# DSC 204A: Scalable Data Systems Fall 2025

Staff
Instructor: Hao Zhang
TAs: Mingjia Huo, Yuxuan Zhang

② @haozhangml ② @haoailab

 haozhang@ucsd.edu

### Where We Are

Machine Learning Systems

Big Data

Cloud

2000 - 2016

Foundations of Data Systems

1980 - 2000

# Logistics

- BoQ Survey completion rate: 107%
  - All of you will get 1 bonus point
- Overall Feedback
  - Want more advanced stuff, less basics
  - Want to learn ML/LLM Systems (Many expect >= 30% content on it)
  - Want slower pace on content
- Waitlisted Students:
  - Please get in touch with me/TA to get enrolled (there should be room now)

### Evolution of Cloud Infrastructure

- Data Center: Physical space from which a cloud is operated
- 3 generations of data centers/clouds:
  - Cloud 1.0 (Past)
  - Cloud 2.0 (Current)
  - Cloud 3.0 (Ongoing Research)

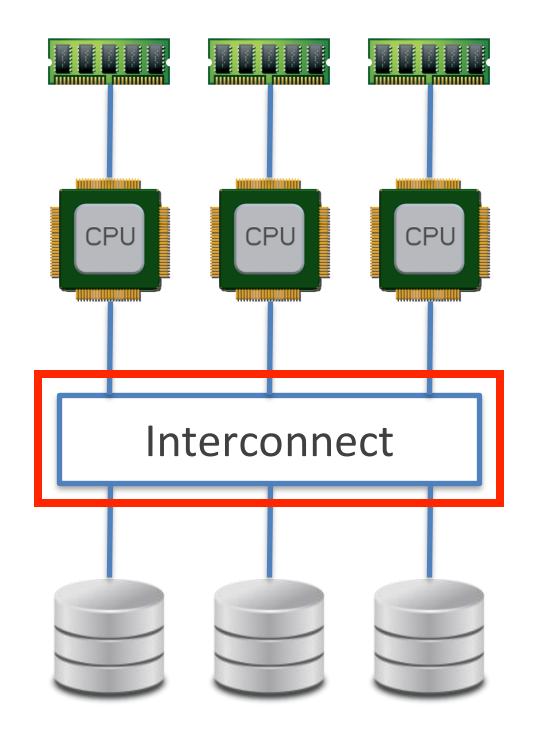
# Cloud 1.0 (Past)

- Networked servers;
- User rents servers (time-sliced access) needed for data/software

# Cloud 2.0 (Current)

- "Virtualization" of networked servers;
- User rents amount of resource capacity (e.g., memory, disk);
- Cloud provider has a lot more flexibility on provisioning (multitenancy, load balancing, more elasticity, etc.)

### Parallelism in the Cloud

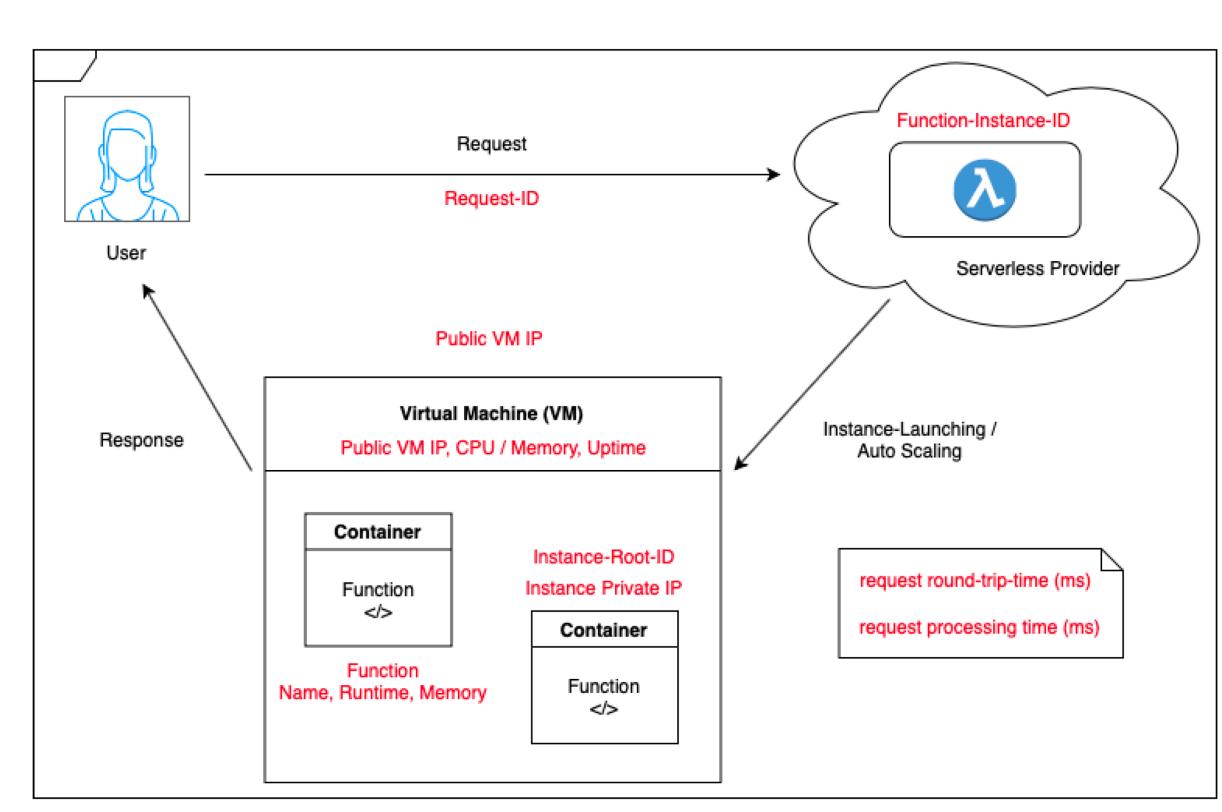


Modern networks in data centers have become much faster: 100GbE to even TbE!

- Decoupling of compute+memory from storage is common in cloud
  - Hybrids of shared-disk parallelism + shared-nothing parallelism
  - E.g., store datasets on S3 and read as needed to local EBS

# Cloud 3.0 (Ongoing Research)

- Full resource disaggregation! That is, compute, memory, storage, etc. are all network-attached and elastically added/removed
- User gives a program (function) to run and specifies CPU and DRAM needed
- Cloud provider abstracts away all resource provisioning entirely
- Aka Function-as-a-Service (FaaS)



# Cloud 3.0 (Ongoing Research)

- "Serverless" and disaggregated resources all connected to fast networks
- Serverless paradigm gaining traction for some applications, e.g., online ML prediction serving on websites

Cold start

Keep warm

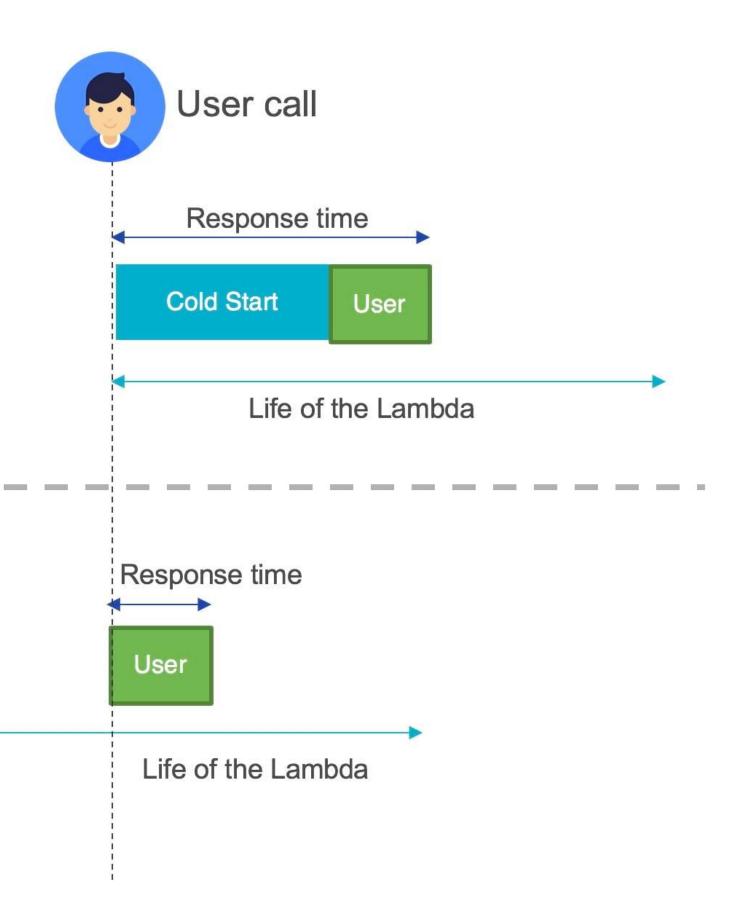
Cold Start

Cron

Life of the Lambda

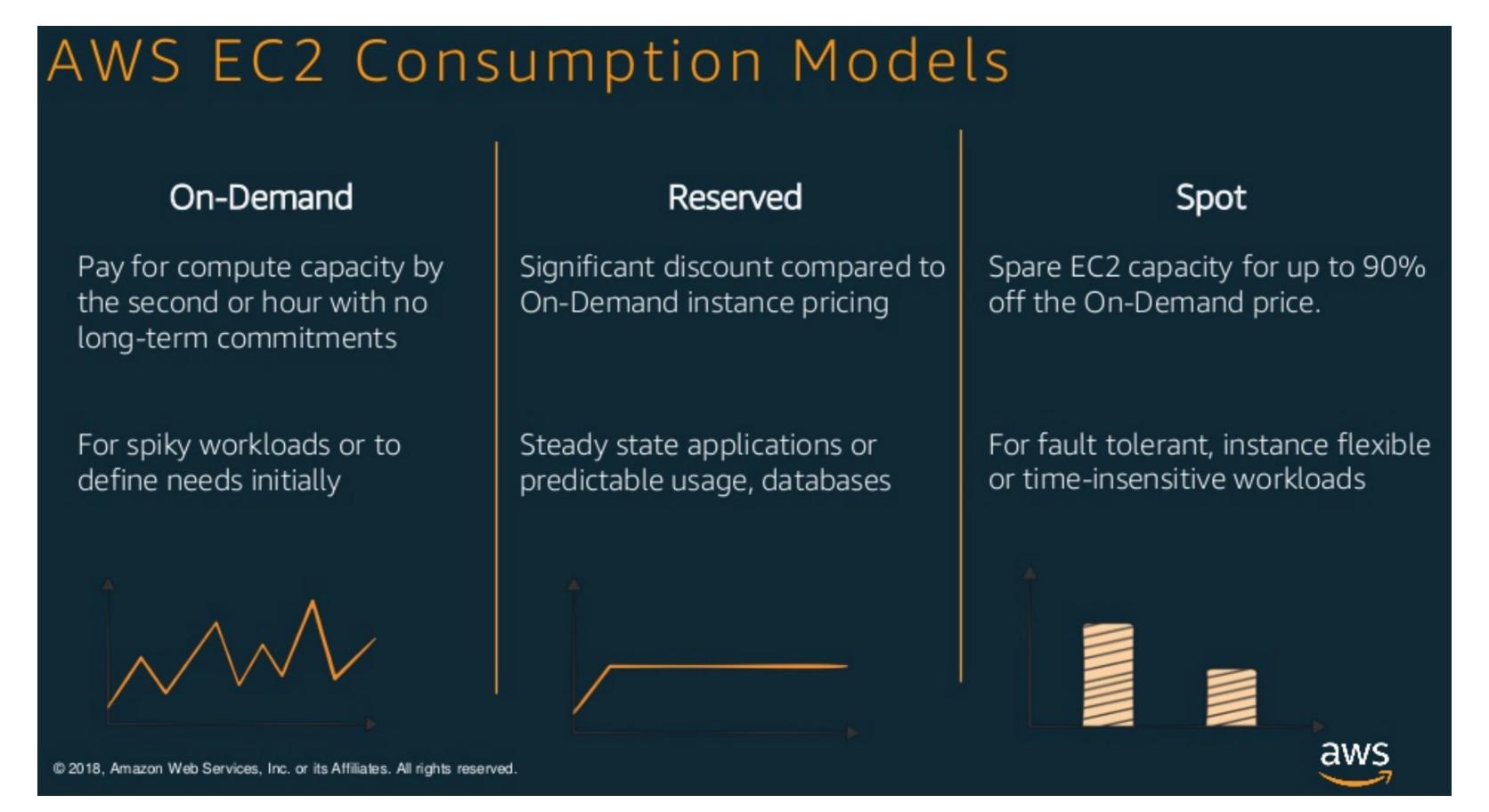
Cron

 Higher resource efficiency; much cheaper, often by 10x vs Spot instances



# New Cloud Renting Paradigms

- Cloud 2.0's flexibility enables radically different paradigms
- AWS example below; Azure and GCP have similar gradations



