Recall: 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 John….”
     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.
     4. Hang up
     Callee: Bye
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
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?
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 on 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 on 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.
Detailed Algorithm

**Coordinator Algorithm**

- Coordinator sends **VOTE-REQ** to all workers
- 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
State Machine of Coordinator

- Coordinator implements simple state machine:

 INIT
  Recv: START
  Send: VOTE-REQ

 WAIT
  Recv: VOTE-ABORT
  Send: GLOBAL-ABORT
  Recv: all VOTE-COMMIT
  Send: GLOBAL-COMMIT

 ABORT
 COMMIT
State Machine of Workers

- **INIT**
  - Send: VOTE-ABORT
  - Recv: VOTE-REQ

- **READY**
  - Send: VOTE-COMMIT
  - Recv: VOTE-REQ

- **ABORT**
  - Recv: GLOBAL-ABORT

- **COMMIT**
  - Recv: GLOBAL-COMMIT
Dealing with Worker Failures

- Failure only affects states in which the coordinator is waiting for messages
- Coordinator only waits for votes in “WAIT” state
- In WAIT, if doesn’t receive N votes, it times out and sends GLOBAL-ABORT
Example of Worker Failure

- Coordinator
- Worker 1
- Worker 2
- Worker 3

Processes:
- INIT
- WAIT
- ABORT
- COMM
- VOTE-REQ
- VOTE-COMMIT
- GLOBAL-ABORT
- Timeout
- Time

Worker 3 is marked with an "X" indicating a failure.
Dealing with Coordinator Failure

- Worker waits for VOTE-REQ in INIT
  - Worker can time out and abort (coordinator handles it)
- Worker waits for GLOBAL-* message in READY
  - If coordinator fails, workers must BLOCK waiting for coordinator to recover and send GLOBAL_* message
Example of Coordinator Failure #1

```
coordinator

worker 1

worker 2

worker 3

VOTE-REQ

timeout

VOTE-ABORT
```
Example of Coordinator Failure #2

- VOTE
- INIT
- READY
- ABORT
- COMM

coordinator

worker 1

worker 2

worker 3

VOTE-REQ

VOTE-COMMIT

GLOBAL-ABORT

block waiting for coordinator

restarted

coordinator restarted
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**
Distributed Decision Making Discussion (1/2)

• Why is distributed decision making desirable?
  – Fault Tolerance!
  – A group of machines can come to a decision even if one or more of them fail during the process
    » Simple failure mode called “failstop” (different modes later)
  – After decision made, result recorded in multiple places

• Why is 2PC not subject to the General’s paradox?
  – Because 2PC is about all nodes eventually coming to the same decision – not necessarily at the same time!
  – Allowing us to reboot and continue allows time for collecting and collating decisions
Announcements

• HW 5 due Monday 11/22

• Midterm 2 regrades will close Sunday 11/21
Network Concepts

- **Network (interface) card/controller**: hardware that physically connects a computer to the network
- A computer can have more than one networking cards
  - E.g., one card for wired network, and one for wireless network
Network Concepts (cont’d)

- Typically, each network card is associated with two addresses:
  - Media Access Control (MAC), or physical, address
  - IP, or network, address (can be shared by network cards on the same host)
Network Concepts (cont’’d)

- **MAC address**: 48-bit unique identifier assigned by card vendor
- **IP Address**: 32-bit (or 128-bit for IPv6) address assigned by network administrator or dynamically when computer connects to network
Network Concepts (cont’d)

• **Connection**: communication channel between two processes
  
  • Each endpoint is identified by a **port number**
    
    – **Port number**: 16-bit identifier assigned by app or OS
    
    – Globally, an endpoint is identified by (IP address, port number)

---

![Diagram showing network concepts](image-url)
Main Network Functionalities

- **Delivery**: deliver packets between any two hosts in the Internet
  - E.g., how do you deliver a packet from a host in Berkeley to a host in Tokyo?
- **Reliability**: tolerate packet losses
  - E.g., how do you ensure all bits of a file are delivered in the presence of packet loses?
- **Flow control**: avoid overflowing the receiver buffer
  - Recall our bounded buffer example: stop sender from overflowing receiver’s buffer
    » E.g., how do you ensure that a sever that can send at 10Gbps doesn’t overwhelm a mobile phone?
- **Congestion control**: avoid overflowing the buffer of a router along the path
  - What happens if this happens?
Protocol Standardization

• Ensure communicating hosts speak the same protocol
  – Standardization to enable multiple implementations
  – Or the same folks have to write all the software

• Standardization: Internet Engineering Task Force
  – Based on working groups that focus on specific issues
  – Produces “Request For Comments” (RFCs)
    » Promoted to standards via rough consensus and running code
  – IETF Web site is http://www.ietf.org
  – RFCs archived at http://www.rfc-editor.org

• De facto standards: same folks writing the code
  – P2P file sharing, Skype, <your protocol here>…
Layering: The Problem

• Many different applications
  – email, web, P2P, etc.

