CS162
Operating Systems and Systems Programming
Lecture 25

Distributed Decision Making, Networking and TCP/IP

April 20th, 2023
Prof. John Kubiatowicz
http://cs162.eecs.Berkeley.edu
Recall: Societal Scale Information Systems

- The world is a large distributed system
  - Microprocessors in everything
  - Vast infrastructure behind them

Internet Connectivity

MEMS for Sensor Nets

scalable, Reliable, Secure Services

Databases
Information Collection
Remote Storage
Online Games
Commerce

...
Centralized vs Distributed Systems

- **Centralized System**: major functions performed by a single physical computer
  - Originally, everything on single computer
  - Later: client/server model
- **Distributed System**: physically separate computers working together on task
  - Early model: multiple servers working together
    » Probably in the same room or building
    » Often called a “cluster”
  - Later models: peer-to-peer/wide-spread collaboration
Distributed Systems: Motivation/Issues/Promise

• Why do we want distributed systems?
  – Cheaper and easier to build lots of simple computers
  – Easier to add power incrementally
  – Users can have complete control over some components
  – Collaboration: much easier for users to collaborate through network resources (such as network file systems)

• The *promise* of distributed systems:
  – *Higher availability*: one machine goes down, use another
  – *Better durability*: store data in multiple locations
  – *More security*: each piece easier to make secure
Distributed Systems: Reality

- Reality has been disappointing
  - **Worse availability**: depend on every machine being up
    - Lamport: “A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.”
  - **Worse reliability**: can lose data if any machine crashes
  - **Worse security**: anyone in world can break into system

- Coordination is more difficult
  - Must coordinate multiple copies of shared state information
  - What would be easy in a centralized system becomes a lot more difficult

- Trust/Security/Privacy/Denial of Service
  - Many new variants of problems arise as a result of distribution
  - Can you trust the other members of a distributed application enough to even perform a protocol correctly?
  - Corollary of Lamport’s quote: “A distributed system is one where you can’t do work because some computer you didn’t even know existed is successfully coordinating an attack on my system!”
Distributed Systems: Goals/Requirements

• **Transparency:** the ability of the system to mask its complexity behind a simple interface

• **Possible transparencies:**
  – **Location:** Can’t tell where resources are located
  – **Migration:** Resources may move without the user knowing
  – **Replication:** Can’t tell how many copies of resource exist
  – **Concurrency:** Can’t tell how many users there are
  – **Parallelism:** System may speed up large jobs by splitting them into smaller pieces
  – **Fault Tolerance:** System may hide various things that go wrong

• **Transparency and collaboration require some way for different processors to communicate with one another**
How do entities communicate? A Protocol!

- A protocol is an agreement on how to communicate, including:
  - Syntax: how a communication is specified & structured
    » Format, order messages are sent and received
  - Semantics: what a communication means
    » Actions taken when transmitting, receiving, or when a timer expires

- Described formally by a state machine
  - Often represented as a message transaction diagram
  - Can be a partitioned state machine: two parties synchronizing duplicate sub-state machines between them
  - Stability in the face of failures!
Examples of Protocols in Human Interactions

- Telephone
  1. (Pick up / open up the phone)
  2. Listen for a dial tone / see that you have service
  3. Dial
  4. Should hear ringing …
  5. Callee: “Hello?”
  6. Caller: “Hi, it’s Anthony…”
     Or: “Hi, it’s me” (← what’s that about?)
  7. Caller: “Hey, do you think … blah blah blah …” pause

1. Callee: “Yeah, blah blah blah …” pause
2. Caller: Bye
3. Callee: Bye
4. Hang up
Distributed Applications

• How do you actually program a distributed application?
  – Need to synchronize multiple threads, running on different machines
    » No shared memory, so cannot use test&set
  – One Abstraction: send/receive messages
    » Already atomic: no receiver gets portion of a message and two receivers cannot
      get same message

• Interface:
  – Mailbox (mbox): temporary holding area for messages
    » Includes both destination location and queue
  – Send(message,mbox)
    » Send message to remote mailbox identified by mbox
  – Receive(buffer,mbox)
    » Wait until mbox has message, copy into buffer, and return
    » If threads sleeping on this mbox, wake up one of them
Using Messages: Send/Receive behavior

• When should `send(message, mbox)` return?
  – When receiver gets message? (i.e. ack received)
  – When message is safely buffered on destination?
  – Right away, if message is buffered on source node?

