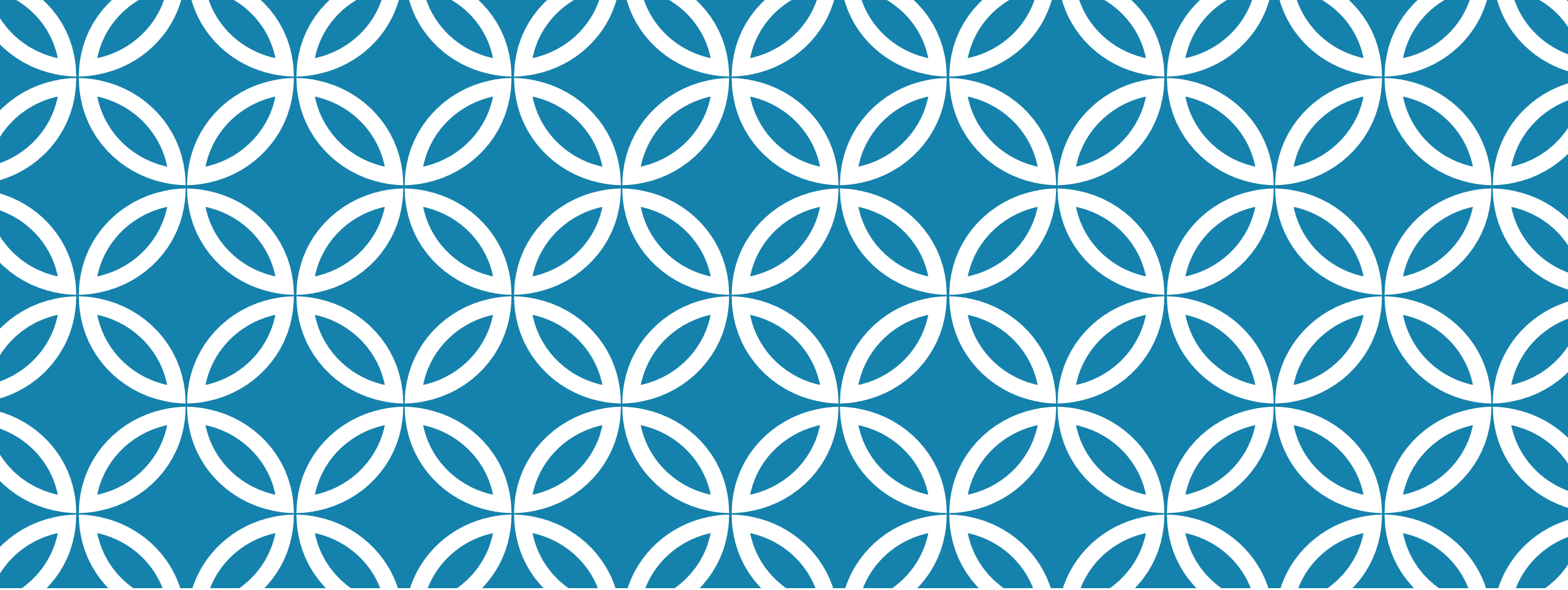
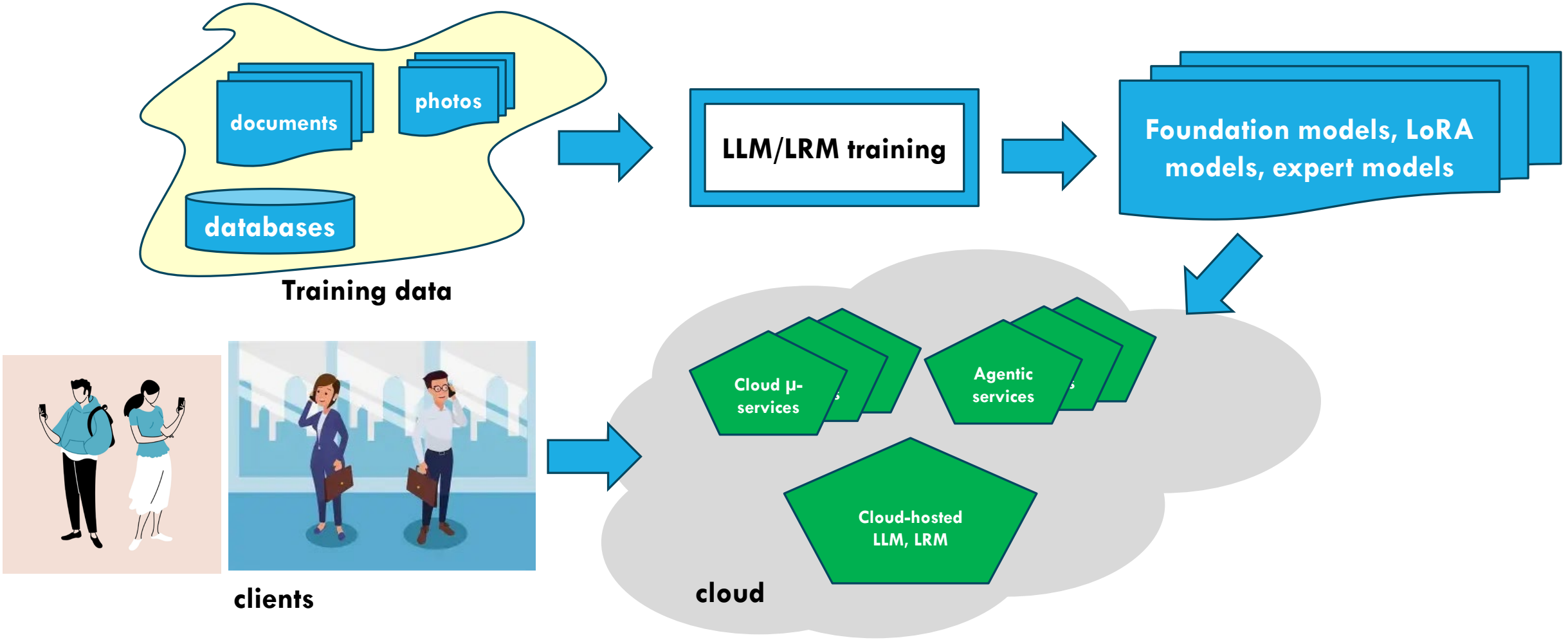


