Client sends key/value lookup to *any* node
- Node forwards client request to node with m-bit ID closest to, but not greater than key k
- Nodes must continually refresh finger tables by communicating with adjacent nodes to incorporate node joins/departures



UNSTRUCTURED PEER-TO-PEER

- **No topology:** ***How do nodes find out about each other?***
- Each node maintains ad hoc list of neighbors
- Facilitates nodes frequently joining, leaving, ad hoc systems
- **Neighbor:** node reachable from another via a network path
- Neighbor lists constantly refreshed
  - Nodes query each other, remove unresponsive neighbors
- Forms a "random graph"
- Predetermining network routes not possible
  - How would you calculate the route algorithmically?
- Routes must be discovered

January 30, 2019	TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma	L7.15

UNSTRUCTURED PEER-TO-PEER

- Methods to find/disseminate data in unstructured peer-to-peer networks
- Flooding
- Random Walks
- Policy-based search
- Alternate topology:
- Hierarchically organized peer-to-peer networks

January 30, 2019	TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma	L7.16

SEARCHING FOR DATA: UNSTRUCTURED PEER-TO-PEER SYSTEMS

- **Flooding**
- `[Node u]` sends request for data item to all neighbors
- `[Node v]`
  - Searches locally, responds to `[Node u]` (or forwarder) if having data
  - Forwards request to **ALL** neighbors
  - Ignores repeated requests
- Features
  - High network traffic
  - Fast search results by saturating the network with requests
  - Variable # of hops
  - Max number of hops or time-to-live (TTL) often specified
  - Requests can "retry" by gradually increasing TTL/max hops until data is found

SEARCHING FOR DATA - 2

- **Random walks**
- `[Node u]` asks a randomly chosen neighbor `[node v]`
- If `[node v]` does not have data, forwards request to a random neighbor
- Features
  - Low network traffic
  - Akin to sequential search
  - Longer search time
  - `[node u]` can perform parallel random walks to reduce search time
  - As few as 16..64 random walks effective to reduce search time
  - Timeout required - need to coordinate stopping network-wide walk when data is found...

SEARCHING FOR DATA - 3

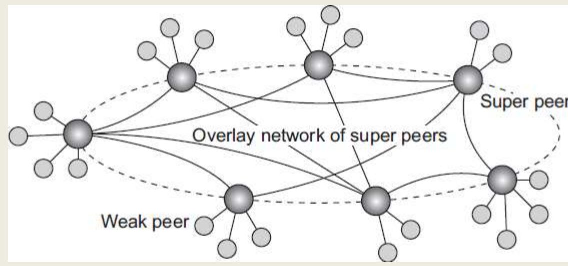
- **Policy-based search methods**
- Incorporate history and knowledge about the ad hoc network ***at the node-level*** to enhance effectiveness of queries
- Nodes maintain lists of preferred neighbors which often succeed at resolving queries
- Favor neighbors having highest number of neighbors
  - Can help minimize hops

HIERARCHICALLY ORGANIZED PEER-TO-PEER NETWORKS

- **Problem:**
  Ad hoc system search performance does not scale well as system grows
- Allow nodes to assume roles to improve search
- Content delivery networks (CDNs) *(video streaming)*
  - Store (cache) data at nodes local to the requester (client)
  - Broker node – tracks resource usage and node availability
    - Track where data is needed
    - Track which nodes have capacity (disk/CPU resources) to host data
- Node roles
  - Super peer –Broker node, routes client requests to storage nodes
  - Weak peer – Store data

HIERARCHICALLY ORGANIZED PEER-TO-PEER NETWORKS - 2

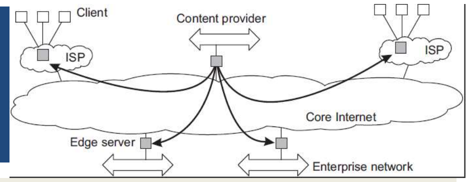
- Super peers
  - Head node of local centralized network
  - Interconnected via overlay network with other super peers
  - May have replicas for fault tolerance
- Weak peers
  - Rely on super peers to find data
- Leader-election problem:
  - Who can become a super peer?
  - What requirements must be met to become a super peer?