• Actually two questions here:
  – When can the sender be sure that receiver actually received the message?
  – When can sender reuse the memory containing message?

• Mailbox provides 1-way communication from T1→T2
  – T1→buffer→T2
  – Very similar to producer/consumer
    » Send = V, Receive = P
    » However, can’t tell if sender/receiver is local or not!
Messaging for Producer-Consumer Style

• Using send/receive for producer-consumer style:

  Producer:
  ```c
  int msg1[1000];
  while(1) {
    prepare message;
    send(msg1,mbox);
  }
  ```

  Consumer:
  ```c
  int buffer[1000];
  while(1) {
    receive(buffer,mbox);
    process message;
  }
  ```

• No need for producer/consumer to keep track of space in mailbox: handled by send/receive
  – This is one of the roles of the window in TCP: window is size of buffer on far end
  – Restricts sender to forward only what will fit in buffer
Messaging for Request/Response communication

- What about two-way communication?
  - Request/Response
    » Read a file stored on a remote machine
    » Request a web page from a remote web server
  - Also called: client-server
    » Client = requester, Server = responder
    » Server provides “service” (file storage) to the client

- Example: File service

  Client: (requesting the file)
  ```c
  char response[1000];
  send("read rutabaga", server_mbox);
  receive(response, client_mbox);
  ```

  Server: (responding with the file)
  ```c
  char command[1000], answer[1000];
  receive(command, server_mbox);
  decode command;
  read file into answer;
  send(answer, client_mbox);
  ```
Adminstrivia

• Midterm 3: Next Thursday!
  – No class on day of midterm
  – Three double-sided pages of notes
  – Watch for Ed post about where you should go: we have multiple exam rooms
  – Conflict request form due Thursday!

• All material up to next Tuesday’s lecture is fair game

• Final deadlines during RRR week:
  – Yes, there will be office hour – watch for specifics
Distributed Consensus Making

• Consensus problem
  – All nodes propose a value
  – Some nodes might crash and stop responding
  – Eventually, all remaining nodes decide on the same value from set of proposed values

• Distributed Decision Making
  – Choose between “true” and “false”
  – Or Choose between “commit” and “abort”

• Equally important (but often forgotten!): make it durable!
  – How do we make sure that decisions cannot be forgotten?
    » This is the “D” of “ACID” in a regular database
  – In a global-scale system?
    » What about erasure coding or massive replication?
    » Like BlockChain applications!
General’s Paradox

• General’s paradox:
  – Constraints of problem:
    » Two generals, on separate mountains
    » Can only communicate via messengers
    » Messengers can be captured
  – Problem: need to coordinate attack
    » If they attack at different times, they all die
    » If they attack at same time, they win
  – Named after Custer, who died at Little Big Horn because he arrived a couple of days too early
General’s Paradox (con’t)

• Can messages over an unreliable network be used to guarantee two entities do something simultaneously?
  – Remarkably, “no”, even if all messages get through
    - No way to be sure last message gets through!
    - In real life, use radio for simultaneous (out of band) communication

• So, clearly, we need something other than simultaneity!
Two-Phase Commit

• Since we can’t solve the General’s Paradox (i.e. simultaneous action), let’s solve a related problem

• **Distributed transaction**: Two or more machines agree to do something, or not do it, **atomically**
  – No constraints on time, just that it will eventually happen!