# More on Spot vs On-Demand

	Spot Instances	On-Demand Instances
Launch time	Can only be launched immediately if the Spot Request is active and capacity is available.	Can only be launched immediately if you make a manual launch request and capacity is available.
Available capacity	If capacity is not available, the Spot Request continues to automatically make the launch request until capacity becomes available.	If capacity is not available when you make a launch request, you get an insufficient capacity error (ICE).
Hourly price	The hourly price for Spot Instances varies based on demand.	The hourly price for On-Demand Instances is static.
Rebalance recommendation	The signal that Amazon EC2 emits for a running Spot Instance when the instance is at an elevated risk of interruption.	You determine when an On- Demand Instance is interrupted (stopped, hibernated, or terminated).
Instance interruption	You can stop and start an Amazon EBS-backed Spot Instance. In addition, the Amazon EC2 Spot service can interrupt an individual Spot Instance if capacity is no longer available, the Spot price exceeds your maximum price, or demand for Spot Instances increases.	You determine when an On- Demand Instance is interrupted (stopped, hibernated, or terminated).

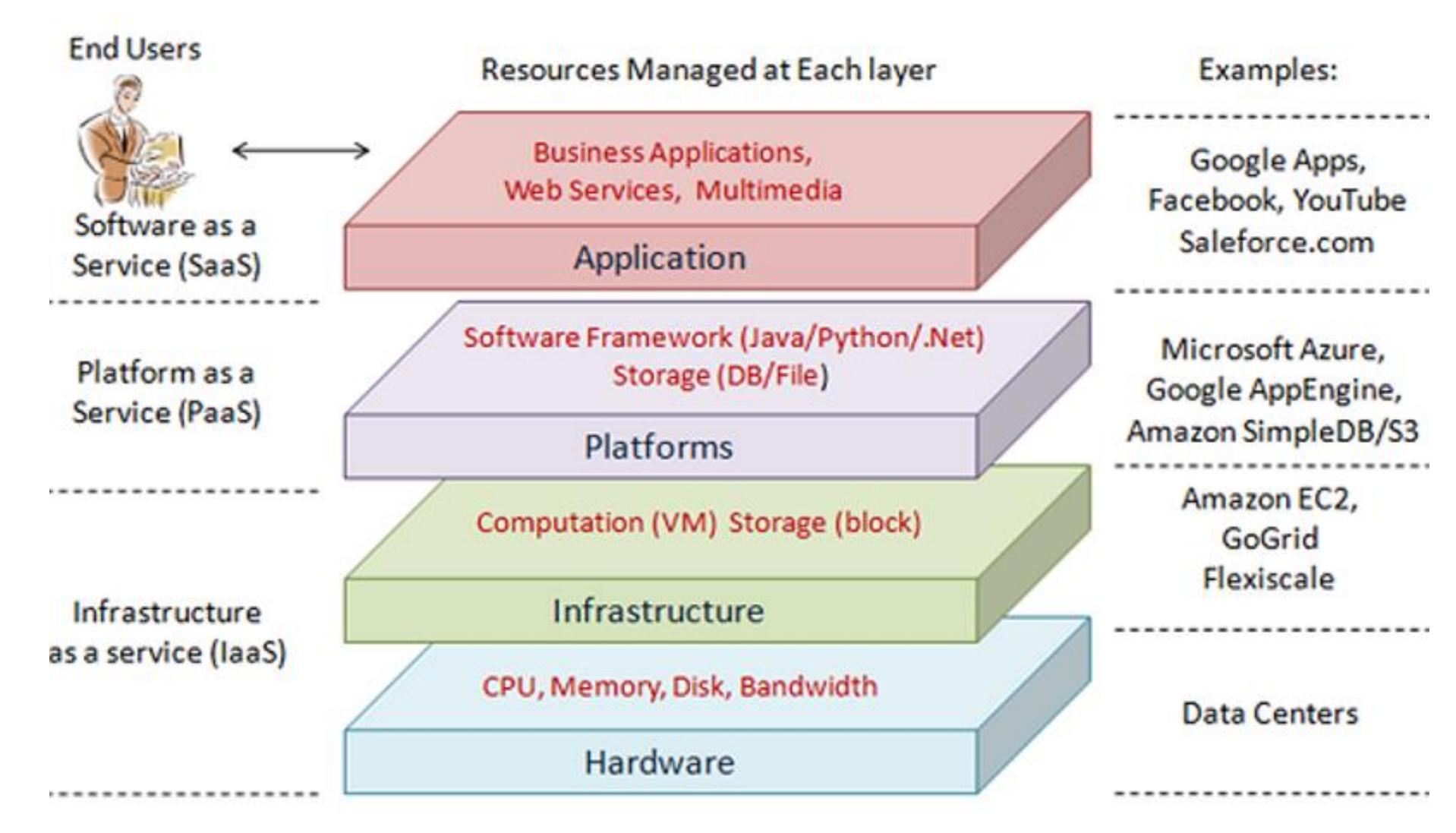
# Advantage and disadvantage

- Cloud 1.0:
  - +: Simple, Perfect isolation,
  - -: Expensive.
- Cloud 2.0:
  - +: Cheaper than Cloud 1.0.
  - -: Some resource waste
- Cloud 3.0:
  - +: Cheapest
  - -: Cold-start issues, Security & Privacy, Hard to manage.

# Recap: Cloud Computing v.s. on-premise clusters

- Compute, storage, memory, networking, etc. are virtualized and exist on remote servers; rented by application users
- Main pros of cloud vs on-premise clusters:
  - Manageability: Managing hardware is not user's problem
  - Pay-as-you-go: Fine-grained pricing economics based on actual usage (granularity: seconds to years!)
  - Elasticity: Can dynamically add or reduce capacity based on actual workload's demand
- Infrastructure-as-a-Service (laaS); Platform-as-a-Service (PaaS);
   Software-as-a-Service (SaaS)

# Cloud Computing is a Huge Ecosystem



# However: we are at a transition point, again.

# Profit chai









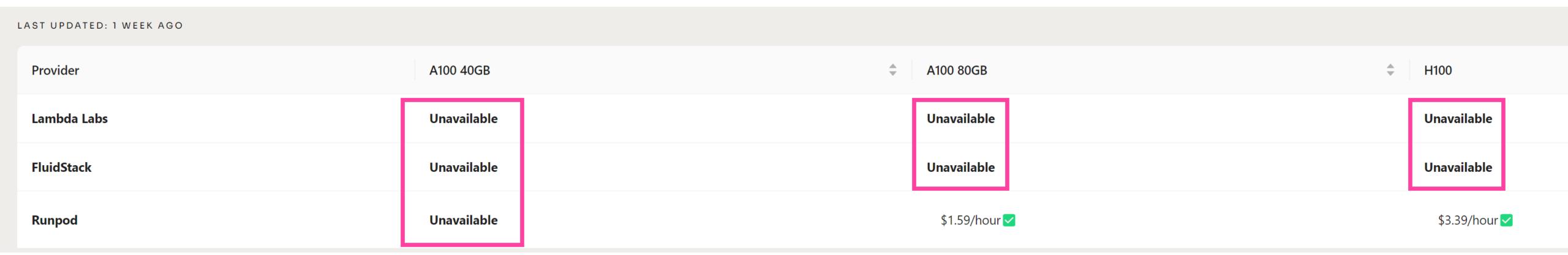






# New Trends in the Deep Learning Era

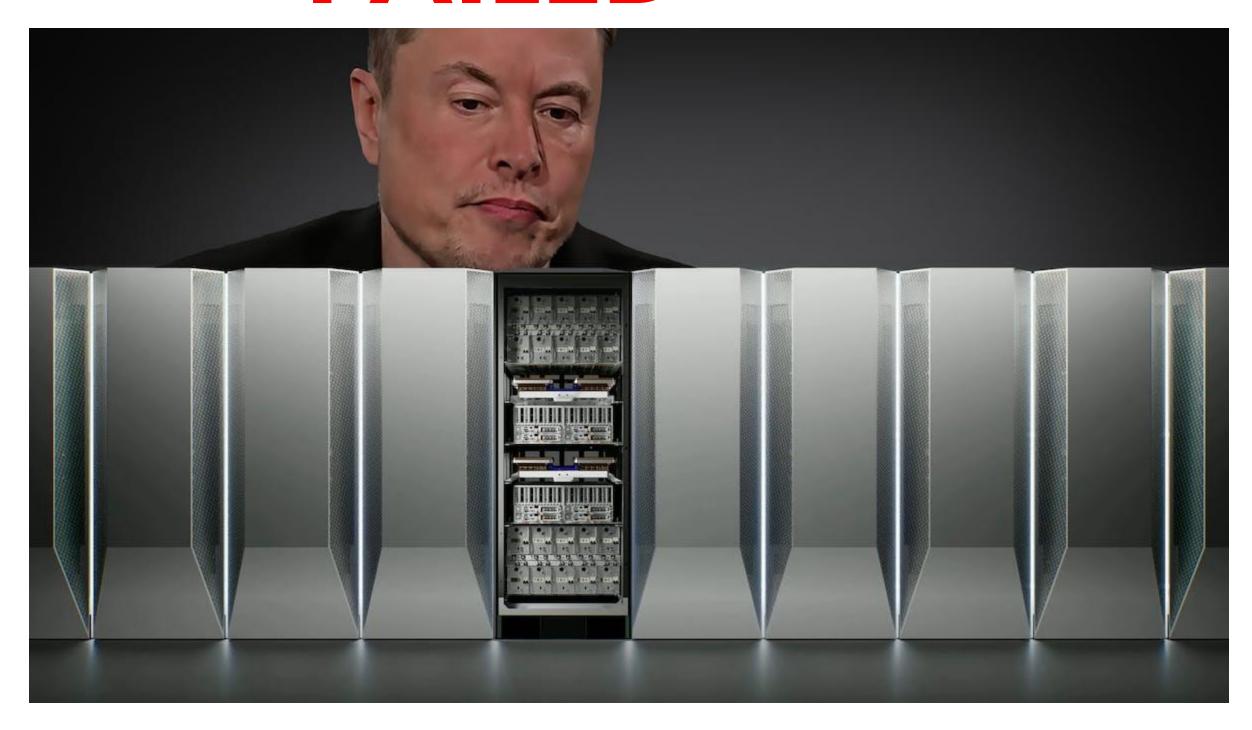
- New trends:
  - Reservation takes precedence than on-demand/spot
  - GPU vertical cloud
  - Community cloud