TYPES OF SYSTEM ARCHITECTURES

- Centralized system architectures
  - Client-server
  - Multitiered
- Decentralized peer-to-peer architectures
  - Structured
  - Unstructured
  - Hierarchically organized
- Hybrid architectures

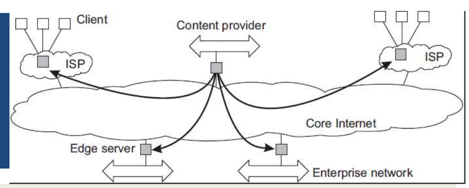
HYBRID ARCHITECTURES



- Combine centralized server concepts with decentralized peer-to-peer models
- <u>Edge-server systems:</u>
- Ad hoc peer-to-peer devices connect to the internet through an edge server (origin server)
- Edge servers (provided by an ISP) can optimize content and application distribution by storing assets near the edge
- <u>Example:</u>
- AWS Lambda@Edge: Enables Node.js Lambda Functions to execute "at the edge" harnessing existing CloudFront Content Delivery Network (CDN) servers
- https://www.infoq.com/news/2017/07/aws-lambda-at-edge

January 30, 2019	TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma	L7.23

HYBRID ARCHITECTURES - 2



- <u>Fog computing:</u>
- Extend the scope of managed resources beyond the cloud to leverage compute and storage capacity of end-user devices
- End-user devices become part of the overall system
- Middleware extended to incorporate managing edge devices as participants in the distributed system
- Cloud → in the sky
  - *compute/resource capacity is huge, but far away…*
- Fog → (devices) on the ground
  - *compute/resource capacity is constrained and local…*

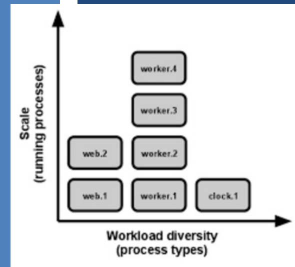
January 30, 2019	TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma	L7.24

COLLABORATIVE DISTRIBUTED SYSTEM EXAMPLE

- **BitTorrent Example:**
  File sharing system – users must contribute as a file host to be eligible to download file resources
- Original implementation features hybrid architecture
- Leverages idle client network capacity in the background
- User joins the system by interacting with a central server
- Client accesses global directory from a ***tracker*** server at well known address to access torrent file
- Torrent file tracks nodes having chunks of requested file
- Client begins downloading file chunks and immediately then participates to reserve downloaded content **or network bandwidth is reduced!!**
- Chunks can be downloaded in parallel from distributed nodes

REVIEW QUESTIONS

- What is difference in finding/disseminating data in unstructured vs. structured peer-to-peer networks?
  - Spreading/finding data
  - Flooding, Random walk
- What are some advantages of a decentralized structured peer-to-peer architecture?
- What are some disadvantages?
- What are some advantages of a decentralized unstructured peer-to-peer architecture?
- What are some disadvantages?



CH. 3: PROCESSES

L7.27

CHAPTER 3

- Chapter 3 titled "processes"
- Covers variety of distributed system implementation details
- "Grab bag" of topics

- Processes/threads
- Virtualization
- Clients
- Servers
- Code migration

January 30, 2019	TCSS558: Applied Distributed Computing [Winter 2019] School of Engineering and Technology, University of Washington - Tacoma	L7.28

THREADS

- For implementing a server (or client) threads offer many advantages vs. heavy weight processes
- <u>What is the difference between a process and a thread?</u>
  - Review from Operating Systems
- ***Key difference***: <u>**what do threads share amongst each other that processes do not…. ?**</u>
- <u>**What are the segments of a program stored in memory?**</u>
  - Heap segment (dynamic shared memory)
  - Code segment
  - Stack segment
  - Data segment (global variables)