• **Two-Phase Commit protocol**: Developed by Turing award winner Jim Gray
  – (first Berkeley CS PhD, 1969)
  – Many important DataBase breakthroughs also from Jim Gray
Two-Phase Commit Protocol

- **Persistent stable log on each machine:** keep track of whether commit has happened
  - If a machine crashes, when it wakes up it first checks its log to recover state of world at time of crash
- **Prepare Phase:**
  - The global coordinator requests that all participants will promise to commit or rollback the transaction
  - Participants record promise in log, then acknowledge
  - If anyone votes to abort, coordinator writes "Abort" in its log and tells everyone to abort; each records "Abort" in log
- **Commit Phase:**
  - After all participants respond that they are prepared, then the coordinator writes "Commit" to its log
  - Then asks all nodes to commit; they respond with ACK
  - After receive ACKs, coordinator writes "Got Commit" to log
- Log used to guarantee that all machines either commit or don’t
2PC Algorithm

• One coordinator
• N workers (replicas)
• High level algorithm description:
  – Coordinator asks all workers if they can commit
  – If all workers reply "VOTE-COMMIT", then coordinator broadcasts "GLOBAL-COMMIT"
    Otherwise coordinator broadcasts "GLOBAL-ABORT"
  – Workers obey the GLOBAL messages
• Use a persistent, stable log on each machine to keep track of what you are doing
  – If a machine crashes, when it wakes up it first checks its log to recover state of world at time of crash
Two-Phase Commit: Setup

• One machine (*coordinator*) initiates the protocol
• It asks *every* machine to *vote* on transaction

• Two possible votes:
  – *Commit*
  – *Abort*

• Commit transaction only if unanimous approval
Two-Phase Commit: Preparing

Worker Agrees to Commit
• Machine has **guaranteed** that it will accept transaction
• Must be **recorded in log** so machine will remember this decision if it fails and restarts

Worker Agrees to Abort
• Machine has **guaranteed** that it will **never accept** this transaction
• Must be **recorded in log** so machine will remember this decision if it fails and restarts
Two-Phase Commit: Finishing

Commit Transaction
• Coordinator learns *all machines have agreed to commit*
• Record decision to commit in local log
• Apply transaction, inform voters

Abort Transaction
• Coordinator learns *at least one machine has voted to abort*
• Record decision to abort in local log
• Do not apply transaction, inform voters
Two-Phase Commit: Finishing

Commit Transaction

• Coordinator learns *all machines have agreed to commit*
• Record decision to commit in local log
• Apply transaction, inform voters

Abort Transaction

• Coordinator learns *at least one machine has voted to abort*
• Record decision to abort in local log
• Do not apply transaction, inform voters

Because no machine can take back its decision, exactly one of these will happen
Two Phase Commit (2PC) can be described with interacting state machines

- Coordinator only waits for votes in “WAIT” state
  - In WAIT, if doesn’t receive N votes, it times out and sends GLOBAL-ABORT
- Worker waits for VOTE-REQ in INIT
  - Worker can time out and abort (coordinator handles it)
- Worker waits for GLOBAL-* message in READY
  - Coordinator fails ⇒ workers BLOCK waiting for coordinator to recover and send GLOBAL_* message
**Detailed Algorithm**

**Coordinator Algorithm**
- Coordinator sends **VOTE-REQ** to all workers
- Wait for **VOTE-REQ** from coordinator
- If ready, send **VOTE-COMMIT** to coordinator
- If not ready, send **VOTE-ABORT** to coordinator
- And immediately abort
- If receive **VOTE-COMMIT** from all N workers, send **GLOBAL-COMMIT** to all workers
- If don’t receive **VOTE-COMMIT** from all N workers, send **GLOBAL-ABORT** to all workers