### However

There is a trend of building on-premise super computers again

# FAILED





# But More are Coming!

January 21, 2025 Company

## Announcing The Stargate Project



### Zuckerberg says Meta will build data center the size of Manhattan in latest AI push

CEO says company plans to spend hundreds of billions on developing artificial intelligence products

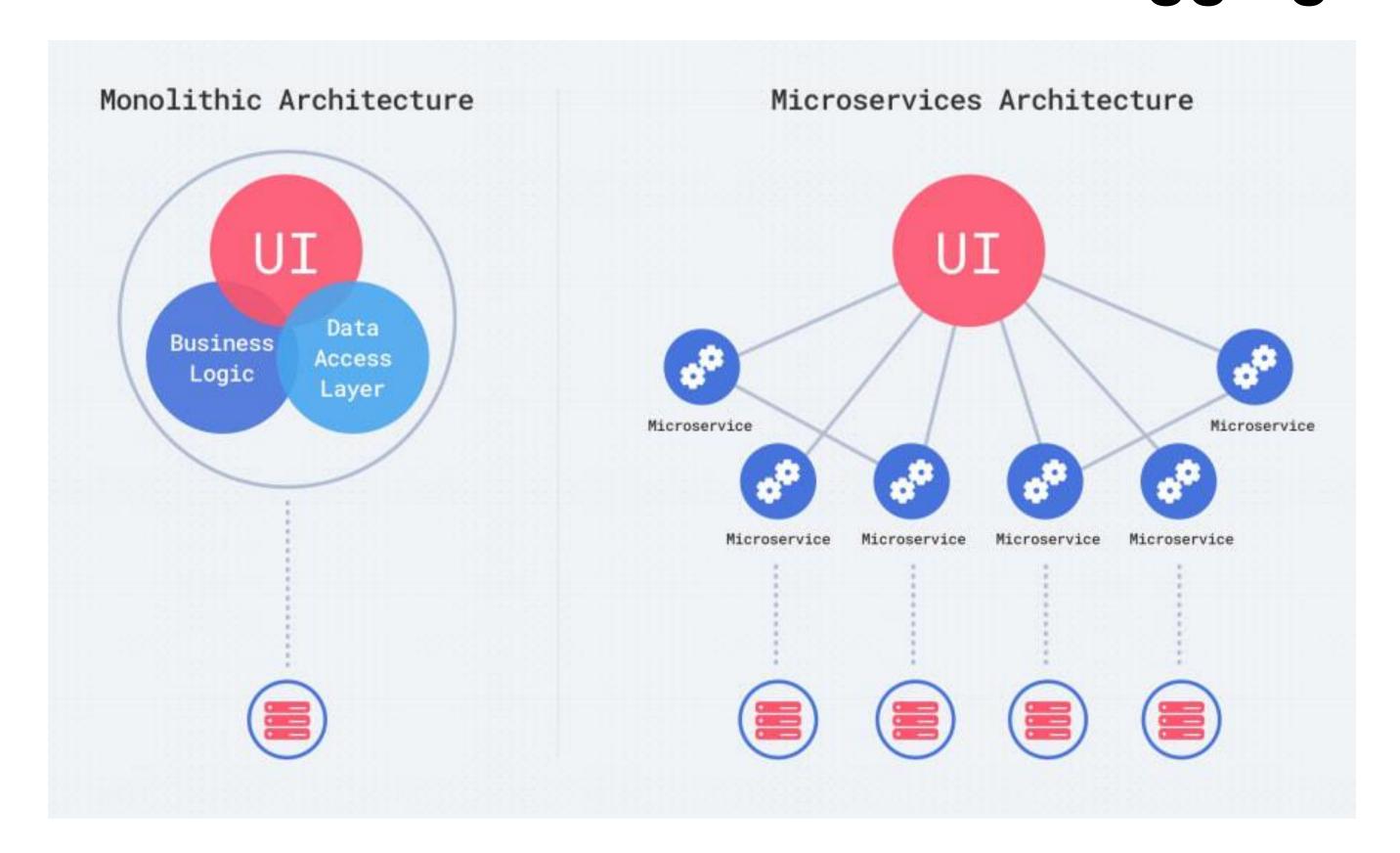


# Open Question after class

Google has pioneered and created many distributed systems and technologies that shape today's cloud computing, but why Amazon (and even Microsoft) wins over Google Cloud (GCP) on Cloud computing market shares?

### Instructor's Take: Amazon's Micro-services vs. Cloud 3.0

• Microservice architecture correlates well with today's cloud trend: resources and services are more and more **disaggregated**.

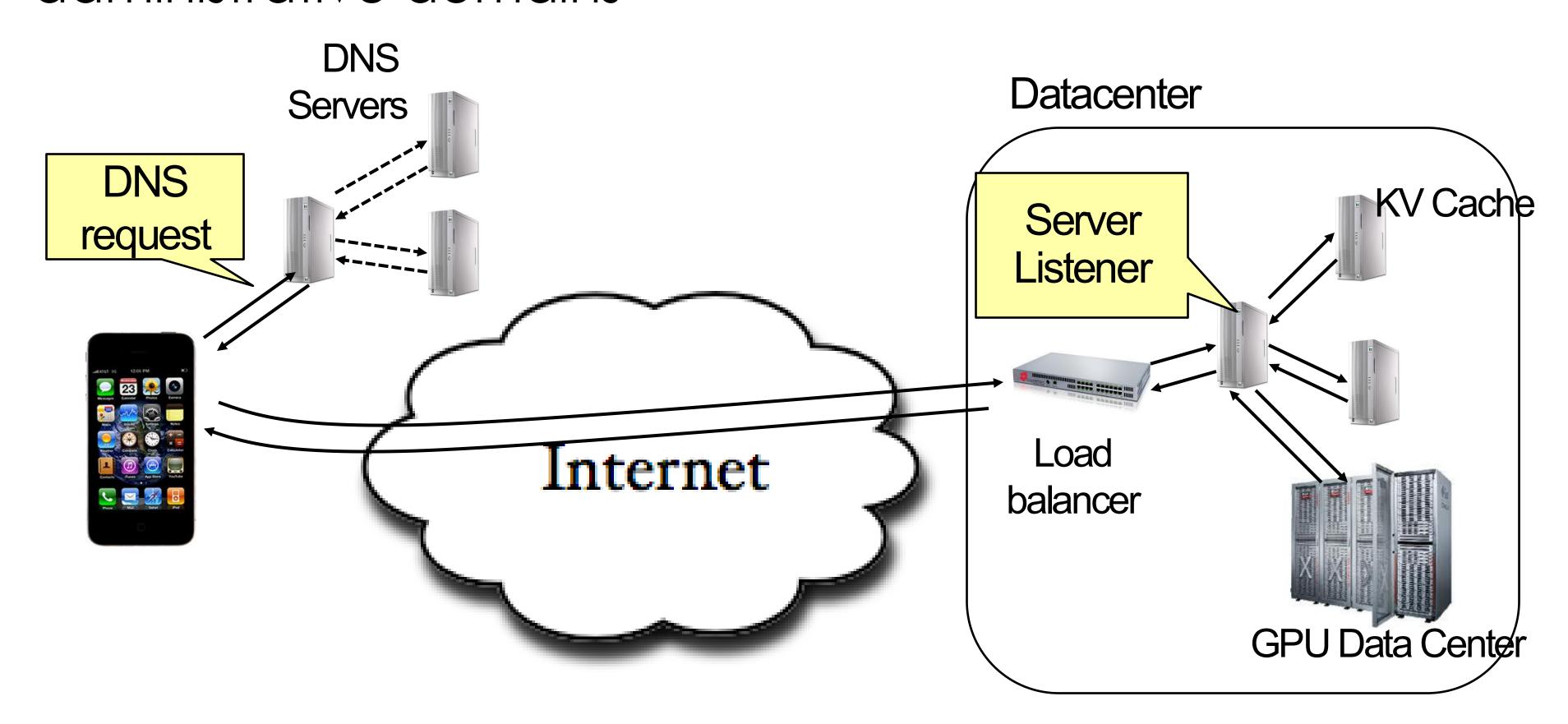


# Part 2: Cloud Computing and Distributed Systems

- Intro to Cloud Compute
- Networking Basics
- Collective Communication

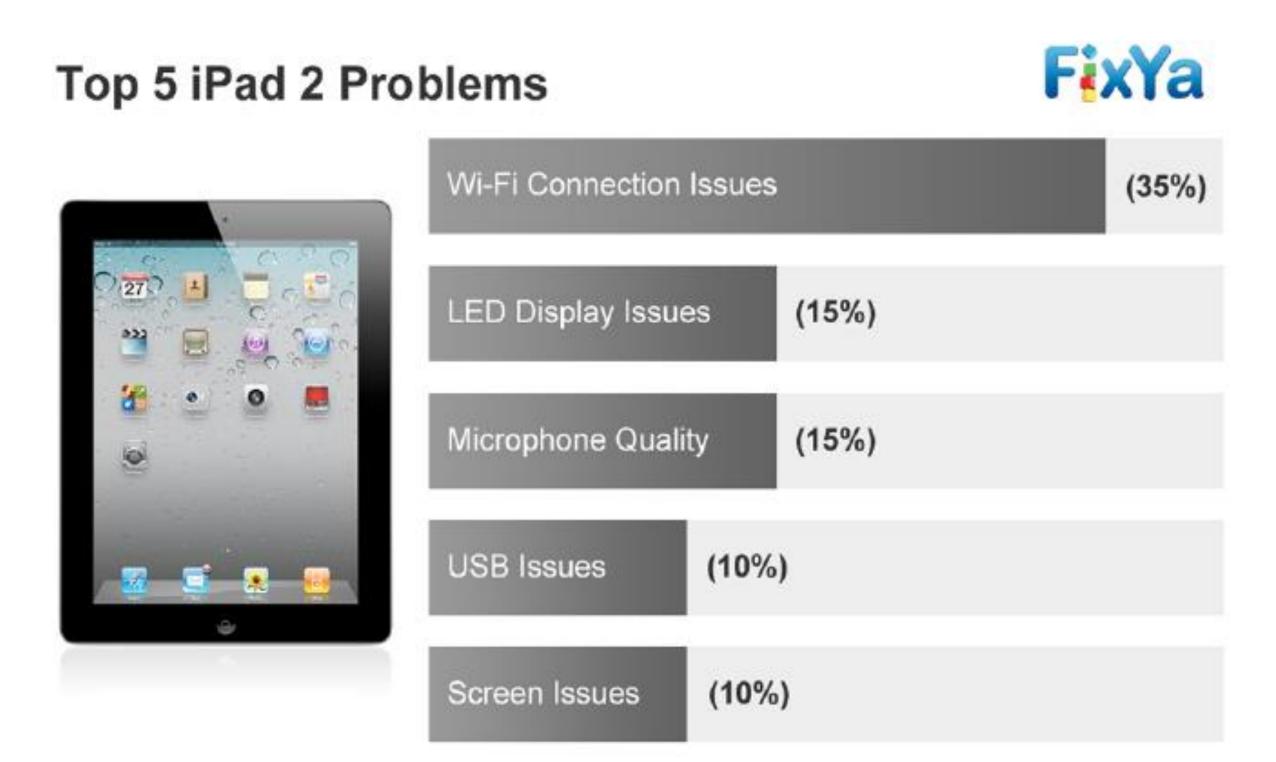
# Example: What's in a ChatGPT Query?