• Many different network styles and technologies
  – Circuit-switched vs packet-switched, etc.
  – Wireless vs. wired vs optical, etc.

• How do we organize this mess?
The Problem (cont’d)

• Re-implement every application for every technology?
• No! But how does the Internet design avoid this?
Solution: Intermediate Layers

- Introduce intermediate layers that provide **set of abstractions** for various network functionality & technologies
  - A new app/media implemented only once
  - Variation on “add another level of indirection”

```
Application                  Skype  SSH  NFS  HTTP
Intermediate layers          
Transmission Media           Coaxial cable  Fiber optic  Packet radio
```
Partition system into modules & abstractions:

- Well-defined interfaces give flexibility
  - *Hides* implementation - thus, it can be freely changed
  - Extend functionality of system by adding new modules
- E.g., libraries encapsulating set of functionality
- E.g., programming language + compiler abstracts away not only how the particular CPU works …
  - … but also the *basic computational model*
- Well-defined interfaces hide information
  - Isolate *assumptions*
  - Present high-level *abstractions*
  - *But can impair performance*
Network System Modularity

Like software modularity, but:

• Implementation distributed across many machines (routers and hosts)

• Must decide:
  – How to break system into modules
    » Layering
  – What functionality does each module implement
    » End-to-End Principle
  – Where state is stored
    » Fate-sharing

• We will address these choices in turn
Layering: A Modular Approach

• Partition the system
  – Each layer *solely* relies on services from layer below
  – Each layer *solely* exports services to layer above

• Interface between layers defines interaction
  – Hides implementation details
  – Layers can change without disturbing other layers
Properties of Layers (OSI Model)

- **Service**: what a layer does
- **Service interface**: how to access the service  
  - Interface for layer above
- **Protocol** (*peer interface*): how peers communicate to implement the service  
  - Set of rules and formats that specify the communication between network elements  
  - Does *not* specify the implementation on a single machine, but how the layer is implemented *between* machines
OSI Layering Model

• Open Systems Interconnection (OSI) model
  – Developed by International Organization for Standardization (OSI) in 1984
  – **Seven** layers

• Internet Protocol (IP)
  – Only **five** layers
  – The functionalities of the missing layers (i.e., Presentation and Session) are provided by the Application layer

Application
Presentation
Session
Transport
Network
Datalink
Physical
Physical Layer (1)

- **Service**: move information between two systems connected by a physical link
- **Interface**: specifies how to send and receive bits
- **Protocol**: coding scheme used to represent a bit, voltage levels, duration of a bit
- **Examples**: coaxial cable, optical fiber links; transmitters, receivers
Datalink Layer (2)

- **Service:**
  - Enable end hosts to exchange frames (atomic messages) on the same physical line or wireless link
  - Possible other services:
    - Arbitrate access to common physical media
    - May provide reliable transmission, flow control

- **Interface:** send frames to other end hosts; receive frames addressed to end host

- **Protocols:** addressing, Media Access Control (MAC) (e.g., CSMA/CD - Carrier Sense Multiple Access / Collision Detection)
Datalink Layer (2)

- Each frame has a header which contains a source and a destination MAC address
- MAC (Media Access Control) address
  - Uniquely identifies a network interface
  - 48-bit, assigned by the device manufacturer
MAC Address Examples

• Can easily find MAC addr. on your machine/device:
  – E.g., ifconfig (Linux, Mac OS X), ipconfig (Windows)
Local Area Networks (LANs)

- LAN: group of hosts/devices that
  - are in the same geographical proximity (e.g., same building, room)
  - use same physical communication technology
- Examples:
  - all laptops connected wirelessly at a Starbucks café
  - all devices and computers at home
  - all hosts connected to wired Ethernet in an office

![Ethernet cable and port](image)
LANs

- All hosts in a LAN can share same physical communication media
  - Also called, broadcast channel
- Each frame is delivered to every host
- If a host is not the intended recipient, it drops the frame

MAC Addr: A  MAC Addr: B  MAC Addr: C
Switches

- Hosts in same LAN can be also connected by switches
- A switch forwards frames only to intended recipients
  - Far more efficient than broadcast channel
Media Access Control (MAC) Protocols

• Problem:
  – How do hosts access a broadcast media?
  – How do they avoid collisions?