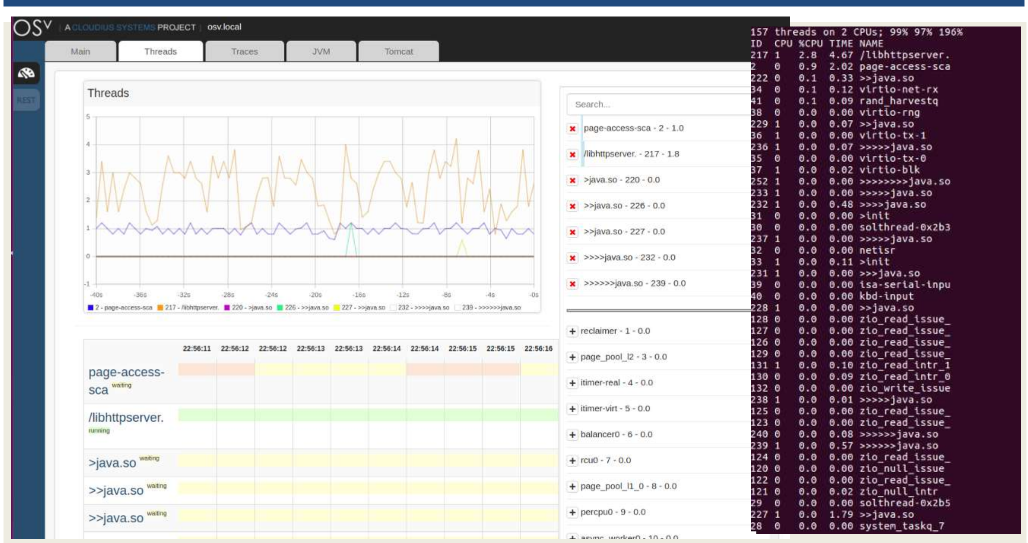
THREADS - 2

- <u>**Do several processes on an operating system share…**</u>
  - <u>**Heap segment?**</u>
  - <u>**Stack segment?**</u>
  - <u>**Code segment?**</u>
- <u>**Can we run multiple copies of the same code?**</u>
- These may be managed as shared pages (across processes) in memory
- Processes are isolated from each other by the OS
  - Each has a separate heap, stack, code segment

THREADS - 3

- Threads avoid the overhead of process creation
- No new heap or code segments required
- What is a context switch?
- Context switching among threads is considered to be more efficient than context switching processes
- Less elements to swap-in and swap-out
- Unikernel: specialized single process OS for the cloud
- Example: Osv, Clive, MirageOS (see: http://unikernel.org/projects/)
- Single process operating system with many threads
- Developed for the cloud to run only one application at a time

OSV: ONE PROCESS, MANY THREADS



THREADS - 4

- Important implications with threads:
- (1) multi-threading should lead to performance gains
- (2) thread programming requires additional effort when threads share memory
  - Known as thread **synchronization**, or enabling **concurrency**

- Access to **critical sections** of code which modify shared variables must be **mutually exclusive**
  - No more than one thread can execute at any given time
  - Critical sections must run **atomically** on the CPU

BLOCKING THREADS

- Example: spreadsheet with formula to compute sum of column
- User modifies values in column

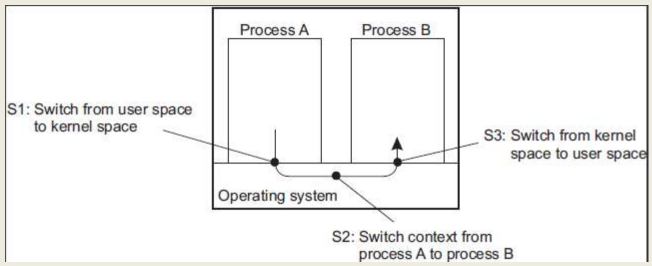
- Multiple threads:

1. Supports interaction (UI) activity with user
2. Updates spreadsheet calculations in parallel
3. Continually backs up spreadsheet changes to disk

- Single core CPU
  - Tasks appear as if they are performed simultaneously
- Multi core CPU
  - Tasks ***execute*** simultaneously

INTERPROCESS COMMUNICATION

- IPC – mechanism using pipes, message queues, and shared memory segments
- IPC mechanisms incur context switching
  - Process I/O must execute in kernel mode
- **How many context switches are required for process A to send a message to process B using IPC?**
- **#1 C/S:** Proc A→kernel thread
- **#2 C/S:** Kernel thread→Proc B



QUESTIONS