 Complex interaction of multiple components in multiple administrative domains



# Why is Networking Important?

- Virtually all apps you use comm
  - Many times main functionalit services, Amazon, Facebook
- Thus, connectivity is key service
  - Many times, connectivity issu
  - A data center is down -> net
- Some of the hottest opportuniti
  - Datacenter networking
  - OSes for Software Defined Networks (SDNs)
  - Networking for GPU Data Centers

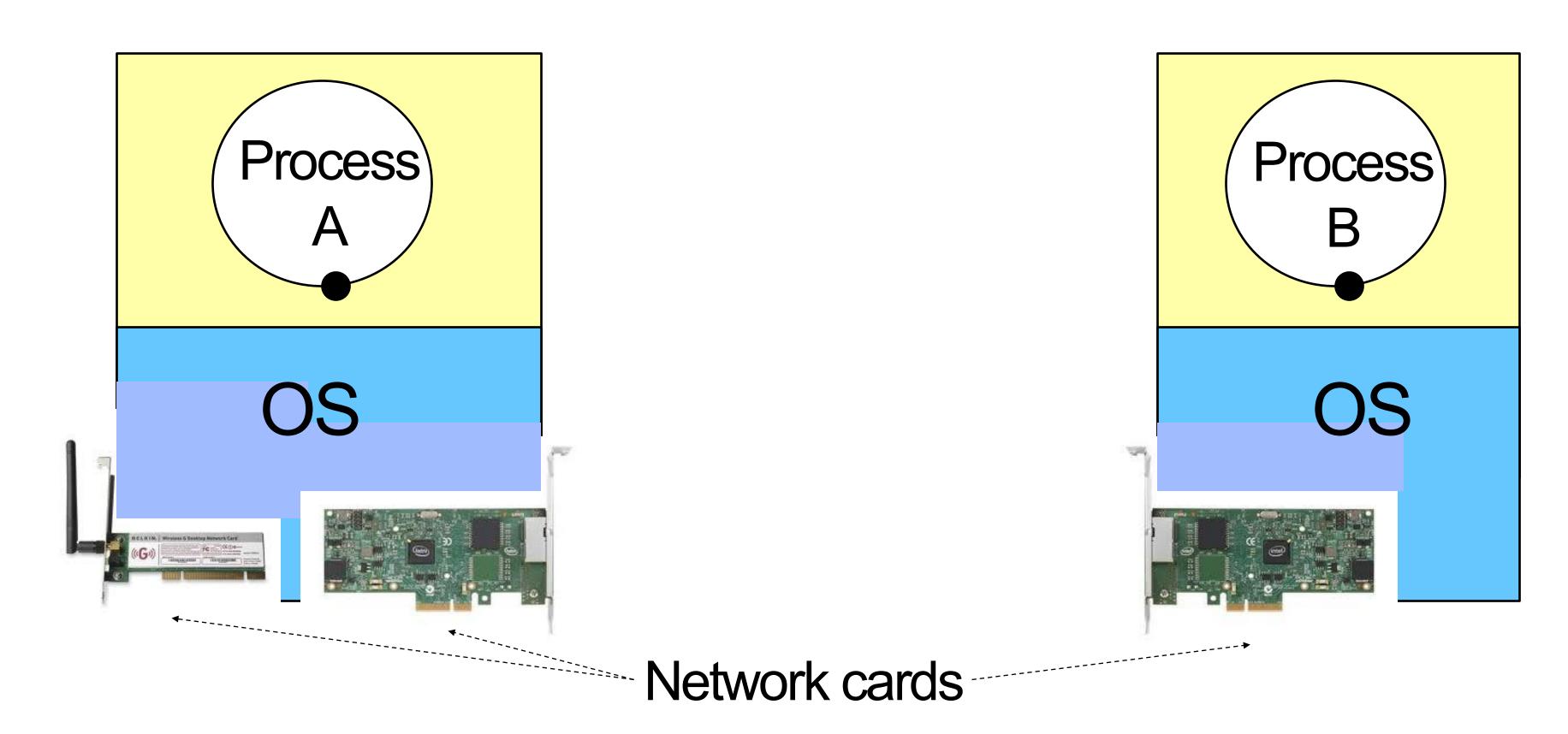


# Networking

- Network Basics
- Collective Communication

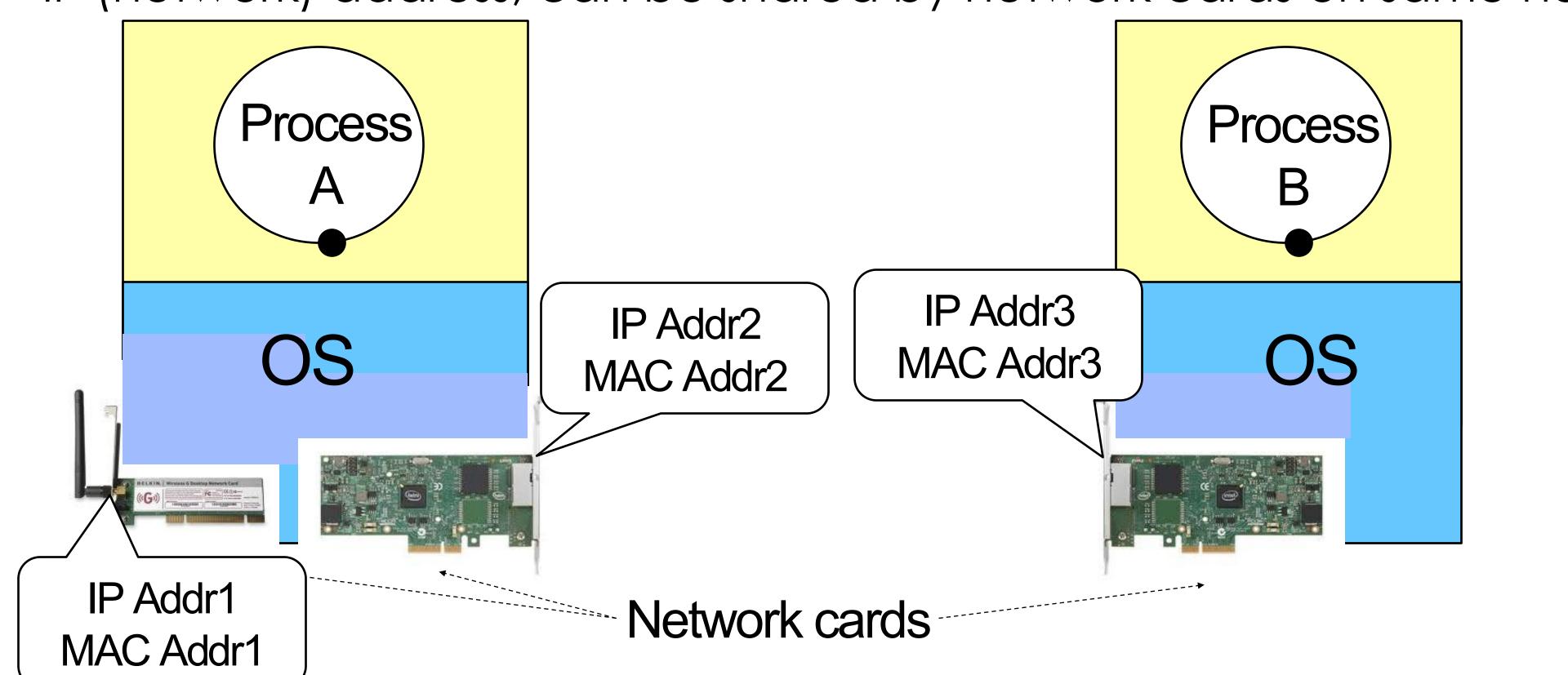
### Network Hardware

 Network (interface) card/controller: hardware that physically connects a computer to the network



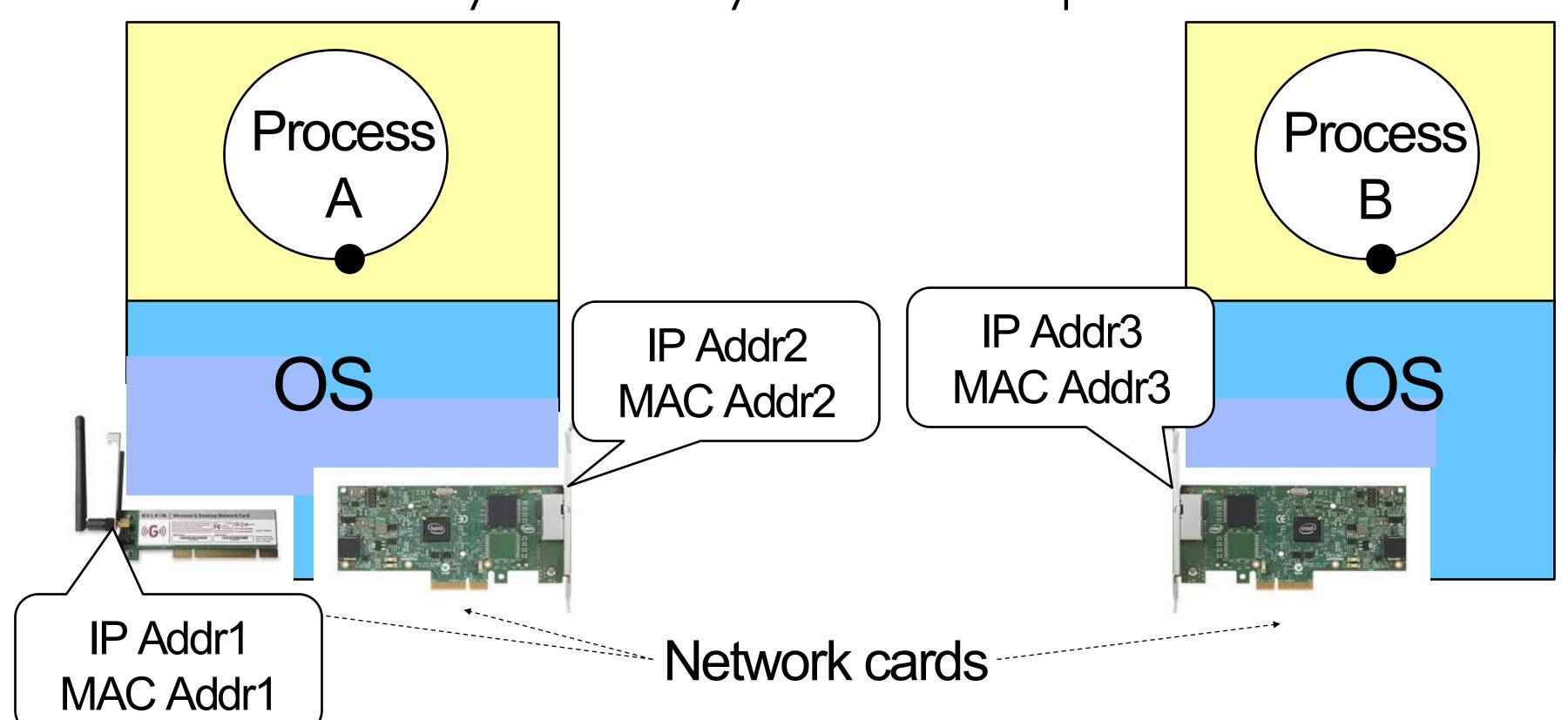
### Network Addresses

- Typically, each network card is associated two addresses:
  - Media Access Control (MAC), or physical, address
  - IP (network) address; can be shared by network cards on same host