• Three solutions:
  – Channel partition
  – “Taking turns”
  – Random access
MAC Protocols

• Channel partitioning protocols:
  – Allocate 1/N bandwidth to every host
  – Share channel efficiently and fairly at high load
  – Inefficient at low load (where load = # senders):
    » 1/N bandwidth allocated even if only 1 active node!
  – E.g., Frequency Division Multiple Access (FDMA); optical networks

• “Taking turns” protocols:
  – Pass a token around active hosts
  – A host can only send data if it has the token
  – More efficient at low loads: single node can use >> 1/N bandwidth
  – Overhead in acquiring the token
  – Vulnerable to failures (e.g., failed node or lost token)
  – E.g., Token ring
MAC Protocols

• Random Access
  – Efficient at low load: single node can fully utilize channel
  – High load: collision overhead

• Key ideas of random access:
  – **Carrier sense (CS)**
    » *Listen before speaking, and don’t interrupt*
    » Checking if someone else is already sending data
    » … and waiting till the other node is done
  – **Collision detection (CD)**
    » *If someone else starts talking at the same time, stop*
    » Realizing when two nodes are transmitting at once
    » …by detecting that the data on the wire is garbled
  – **Randomness**
    » *Don’t start talking again right away*
    » Waiting for a random time before trying again

– Examples: CSMA/CD, Ethernet, best known implementation
(Inter) Network Layer (3)

• **Service:**
  – Deliver packets to specified network addresses across multiple datalink layer networks
  – Possible other services:
    » Packet scheduling/priority
    » Buffer management

• **Interface:** send packets to specified network address destination; receive packets destined for end host

• **Protocols:** define network addresses (globally unique); construct forwarding tables; packet forwarding
(Inter) Network Layer (3)

- **IP address**: unique addr. assigned to network device
- Assigned by network administrator or dynamically when host connects to network

![Network Layer Diagram](image)

- **Transport Layer**
  - Network Layer
  - Datalink Layer
  - Physical Layer

- **Network Layer**
  - Net. Hdr.
  - Net. Payload

- **Datalink Layer**
  - Frame Hdr.
  - Net. Hdr.
  - Net. Payload

- **Physical Layer**
  - Frame Payload

- **Transport Layer**
  - IP Dest. Address
  - IP Src. Address
  - …
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 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
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 to the 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 \texttt{0xA9E5}, i.e., any address of form \texttt{0xA9E5****} belongs to Berkeley
    – Thus, a router in NY needs to keep a \textit{single} entry for \textit{all} hosts at Berkeley
    – If we were using MAC addresses the NY router would need to maintain \textit{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”
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
Transport Layer (4)

• **Service:**
  – Provide end-to-end communication between processes
  – **Demultiplexing** of communication between hosts
  – Possible other services:
    » **Reliability** in the presence of errors
    » **Timing** properties
    » **Rate adaptation** (flow-control, congestion control)

• **Interface:** send message to specific process at given destination; local process receives messages sent to it

• **Protocol:** port numbers, perhaps implement reliability, flow control, packetization of large messages, framing

• Examples: TCP and UDP
Port Numbers

- Port number: 16-bit number identifying the end-point of a transport connection
  - E.g., 80 identifies the port on which a processing implementing HTTP server can be connected
Internet Transport Protocols

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

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

- Services not available
  - Delay and/or bandwidth guarantees
  - Sessions that survive change-of-IP-address
Application Layer (7 - not 5!)

- **Service**: any service provided to the end user
- **Interface**: depends on the application
- **Protocol**: depends on the application

- Examples: Skype, SMTP (email), HTTP (Web), Halo, BitTorrent …

- What happened to layers 5 & 6?
  - “Session” and “Presentation” layers
  - Part of **OSI** architecture, but not Internet architecture
  - Their functionality is provided by application layer
Application Layer (5)

- **Application Layer**
  - **Transport Layer**
    - **Network Layer**
      - **Datalink Layer**
        - **Physical Layer**
          - Data: 101010100110101110
          - Data: 101010100110101110
Five Layers Summary

- Lower three layers implemented everywhere
- Top two layers implemented only at hosts
- Logically, layers interacts with peer’s corresponding layer
Physical Communication

- Communication goes down to physical network
- Then from network peer to peer
- Then up to relevant layer
Summary (1/2)

• Two-phase commit: distributed decision making
  – First, make sure everyone guarantees they will commit if asked (prepare)
  – Next, ask everyone to commit

• 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
Summary (2/2)

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

• 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