**Worker Algorithm**
- Wait for **VOTE-REQ** from coordinator
- If ready, send **VOTE-COMMIT** to coordinator
- If not ready, send **VOTE-ABORT** to coordinator
  - And immediately abort
- If receive **GLOBAL-COMMIT** then commit
- If receive **GLOBAL-ABORT** then abort
Failure Free Example Execution

coordinator

worker 1

worker 2

worker 3

VOTE-REQ

GLOBAL-COMMIT

VOTE-COMMIT

time

4/20/23
Kubiatowicz CS162 © UCB Spring 2023

Lec 25.26
Example of Worker Failure

```
INIT
WAIT
ABORT
COMM
GLOBAL
ABORT

coordinator

worker 1

worker 2

worker 3

timeout

time

VOTE-REQ

VOTE-COMMIT

x

-REQ
-VOTE
-ABORT
-COMMIT
```
Example of Coordinator Failure #1

coordinator

worker 1

worker 2

worker 3

- VOTE-REQ
- timeout

- VOTE-ABORT
- timeout

- INIT
- READY
- ABORT
- COMM

4/20/23
Kubiatowicz CS162 © UCB Spring 2023
Lec 25.28
Example of Coordinator Failure #2

Coordinator

Worker 1

Worker 2

Worker 3

Restarted

Block waiting for coordinator

VOTE-REQ

VOTE-COMMIT

GLOBAL-ABORT

INIT

READY

ABORT

COMM

4/20/23
Kubiatowicz CS162 © UCB Spring 2023
Lec 25.29
Durability

• All nodes use **stable storage** to store current state
  – stable storage is non-volatile storage (e.g. backed by disk) that guarantees atomic writes.
  – E.g.: SSD, NVRAM

• Upon recovery, nodes can restore state and resume:
  – Coordinator **aborts** in INIT, WAIT, or ABORT
  – Coordinator **commits** in COMMIT
  – Worker **aborts** in INIT, ABORT
  – Worker **commits** in COMMIT
  – Worker “**asks**” Coordinator in READY
Alternatives to 2PC

• Three-Phase Commit: One more phase, allows nodes to fail or block and still make progress.
• PAXOS: An alternative used by Google and others that does not have 2PC blocking problem
  – Develop by Leslie Lamport (Turing Award Winner)
  – No fixed leader, can choose new leader on fly, deal with failure
  – Some think this is extremely complex!
• RAFT: PAXOS alternative from John Osterhout (Stanford)
  – Simpler to describe complete protocol

• What happens if one or more of the nodes is malicious?
  – Malicious: attempting to compromise the decision making
  – Use a more hardened decision making process: Byzantine Agreement and Block Chains
Byzantine General’s Problem

• Byzantine General’s Problem (n players):
  – One General and n-1 Lieutenants
  – Some number of these (f) can be insane or malicious

• The commanding general must send an order to his n-1 lieutenants such that the following Integrity Constraints apply:
  – IC1: All loyal lieutenants obey the same order
  – IC2: If the commanding general is loyal, then all loyal lieutenants obey the order he sends
Byzantine General’s Problem (con’t)

- Impossibility Results:
  - Cannot solve Byzantine General’s Problem with n=3 because one malicious player can mess up things
  - With f faults, need n > 3f to solve problem

- Various algorithms exist to solve problem
  - Original algorithm has #messages exponential in n
  - Newer algorithms have message complexity $O(n^2)$
    » One from MIT, for instance (Castro and Liskov, 1999)

- Use of BFT (Byzantine Fault Tolerance) algorithm
  - Allow multiple machines to make a coordinated decision even if some subset of them (< n/3 ) are malicious
Is a BlockChain a Distributed Decision Making Algorithm?