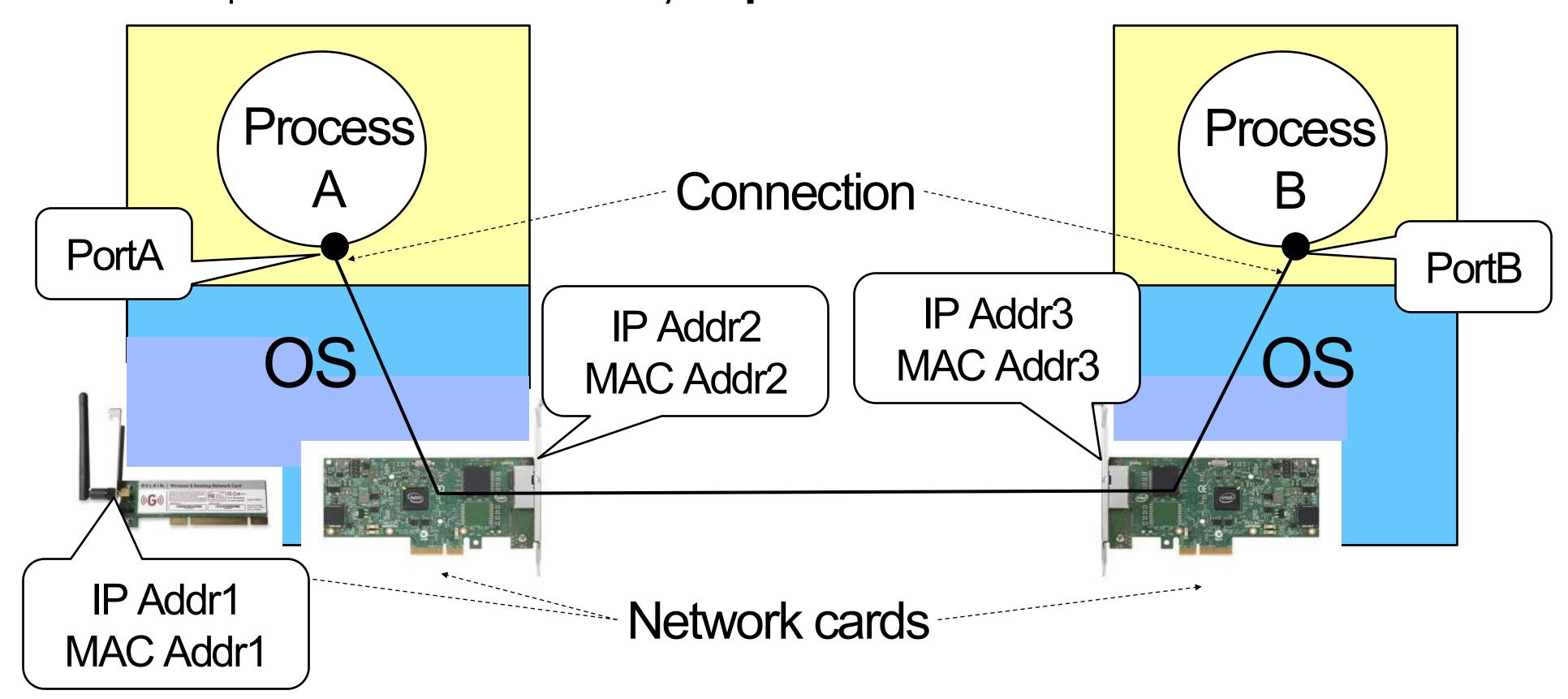
# Network Addresses (cont'd)

- 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 Identifier (cont'd)

- Connection: communication channel between two processes
- Each endpoint is identified by a port number



# Common Port Numbers

Application	Port number
Wake-on-LAN	9
FTP data	20
FTP control	21
SSH	22
Telnet	23
DNS	53
HTTP	80
SNMP	161

### Abstract Network

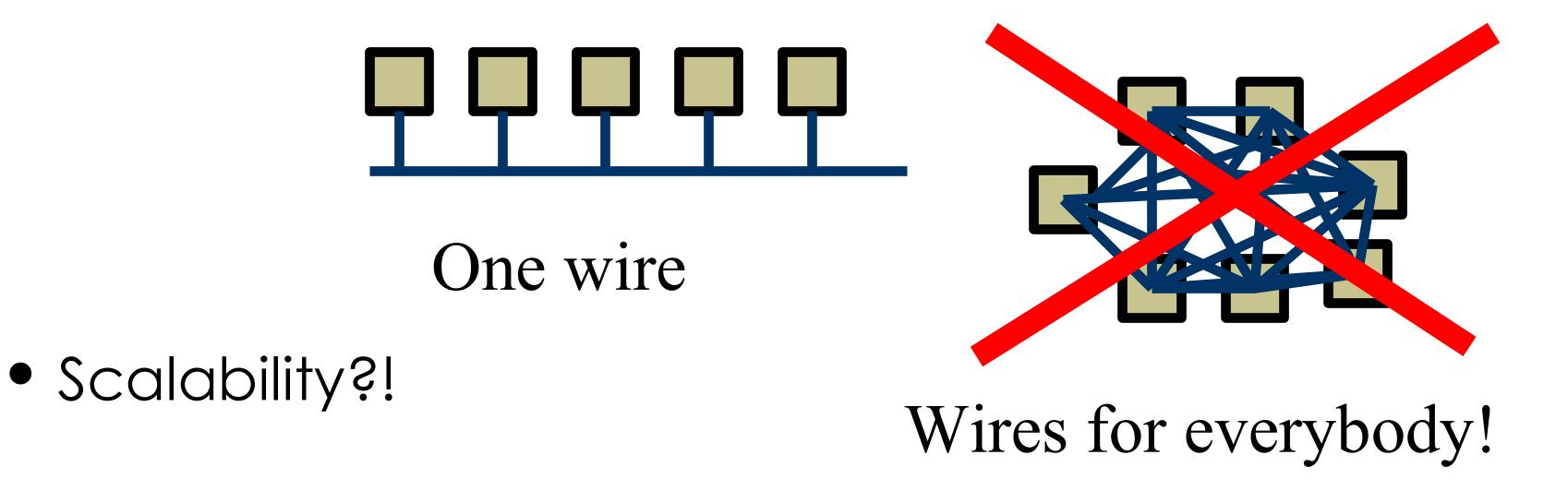


- Electrical questions
  - Voltage, frequency, ...Wired or wireless?
- Link-layer issues: How to send data?
  When to talk can either side talk at once?

  - What to say low-level format?

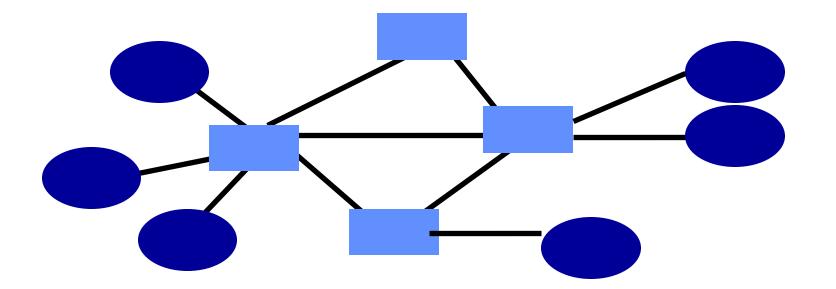
# Basic Building Block: Links

• ... But what if we want more hosts?



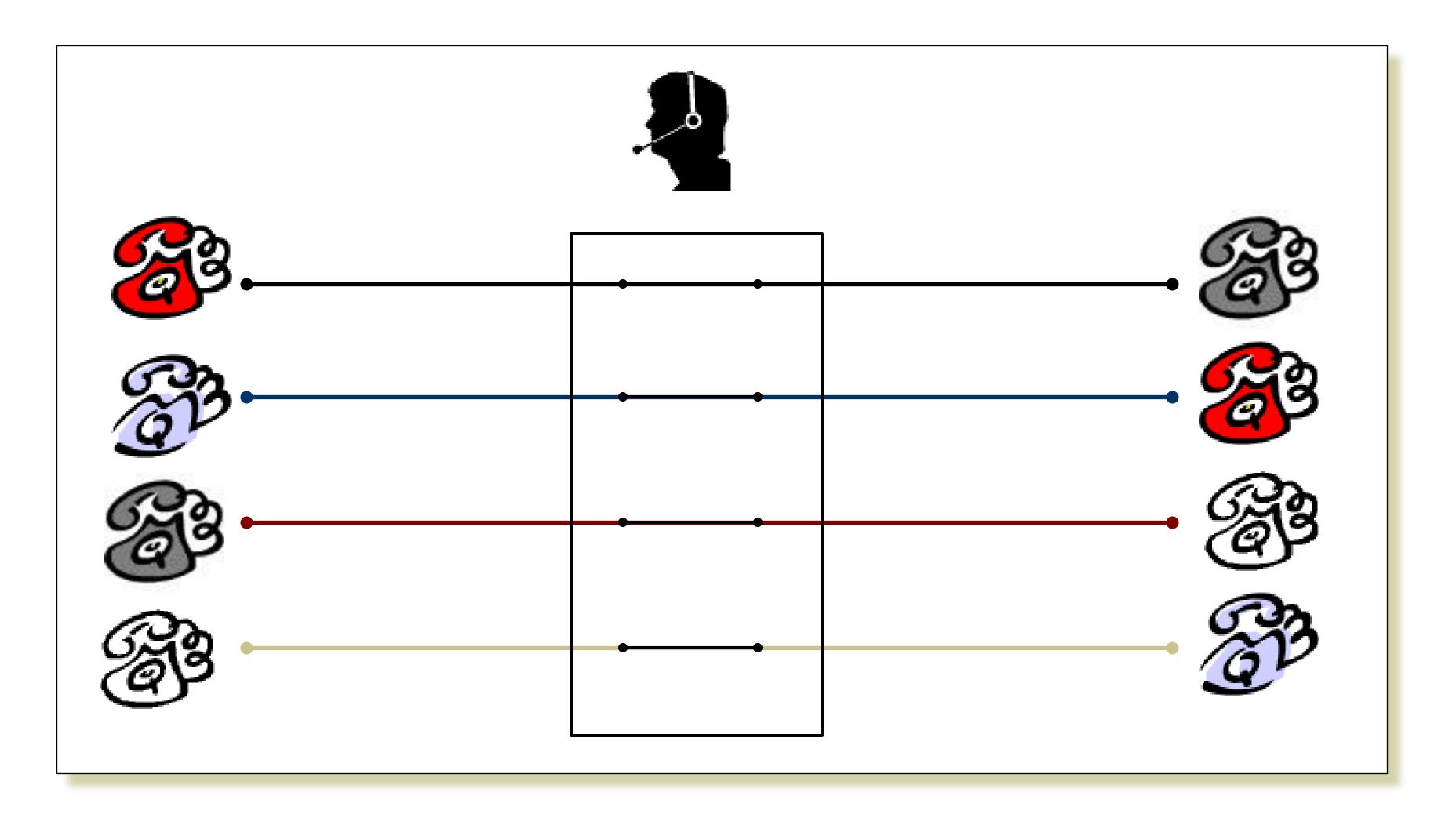
# Idea: Multiplexing

Need to share network resources



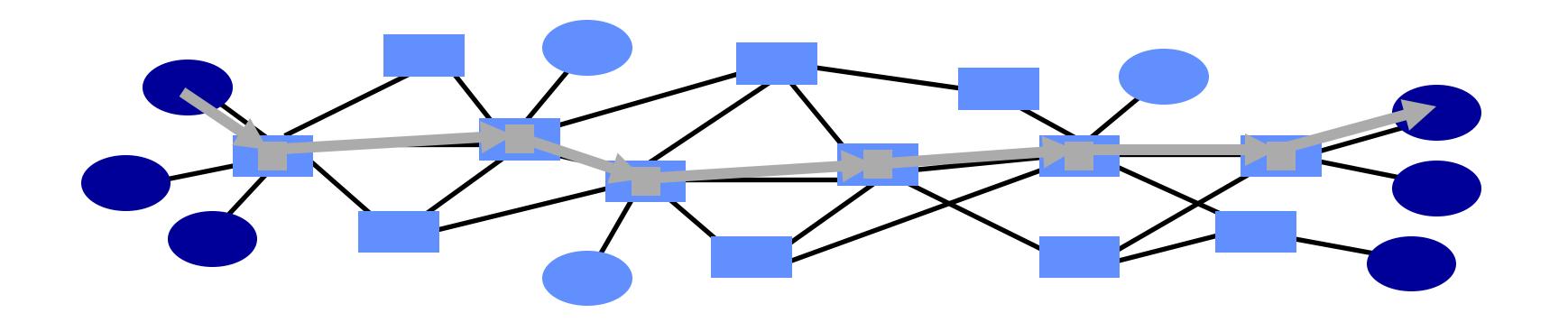
- How? Switched network
  - Party "A" gets resources sometimes
    Party "B" gets them sometimes
- Interior nodes act as "Switches"
- What mechanisms to share resources?

