Agenda

• Administrative Information

• System Model

• Failures and Fault-Detection

• Communication

• Partition/Replication
Administrivia

• Ice-cream social: Thursdays 7:30pm-9pm

• Liteminer: will be released today

• Course status: currently ~ 100 students
  • Anticipate class stabilization by Thursday/Friday
  • Depending on stabilization – reach out to students
System Model
This Class

Challenges:
- Availability
- Performance
- Consistency
- Security

Algorithms and designs to enable a set of components to behave as one during arbitrary failures modes

The NETWORK!!!

Processes!! → Process!!

The NETWORK!!!
System Model

Components
- Nodes (i.e., processes)
- Channel (i.e., the network)
- No shared memory/global clock

Channel Properties:
- Unidirectional
- Allows communication

Nodes
- Execute program
- Store data in local memory (lost)
- Store data in disk (persistent)
- Local clock

Two Types of Events in our system:
- Local processing (e.g., function call)
- Message (data is sent between nodes)
Synchronous Versus Asynchronous

**Synchronous**
- Places bounds
  - Processing is bounded
  - Network transfer is bounded
  - Clock difference is bounded
- Less practical
- Easy to think about
- Most systems make this assumption

**Asynchronous**
- no bounds
- Most practical
Failures and Fault-detection
Reliability & Availability

• Things will crash. Deal with it!
  – Assume you could start with super reliable servers (MTBF of 30 years)
  – Build computing system with 10 thousand of those
  – Watch one fail per day

• Fault-tolerant software is inevitable

• Typical yearly flakiness metrics
  – 1-5% of your disk drives will die
  – Servers will crash at least twice (2-4% failure rate)
GUNS, SQUIRRELS, AND STEEL: THE MANY WAYS TO KILL A DATA CENTER
The Joys of Real Hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
~1 network rewiring (rolling ~5% of machines down over 2-day span)
~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
~5 racks go wonky (40-80 machines see 50% packetloss)
~8 network maintenances (4 might cause ~30-minute random connectivity losses)
~12 router reloads (takes out DNS and external vips for a couple minutes)
~3 router failures (have to immediately pull traffic for an hour)
~dozens of minor 30-second blips for dns
~1000 individual machine failures
~thousands of hard drive failures
  slow disks, bad memory, misconfigured machines, flaky machines, etc.

Long distance links: wild dogs, sharks, dead horses, drunken hunters, etc.
Impact of Scale on Failure Probability

- Assumption: Failure probability for a server = 0.01
System Model

Components
- Nodes (i.e., processes)
- Channel (i.e., the network)
- No shared memory/global clock

Failure!!
- Nodes fail independently
- Network can fail
Node: Failure Types/Modes

• Crash (Fail-stop)
  • Node stops working

• Partition
  • Network failed

• Byzantine Failures
  • Node starts lying

Hard to defend against: think BitCoin

• Gray Failure (@2017)
  • HoTOS papers from Microsoft/Google
  • Node is extremely slow
Channel/Network: Failure Types/Modes

- Drop packets
- Reorder packets
- Duplicate packets
More on Network Failures

• Packets are dropped:
  • Errors in optical layer
  • Router buffers are full

• Packets are reordered:
  • Different paths

• Packets are duplicated:
  • Special protocols

Microsoft on Optical Topologies: http://netseminar.stanford.edu/seminars/05_25_17.pdf
Failure Detection

• How would you detect another node has failed?

```
"Easy" to detect failures
Estimate time
Wait
If no response \rightarrow failure!
```
Communication
Networking Background

• Sending items → put in packages
Packets: The Networking Packages

- Data is cut up and put in packets
  - There’s a limit to packet size
  - There’s a hardware limit to how fast you can put packets into a network
Networking Protocols: Transport

• TCP: Transport Control Protocol
  • QUIC – is a specific algorithm here

• UDP: User Datagram Protocol

• How do you pick one versus the other?
TCP: Transport Control Protocol

• Connection oriented
• Provides build in loss recovery
• Paces the application

Hello, Are you there?
Hello, yes!
Got it!
Got it!
UDP: User Datagram Protocol

- NO Connection oriented
- Send as fast as you want
- No loss detection/recovery
TCP Versus UDP

- **TCP:**
  - Used by most applications
  - No need to re-implement loss recovery logic
  - Costly: slow to start and generally slower
  - BUT FAIR!!!

- **UDP:**
  - Good for short flows
  - Often used by real-time applications (e.g., Games, live-video)
    - WHY?
Communication between Nodes

• Abstraction: all nodes running as one process
  • Ideal: everything is a function call.

\[ Y = \text{sendTo}(\text{NodeX}, \text{data}) \]
Communication between Nodes

• Abstraction: all nodes running as one process
  • Ideal: everything is a function call.

Who is NodeX
pack data into a packet
Send packet to X
Wait()
Get result from X
Unpack result from packet
Return result

Y = sendTo(NodeX, data)

NodeA

NodeX

Hey, here’s some data.
do something
Working!

Done!
Remote Procedure Call

- Framework for automating this process
  - No need to re-write this code over (and over)
  - Hide complexity
  - Simplify coding
  - Provides similar abstraction (function calls)

- Developer defines:
  - Data types.
Key Challenges: Failures

• What can fail?

• How do you deal with these failures?