- BlockChain: a chain of blocks connected by hashes to root block
  - The Hash Pointers are unforgeable (assumption)
  - The Chain has no branches except perhaps for heads
  - Blocks are considered “authentic” part of chain when they have authenticity info in them
- How is the head chosen?
  - Some consensus algorithm
  - In many BlockChain algorithms (e.g. BitCoin, Ethereum), the head is chosen by solving hard problem
    » This is the job of “miners” who try to find “nonce” info that makes hash over block have specified number of zero bits in it
    » The result is a “Proof of Work” (POW)
    » Selected blocks above (green) have POW in them and can be included in chains
  - Longest chain wins
Is a Blockchain a Distributed Decision Making Algorithm? (Con’t)

- Decision means: Proposal is locked into BlockChain
  - Could be Commit/Abort decision
  - Could be Choice of Value, State Transition, ….
- NAK: Didn’t make it into the block chain (must retry!)
- Anyone in world can verify the result of decision making!
Network Protocols

- Networking protocols: many levels
  - Physical level: mechanical and electrical network (e.g., how are 0 and 1 represented)
  - Link level: packet formats/error control (for instance, the CSMA/CD protocol)
  - Network level: network routing, addressing
  - Transport Level: reliable message delivery

- Protocols on today’s Internet:
  - Ethernet
  - WiFi
  - LTE
  - IP
  - UDP
  - TCP
  - RPC
  - NFS
  - WWW
  - e-mail
  - ssh
• **Broadcast Network**: Shared Communication Medium

- Shared Medium can be a set of wires
  » Inside a computer, this is called a bus
  » All devices simultaneously connected to devices

- Originally, Ethernet was a broadcast network
  » All computers on local subnet connected to one another
- More examples (wireless: medium is air): cellular phones (GSM, CDMA, and LTE), WiFi
• **Media Access Control (MAC) Address:**
  – 48-bit physical address for hardware interface
  – Every device (in the world!?) has a unique address

• **Delivery:** When you broadcast a packet, how does a receiver know who it is for? (packet goes to everyone!)
  – Put header on front of packet: [ Destination MAC Addr | Packet ]
  – Everyone gets packet, discards if not the target
  – In Ethernet, this check is done in hardware
    » No OS interrupt if not for particular destination
Carrier Sense, Multiple Access/Collision Detection

- Ethernet (early 80’s): first practical local area network
  - It is the most common LAN for UNIX, PC, and Mac
  - Use wire instead of radio, but still broadcast medium
- Key advance was in arbitration called CSMA/CD: Carrier sense, multiple access/collision detection
  - Carrier Sense: don’t send unless idle
    » Don’t mess up communications already in process
  - Collision Detect: sender checks if packet trampled.
    » If so, abort, wait, and retry.
  - Backoff Scheme: Choose wait time before trying again
- How long to wait after trying to send and failing?
  - What if everyone waits the same length of time? Then, they all collide again at some time!
  - Must find way to break up shared behavior with nothing more than shared communication channel
- Adaptive randomized waiting strategy:
  - Adaptive and Random: First time, pick random wait time with some initial mean. If collide again, pick random value from bigger mean wait time. Etc.
  - Randomness is important to decouple colliding senders
  - Scheme figures out how many people are trying to send!
MAC Address: Unique Physical Address of Interface

- Can easily find MAC addr. on your machine/device:
  - E.g., `ifconfig` (Linux, Mac OS X), `ipconfig` (Windows)
Point-to-point networks

- Why have a shared bus at all? Why not simplify and only have point-to-point links + routers/switches?
  - Originally wasn’t cost-effective, now hardware is cheap!
- **Point-to-point network**: a network in which every physical wire is connected to only two computers
- **Switch**: a bridge that transforms a shared-bus (broadcast) configuration into a point-to-point network
  - Adaptively figures out which ports have which MAC addresses
- **Router**: a device that acts as a junction between physical networks to transfer data packets among them
  - Routes between switching domains using (for instance) IP addresses
The Internet Protocol (IP)

- Internet Protocol: Internet’s network layer
- Service it provides: “Best-Effort” Packet Delivery
  - Tries it’s “best” to deliver packet to its destination
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order
- IP Is a Datagram service!
  - Routes across many physical switching domains (subnets)
IPv4 Address Space