# In the Old Days...Circuit Switching



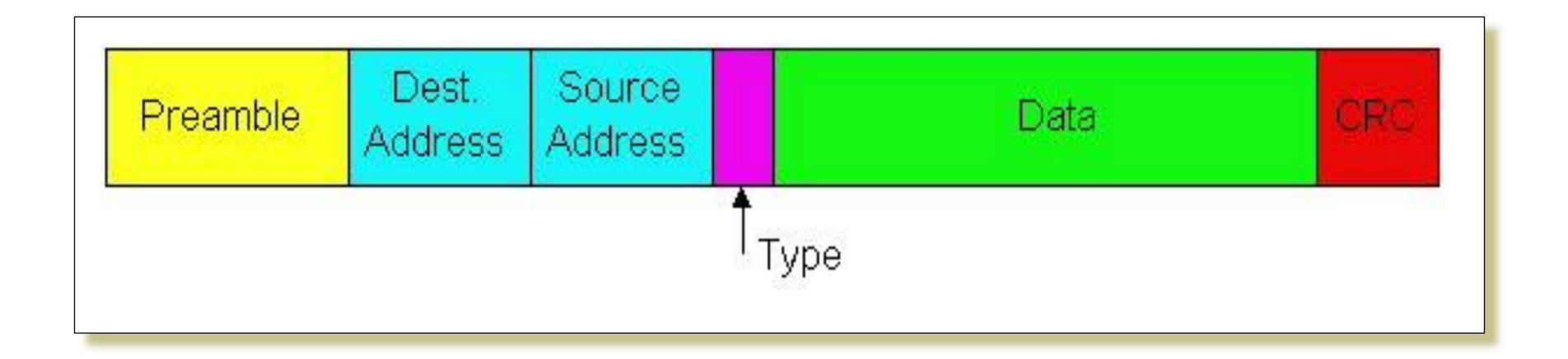
# Packet Switching

- Source sends information as self-contained packets that have an address.
  - Source may have to break up single message in multiple packets
- Each packet travels independently to the destination host.
  Switches use the address in the packet to determine how to forward the
  - packets
  - Store and forward
- Analogy: a letter in surface mail.

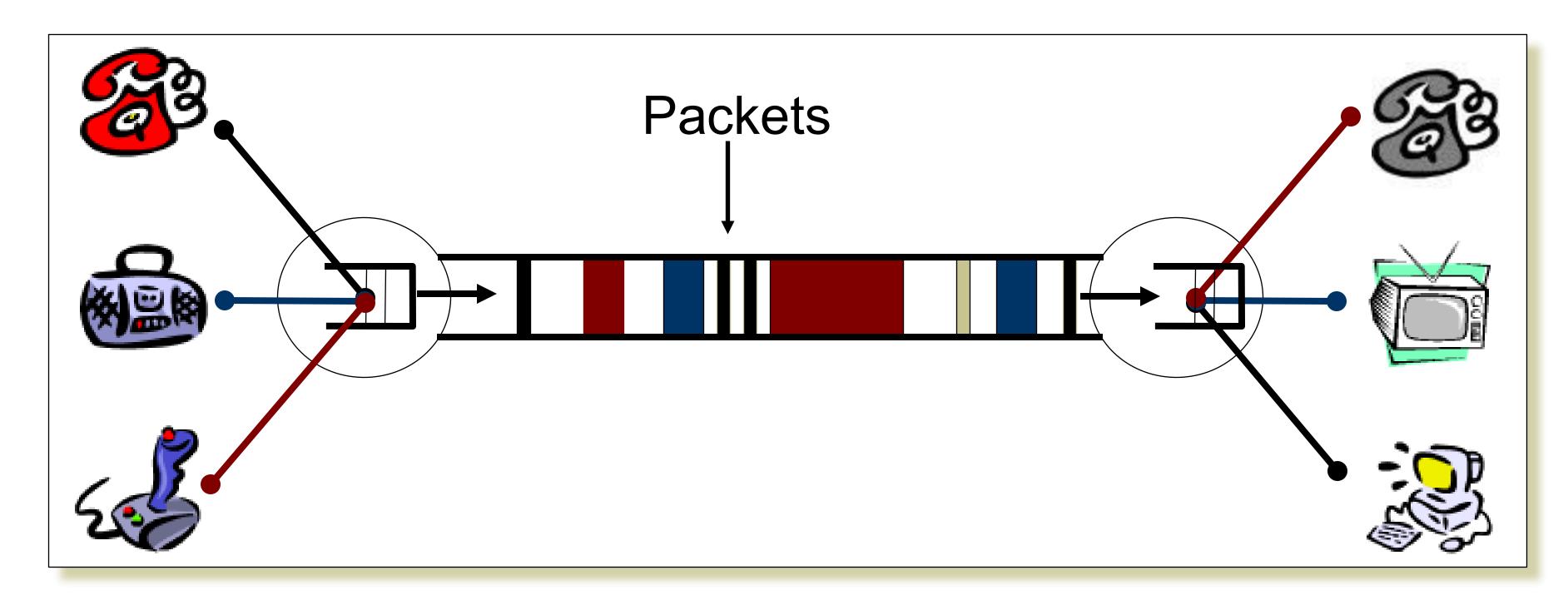


# Example: Ethernet Packet

 Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

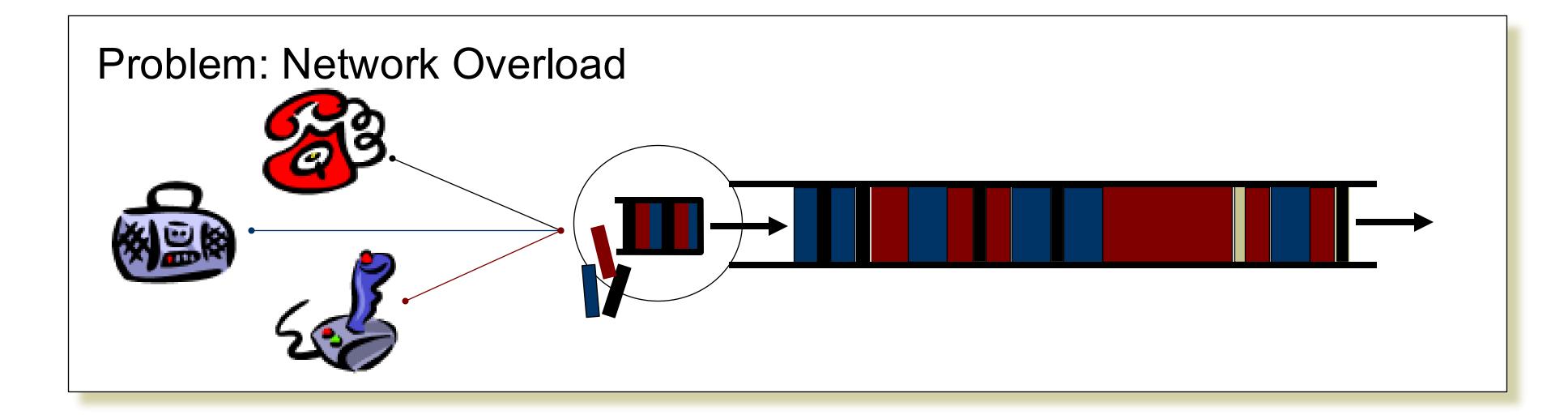


# Packet Switching



- Switches arbitrate between inputs
- Can send from any input that's ready
  - Links never idle when traffic to send
  - (Efficiency!)

#### What if Network is Overloaded?



#### Solution: Buffering and Congestion Control

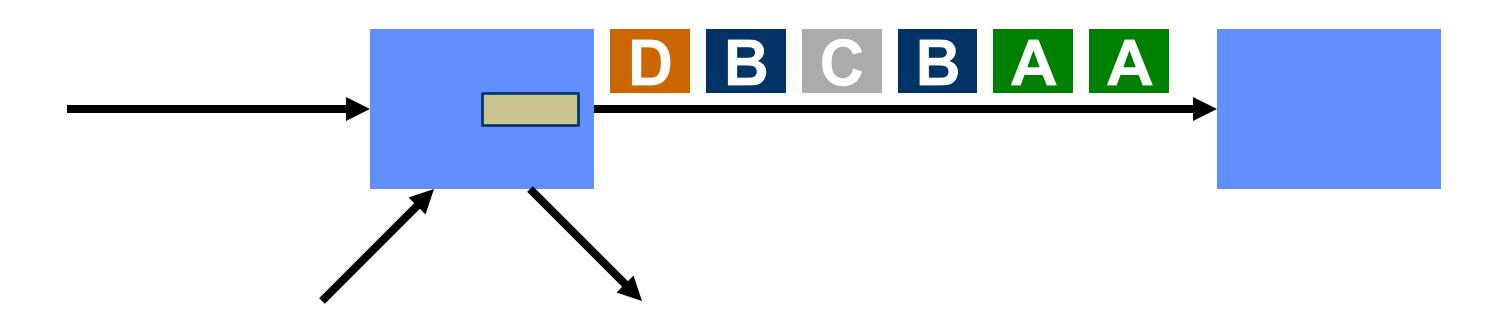
- Short bursts: buffer
- What if buffer overflows?
  - Packets dropped
  - Sender adjusts rate until load = resources > "congestion control"

## Characterizing Network Communication

- Latency how long does it take for the first bit to reach destination
- Capacity how many bits/sec can we push through? (often termed "bandwidth")
- Jitter how much variation in latency?
- Loss / Reliability can the channel drop packets?
- Reordering

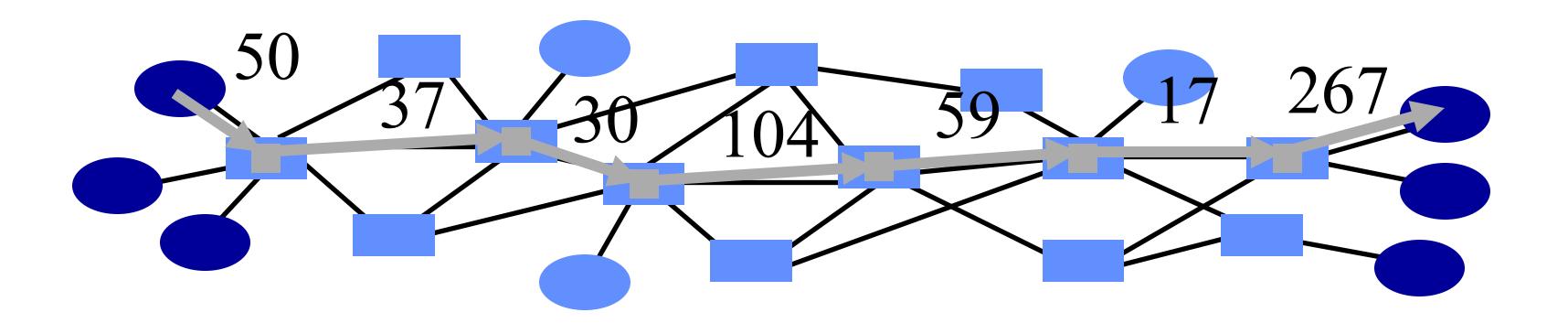
## Packet Delay

- Sum of a number of different delay components:
- Propagation delay on each link.
  Proportional to the length of the link
- Transmission delay on each link.
  Proportional to the packet size and 1/link speed
- Processing delay on each router.
  - Depends on the speed of the router
- Queuing delay on each router.
  Depends on the traffic load and queue size



## Throughput

- When streaming packets, the network works like a pipeline.
  All links forward different packets in parallel
- Throughput is determined by the slowest stage.
  Called the bottleneck link
- Does not really matter why the link is slow.
  - Low link bandwidth
  - Many users sharing the link bandwidth

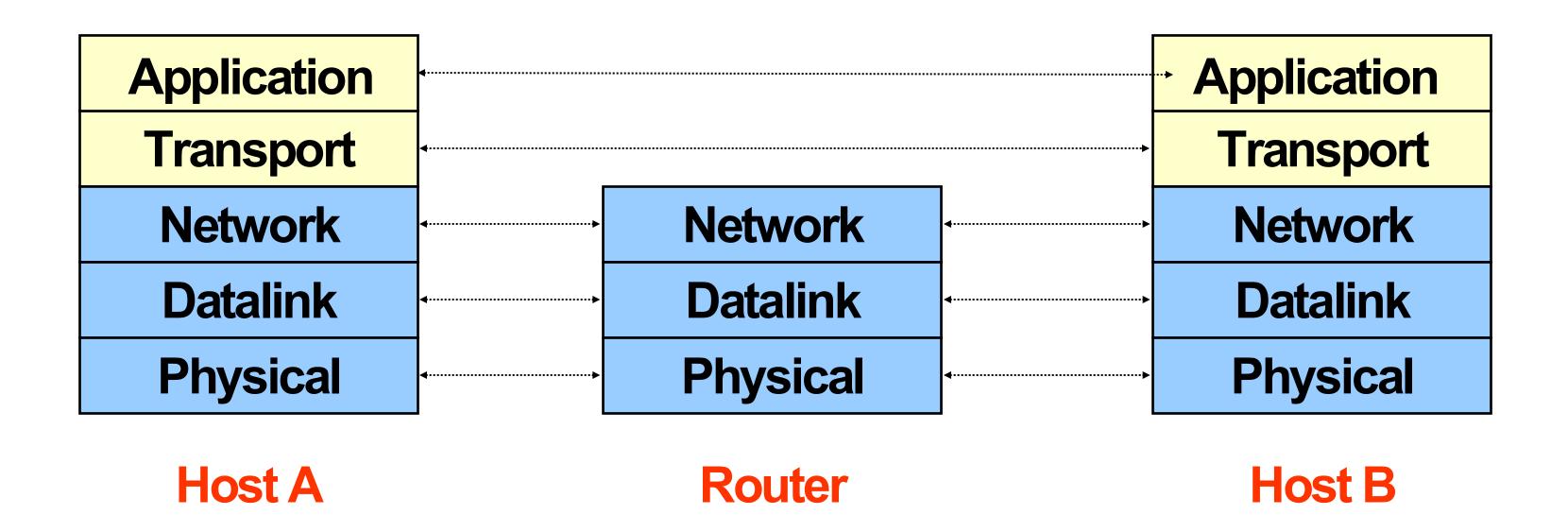


# Some simple calculations (mbps/kbps)

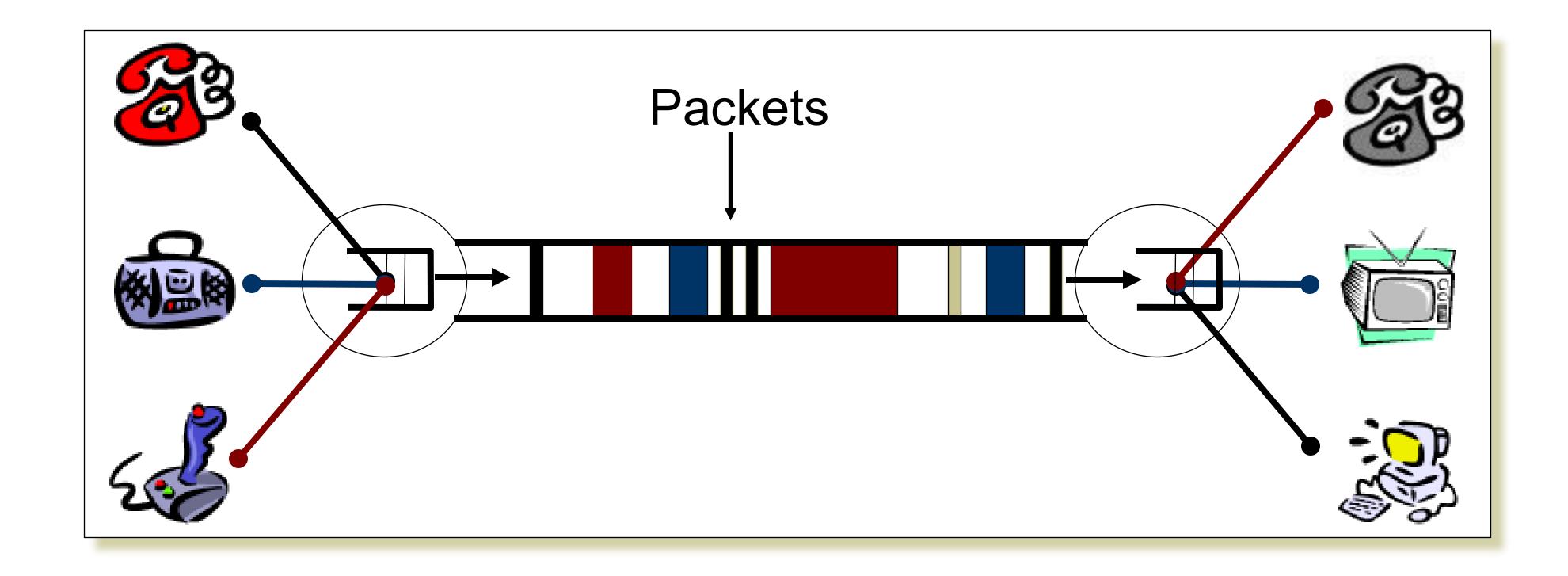
- Cross country latency
  - Distance/speed =  $5 * 10^6 m / 2x10^8 m/s = 25 * 10^3 s = 25 ms$
  - 50ms RTT
- Link speed (capacity) 100Mbps
- Packet size = 1250 bytes = 10 kbits
  - Packet size on networks usually = 1500 bytes across wide area or 9000 bytes in local area
- 1 packet takes
  - 10k/100M = .1 ms to transmit
  - 25ms to reach there
  - ACKs are small → so 0ms to transmit
  - 25ms to get back
- Effective bandwidth = 10kbits/50.1ms = 200kbits/sec

#### Layers

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



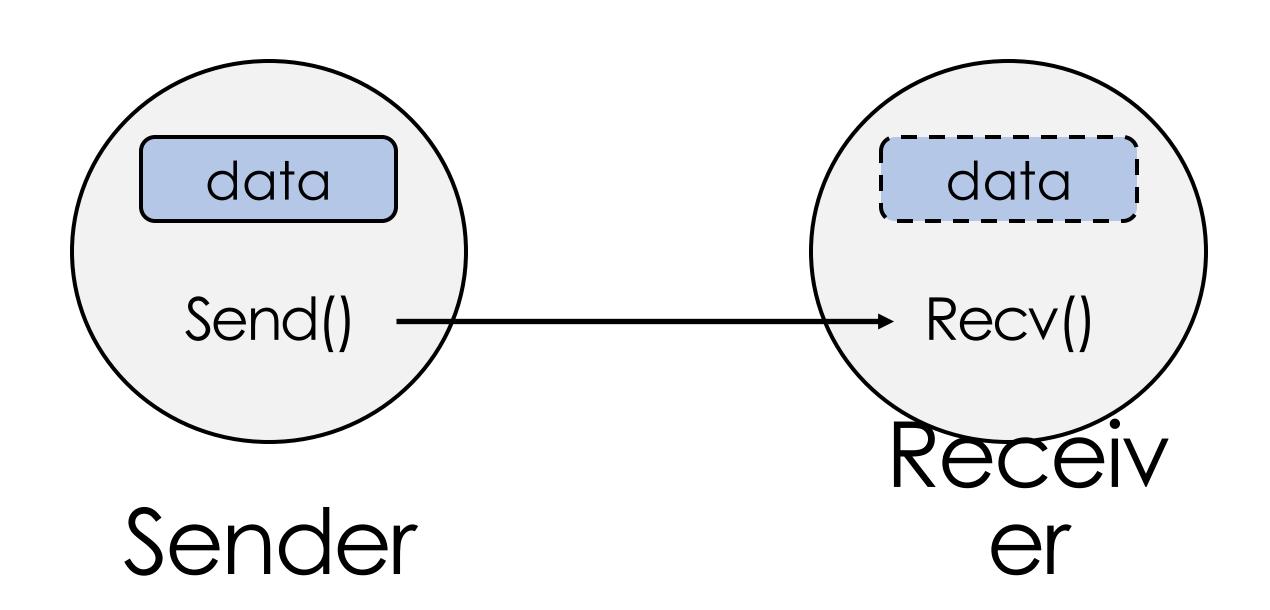
## Communication



# Communication: Point-to-point communication



## Program P2P communication: Very Simple in Ray



```
def send(array: np.array):
    # Running on sender processs
    ref = ray.put(array)
    return ref

def receive(ref):
    # Running on receiver process
    array = ray.get(ref)
    print(array)
```

#### Case study: Gradient update in DL

Gradient / backward computation

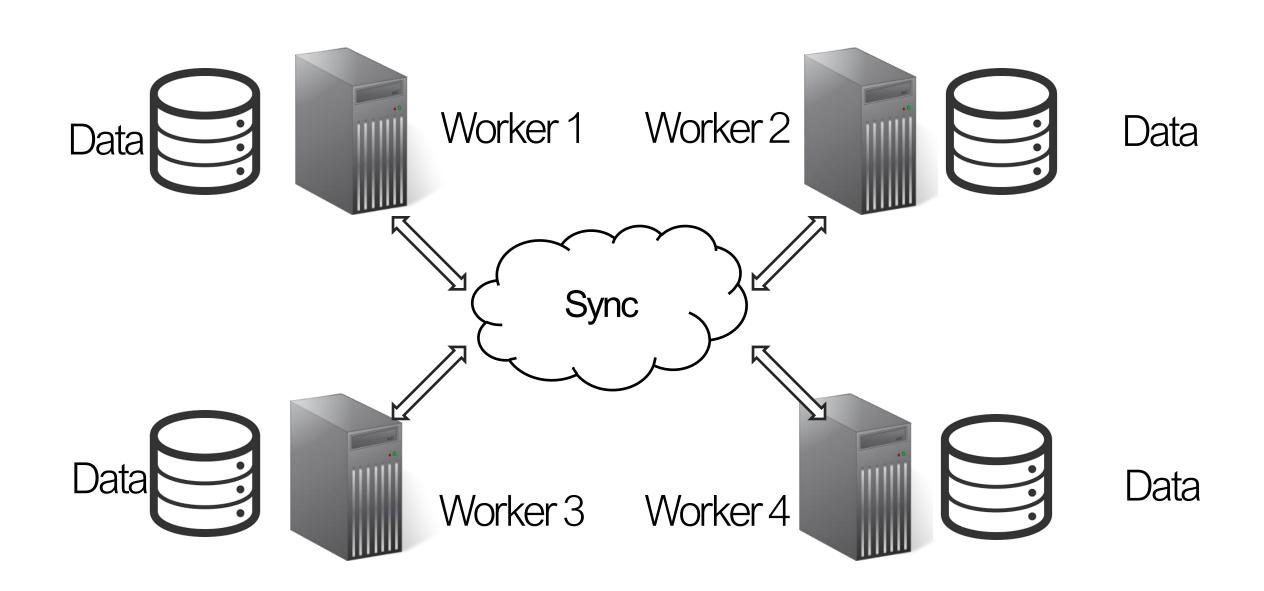
$$oldsymbol{ heta}^{(t)} = oldsymbol{ heta}^{(t-1)} + egin{array}{c} oldsymbol{ heta} \cdot 
abla_{\mathcal{L}}(oldsymbol{ heta}^{(t-1)}, oldsymbol{D}^{(t)}) \ & oldsymbol{ heta} & oldsymbo$$

What If Data is super Big?