- **IP Address**: a 32-bit integer used as destination of IP packet
  - Often written as four dot-separated integers, with each integer from 0—255 (thus representing $8 \times 4 = 32$ bits)
  - Example CS file server is: $169.229.60.83 \equiv 0xA9E53C53$

- **Internet Host**: a computer connected to the Internet
  - Host has one or more IP addresses used for routing
    » Some of these may be private and unavailable for routing
  - Not every computer has a unique IP address
    » Groups of machines may share a single IP address
    » In this case, machines have private addresses behind a “Network Address Translation” (NAT) gateway

- **Subnet**: network connecting hosts with related IP addresses
  - A subnet is identified by 32-bit value, with the bits which differ set to zero, followed by a slash and a mask
    » Example: $128.32.131.0/24$ designates a subnet in which all the addresses look like $128.32.131.XX$
    » Same subnet: $128.32.131.0/255.255.255.0$
  - **Mask**: The number of matching prefix bits
    » Expressed as a single value (e.g., 24) or a set of ones in a 32-bit value (e.g., $255.255.255.0$)
  - Often routing *within* subnet is by MAC address (smart switches)
Address Ranges in IPv4

• IP address space divided into prefix-delimited ranges:
  – Class A: NN.0.0.0/8
    » NN is 1–126 (126 of these networks)
    » 16,777,214 IP addresses per network
    » 10.xx.yy.zz is private
    » 127.xx.yy.zz is loopback
  – Class B: NN.MM.0.0/16
    » NN is 128–191, MM is 0-255 (16,384 of these networks)
    » 65,534 IP addresses per network
    » 172.[16-31].xx.yy are private
  – Class C: NN.MM.LL.0/24
    » NN is 192–223, MM and LL 0-255 (2,097,151 of these networks)
    » 254 IP addresses per networks
    » 192.168.xx.yy are private

• Address ranges are often owned by organizations
  – Can be further divided into subnets
IPv4 Packet Format

- IP Packet Format:

- **IP Datagram**: an unreliable, unordered, packet sent from source to destination
  - Function of network – deliver datagrams!
Wide Area Network

- **Wide Area Network (WAN):** network that covers a broad area (e.g., city, state, country, entire world)
  - E.g., Internet is a WAN
- WAN connects multiple physical (datalink) layer networks (LANs)
- Datalink layer networks are connected by **routers**
  - Different LANs can use different communication technology (e.g., wireless, cellular, optics, wired)
Routers

- **Forward** each packet received on an **incoming link** to an **outgoing link** based on packet’s destination IP address (towards its destination)
- **Store & forward**: packets are buffered before being forwarded
- **Forwarding table**: mapping between IP address and the output link
Packet Forwarding

- Upon receiving a packet, a router
  - read the IP destination address of the packet
  - consults its forwarding table → output port
  - forwards packet to corresponding output port

- Default route (for subnets without explicit entries)
  - Forward to more authoritative router
IP Addresses vs. MAC Addresses

• Why not use MAC addresses for routing?
  – Doesn’t scale

• Analogy
  – MAC address → SSN
  – IP address → (unreadable) home address

• MAC address: uniquely associated with device for the entire lifetime of the device
• IP address: changes as the device location changes
  – Your notebook IP address at school is different from home
IP Addresses vs. MAC Addresses

• Why does packet forwarding using IP addr. scale?

• Because IP addresses can be aggregated
  – E.g., all IP addresses at UC Berkeley start with \(0xA9E5\), i.e., any address of form \(0xA9E5****\) belongs to Berkeley
  – Thus, a router in NY needs to keep a single entry for all hosts at Berkeley
  – If we were using MAC addresses the NY router would need to maintain an entry for every Berkeley host!!