# What if Data is Super Big?

Big Data





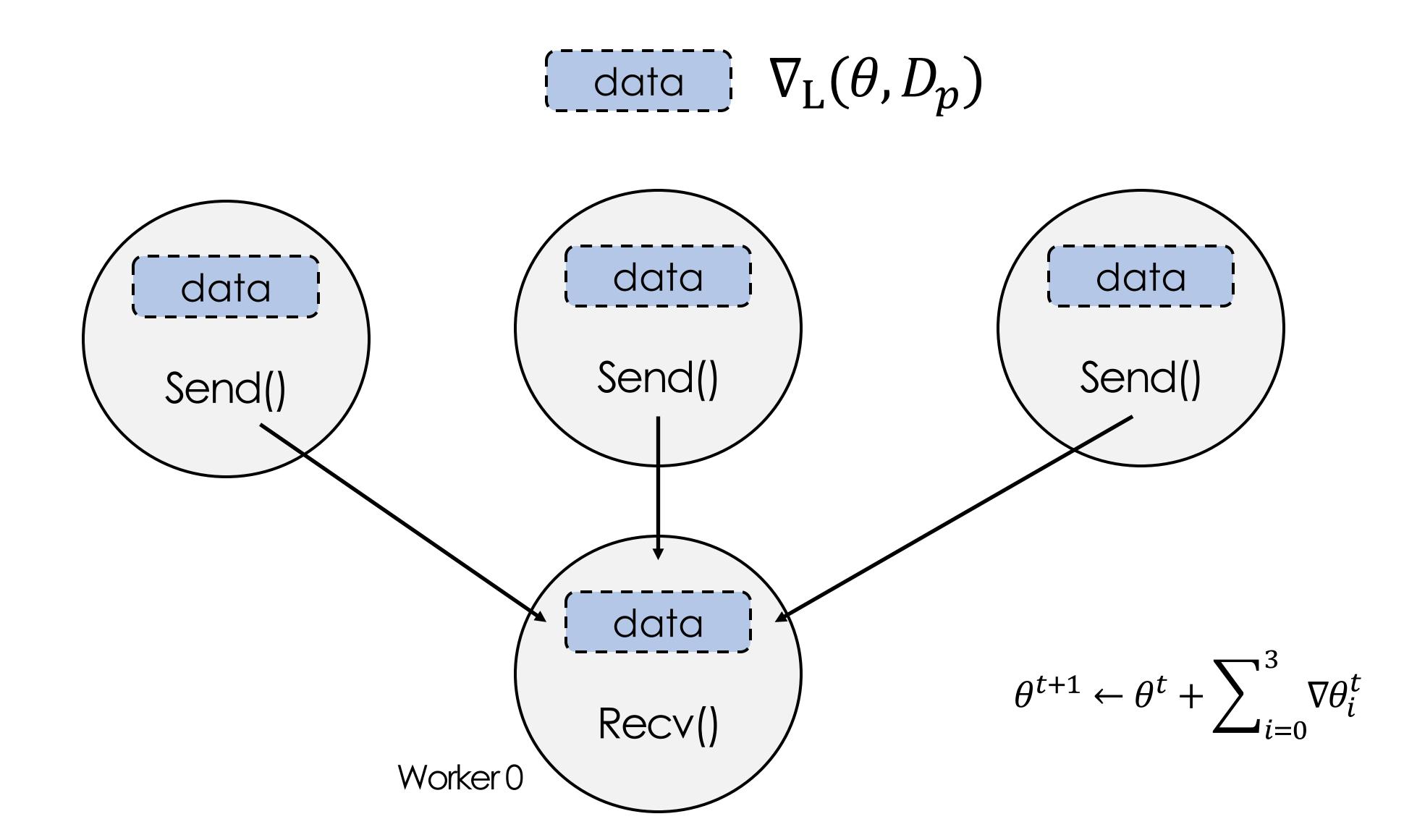
#### Case study: Gradient update

Gradient / backward computation

$$oldsymbol{ heta}^{(t)} = oldsymbol{ heta}^{(t-1)} + oldsymbol{arepsilon} oldsymbol{ heta} \cdot 
abla_{\mathcal{L}} (oldsymbol{ heta}^{(t-1)}, D^{(t)})$$

$$m{ heta}^{(t+1)} = m{ heta}^{(t)} + m{arepsilon} \sum_{p=1}^P 
abla_{\mathcal{L}}(m{ heta}^{(t)}, D_p^{(t)})$$
How to perform this sum?

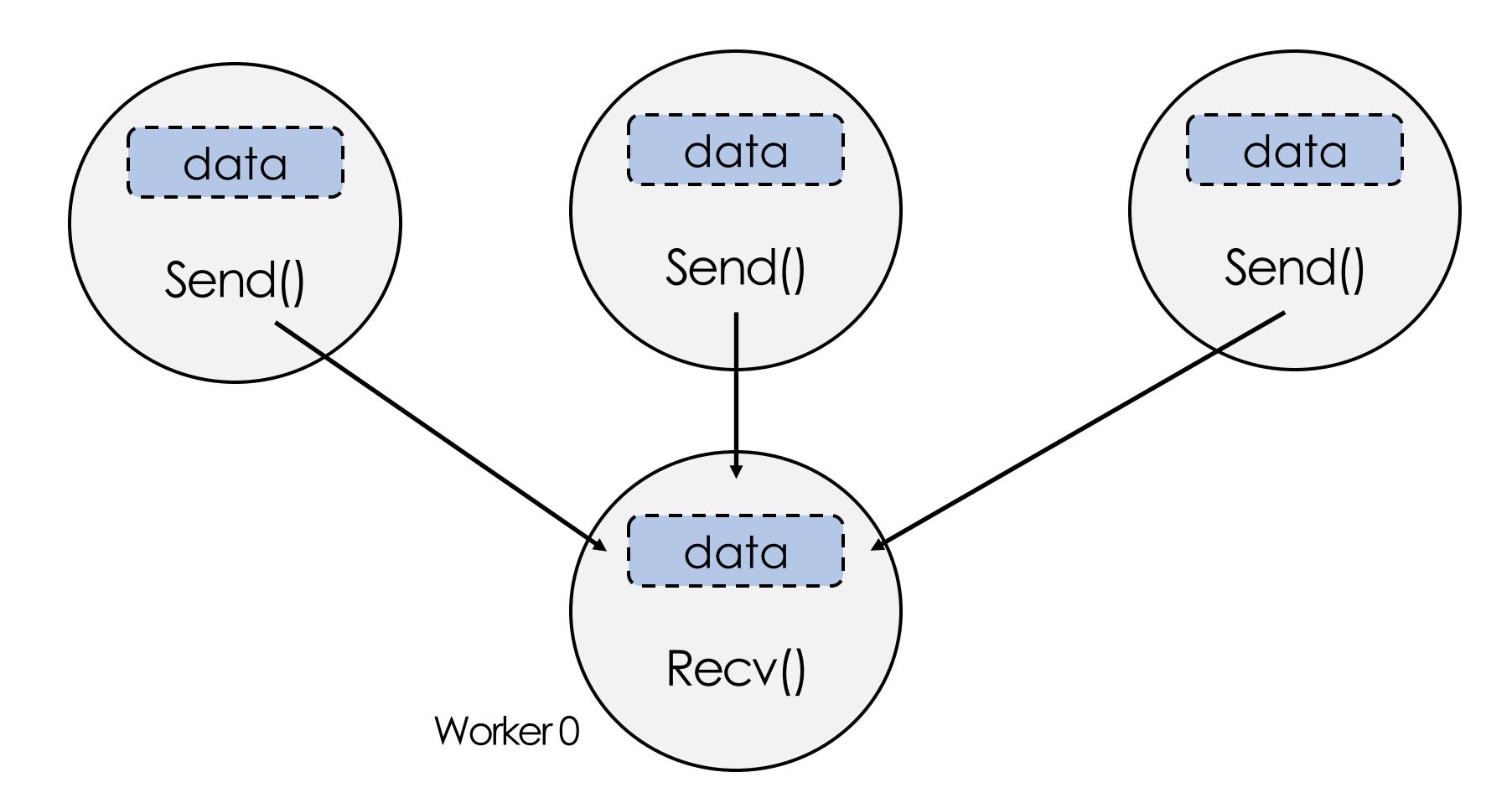
#### Collective Primitive: Reduce



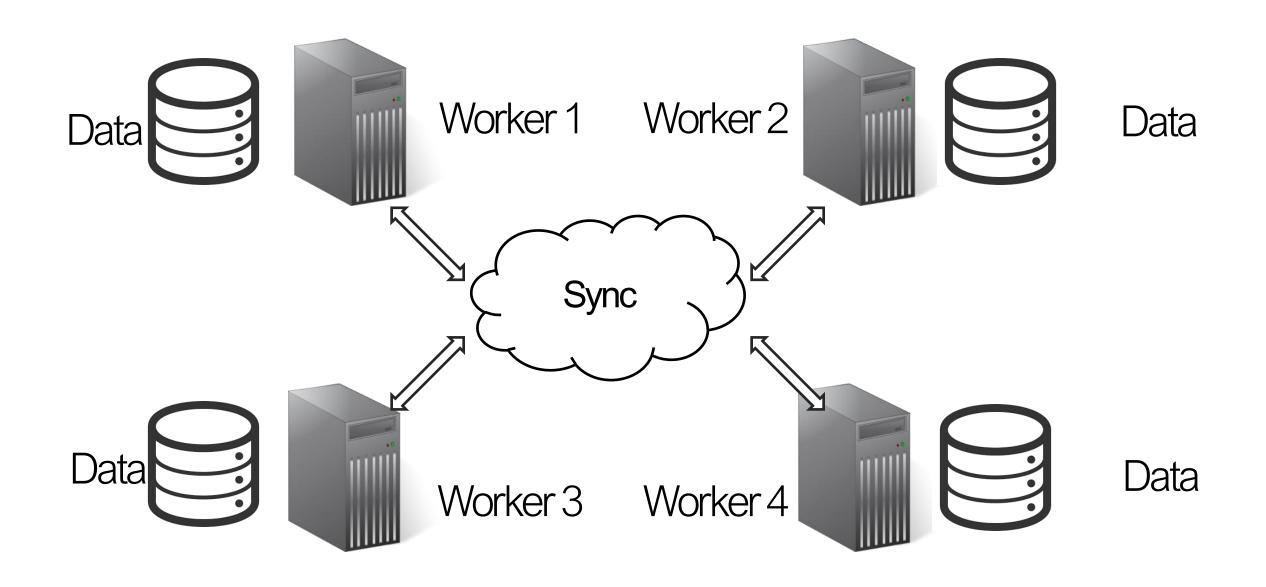
Program This? Will be in PA2!

## Analyze Performance

- Message over networks: 3\*N.
- Can we do better?



#### Not Yet Finished: Synchronization



For each worker

$$m{ heta}^{(t+1)} = m{ heta}^{(t)} + m{arepsilon} \sum_{p=1}^P 
abla_{\mathcal{L}}(m{ heta}^{(t)}, D_p^{(t)})$$
How to perform this sum?

#### Problem: We need All-Reduce