• Analogy:
  – Give this letter to person with SSN: 123-45-6789 vs.
  – Give this letter to “John Smith, 123 First Street, LA, US”
Setting up Routing Tables

• How do you set up routing tables?
  – Internet has no centralized state!
    » No single machine knows entire topology
    » Topology constantly changing (faults, reconfiguration, etc.)
  – Need dynamic algorithm that acquires routing tables
    » Ideally, have one entry per subnet or portion of address
    » Could have “default” routes that send packets for unknown subnets to a different router that has more information

• Possible algorithm for acquiring routing table
  – Routing table has “cost” for each entry
    » Includes number of hops to destination, congestion, etc.
    » Entries for unknown subnets have infinite cost
  – Neighbors periodically exchange routing tables
    » If neighbor knows cheaper route to a subnet, replace your entry with neighbors entry (+1 for hop to neighbor)

• In reality:
  – Internet has networks of many different scales
  – Different algorithms run at different scales
Naming in the Internet

• How to map human-readable names to IP addresses?
  – E.g. www.berkeley.edu ⇒ 128.32.139.48
  – E.g. www.google.com ⇒ different addresses depending on location, and load

• Why is this necessary?
  – IP addresses are hard to remember
  – IP addresses change:
    » Say, Server 1 crashes gets replaced by Server 2
    » Or – google.com handled by different servers

• Mechanism: Domain Naming System (DNS)
Domain Name System

- DNS is a hierarchical mechanism for naming
  - Name divided in domains, right to left: www.eecs.berkeley.edu
- Each domain owned by a particular organization
  - Top level handled by ICANN (Internet Corporation for Assigned Numbers and Names)
  - Subsequent levels owned by organizations
- Resolution: series of queries to successive servers
- Caching: queries take time, so results cached for period of time
How Important is Correct Resolution?

- If attacker manages to give incorrect mapping:
  - Can get someone to route to server, thinking that they are routing to a different server
    » Get them to log into “bank” – give up username and password
- Is DNS Secure?
  - Definitely a weak link
    » What if “response” returned from different server than original query?
    » Get person to use incorrect IP address!
  - Attempt to avoid substitution attacks:
    » Query includes random number which must be returned
- In July 2008, hole in DNS security located!
  - Dan Kaminsky (security researcher) discovered an attack that broke DNS globally
    » One person in an ISP convinced to load particular web page, then all users of that ISP
      end up pointing at wrong address
  - High profile, highly advertised need for patching DNS
    » Big press release, lots of mystery
    » Security researchers told no speculation until patches applied
Network Layering

- **Layering:** building complex services from simpler ones
  - Each layer provides services needed by higher layers by utilizing services provided by lower layers
- The physical/link layer is pretty limited
  - Packets are of limited size (called the “Maximum Transfer Unit or MTU: often 200-1500 bytes in size)
  - Routing is limited to within a physical link (wire) or perhaps through a switch
- Our goal in the following is to show how to construct a secure, ordered, message service routed to anywhere:

<table>
<thead>
<tr>
<th>Physical Reality: Packets</th>
<th>Abstraction: Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited Size (MTU)</td>
<td>Arbitrary Size</td>
</tr>
<tr>
<td>Unordered (sometimes)</td>
<td>Ordered</td>
</tr>
<tr>
<td>Unreliable</td>
<td>Reliable</td>
</tr>
<tr>
<td>Machine-to-machine</td>
<td>Process-to-process</td>
</tr>
<tr>
<td>Only on local area net</td>
<td>Routed anywhere</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>Synchronous</td>
</tr>
<tr>
<td>Insecure</td>
<td>Secure</td>
</tr>
</tbody>
</table>
Recall: IPv4 Packet Format

- IP Packet Format:

  - **IP Header Length**: 4 bits
  - **IHL**: Internet Header Length
  - **Type of Service (ToS)**: 4 bits
  - **Total Length**: 16 bits
  - **Identification**: 16 bits
  - **Flags & Fragmentation**: 4 bits
  - **Fragment Offset**: 13 bits
  - **Protocol**: 8 bits
  - **Header Checksum**: 16 bits
  - **Time to Live (TTL)**: 8 bits
  - **Source IP Address**: 32 bits
  - **Destination IP Address**: 32 bits
  - **Options (if any)**
  - **Data**: Size of datagram (header + data)
  - **Flags & Fragmentation to split large messages**

**IP Datagram**: an unreliable, unordered, packet sent from source to destination
   - Function of network – deliver datagrams!
Building a messaging service on IP

• Process to process communication
  – Basic routing gets packets from machine→machine
  – What we really want is routing from process→process
    » Add “ports”, which are 16-bit identifiers
    » A communication channel (connection) defined by 5 items:
      [source addr, source port, dest addr, dest port, protocol]

• For example: The Unreliable Datagram Protocol (UDP)
  – Layered on top of basic IP (IP Protocol 17)
    » Datagram: an unreliable, unordered, packet sent from source user → dest user (Call it UDP/IP)

  – Important aspect: low overhead!
    » Often used for high-bandwidth video streams
    » Many uses of UDP considered “anti-social” – none of the “well-behaved” aspects of (say) TCP/IP
Internet Architecture: Five Layers

- Lower three layers implemented everywhere
- Top two layers implemented only at hosts
Internet Architecture: Five Layers

- Communication goes down to physical network
- Then from network peer to peer
- Then up to relevant layer
Layering Analogy: Packets in Envelopes
Internet Transport Protocols

- Datagram service (UDP): IP Protocol 17
  - No-frills extension of “best-effort” IP
  - Multiplexing/Demultiplexing among processes

- Reliable, in-order delivery (TCP): IP Protocol 6
  - Connection set-up & tear-down
  - Discarding corrupted packets (segments)
  - Retransmission of lost packets (segments)
  - Flow control
  - Congestion control

- Other examples:
  - DCCP (33), Datagram Congestion Control Protocol
  - RDP (26), Reliable Data Protocol
  - SCTP (132), Stream Control Transmission Protocol
Recall: Sockets in concept

Client

1. Create Client Socket
2. Connect it to server (host:port)
3. read request
4. write request
5. read response
6. Close Client Socket

Server

1. Create Server Socket
2. Bind it to an Address (host:port)
3. Listen for Connection
4. Accept syscall()
5. read request
6. write response
7. Close Connection Socket
Summary (1/3)

- A protocol is **an agreement on how to communicate**, including:
  - **Syntax**: how a communication is specified & structured
    - Format, order messages are sent and received
  - **Semantics**: what a communication means
    - Actions taken when transmitting, receiving, or when a timer expires
- **Consensus problem**
  - All nodes propose a value
  - Some nodes might crash and stop responding
  - Eventually, all remaining nodes decide on the same value from set of proposed values
- **Two-phase commit**: a form of distributed decision making
  - First, make sure everyone guarantees they will commit if asked (prepare)
  - Next, ask everyone to commit
Summary (2/3)

• Byzantine General’s Problem: distributed decision making with malicious failures
  – One general, n-1 lieutenants: some number of them may be malicious (often “f” of them)
  – All non-malicious lieutenants must come to same decision
  – If general not malicious, lieutenants must follow general
  – Only solvable if n ≥ 3f+1

• BlockChain protocols
  – Cryptographically-driven ordering protocol
  – Could be used for distributed decision making
Summary (3/3)

• Internet Protocol (IP): Datagram packet delivery
  – Used to route messages through routes across globe
  – 32-bit addresses, 16-bit ports

• DNS: System for mapping from names \(\Rightarrow\) IP addresses
  – Hierarchical mapping from authoritative domains
  – Recent flaws discovered

• Next time: TCP: Reliable byte stream between two processes on different machines over Internet (read, write, flush)
  – Uses window-based acknowledgement protocol
  – Congestion-avoidance dynamically adapts sender window to account for congestion in network