HOW CLIENTS CONNECT TO CLOUD PLATFORMS AND AGENTS

Professor Ken Birman

CS4414/5416 Lecture 16

TODAY'S LLM/LRM ECOSYSTEM



WE KNOW ALL ABOUT PERFORMANCE ON A SINGLE NUMA MACHINE… BUT THE CLOUD IS MORE!

Starting a new unit for the course

Our focus will be on:

- Networking, connectivity to the cloud
- Understanding the cloud itself: An immense collection of tools offered as "services". Applications use web APIs but leverage these tools, and ML will be a service, too.
- We'll look at two examples of popular services from Facebook to motivate our deeper dive into how such things work.

IDEA MAP FOR TODAY

The Internet, IP addresses and port numbers.
Packets, routing, firewalls, tunnels, network address translation

TCP basics, SSL security

Socket API, Google GRPC

http versus https, VPNs, VPC

INTERNET BASICS

The internet is like a computerized postal system.

Any process can set up a “mailbox” (a **socket**), and post an address on it (**bind** an IP address and port number).

The address can then be registered for use by programs anywhere on the net… with limitations

IP ADDRESSES

Mailroom at 11 Riverside Drive, NYC

IPv4 is 32-bits (but only 28 are useable). IPv6 doubles this.

- There are actually several types of addresses (classes)
- In CS4414 we won't dive into why, or what the others are for

Each address has an associated "port number" because one machine could have many processes using the network. Like a mailbox in an apartment building.

- The IP address is used to route to the computer. Like a street address.
- The port number is used to figure out which process gets the packet.

A SINGLE MACHINE CAN HOST MANY NETWORK SERVICES

This is the role of the port number

As if each each service operates from its own apartment

Some port numbers are standard, like 80 (for web services). Others are allocated on demand and look like random integers

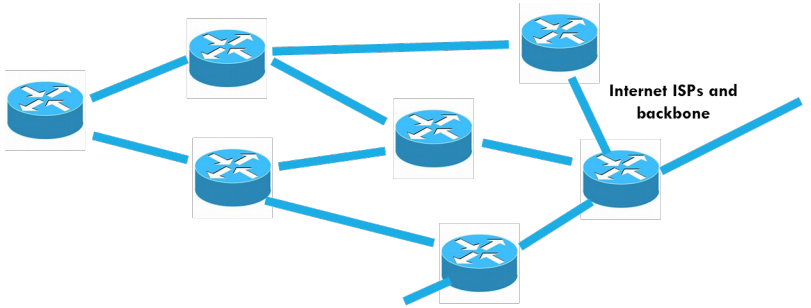
ADDRESS DIRECTORY: DNS

The domain name service (DNS) map server names (www.cornell.edu) to IP addresses (128.253.173.247) and port #

The DNS is operated by for-profit companies. They sell domain addresses, and server owners pay for listings.

Fancy web sites, like Netflix.com, have ways to route you to a data center somewhere near you, for speed.

ROUTERS AND LINKS



The Internet is composed of high-speed network links between routers and switches.

- A **router** takes an incoming packet and looks up the routing rule for sending it to the specified destination.
- Then it forwards the message out on the corresponding link.
- **Switches** are seen in clusters or racks of computers. Unlike a router, which can adapt the route over time, a switch uses fixed routing.

Packets have some maximum size, like 8KB. A "message" from process to process would often be far larger and will be sent as a series of packets.

MAXIMUM PACKET SIZE VARIES

In the wide-area Internet, 1400 bytes for historical reasons. By now this feels too small, but it isn't easy to change...

In a local area network, 8KB is more typical.

In a datacenter you can switch to "fat packets" like 64KB or sometimes, even larger.

NETWORK ADDRESS TRANSLATION

The “space” of addresses is much too small to cover the entire globe, so it evolved into a world of overlapping regions that use their own addresses. Like when a wifi router uses 192.168.0.xxx

Network address translators dynamically modify the (ip-address, port-no) information in packet headers to implement this (they might also block some packets: “firewall”)

NETWORK ADDRESS TRANSLATION, FIREWALLS, ROUTERS, WIFI…

192.168.0.3 port 7653

3752.412.88.16 port 211

WiFi + NAT box + firewall

Internet ISPs and backbone

154.771.21.71 port 73151

NAT box + firewall

Fortnight Server

192.168.364.778 port 921

Amazon AWS Cloud (Data Center)

NETWORK ADDRESS TRANSLATION, FIREWALLS, ROUTERS, WIFI…

192.168.0.3 port 7653

3752.412.88.16 port 211

WiFi + NAT box + firewall

Internet ISPs and backbone

154.771.21.71 port 73151

NAT box + firewall

Fortnight Server

192.168.364.778 port 921

Amazon AWS Cloud (Data Center)

Routes picked to be economical, not necessarily fast or short!

NETWORK ADDRESS TRANSLATION, FIREWALLS, ROUTERS, WIFI…

192.168.0.3 port 7653

3752.412.88.16 port 211

WiFi + NAT box + firewall

Internet ISPs and backbone

154.771.21.71 port 73151

NAT box + firewall

Fortnight Server

192.168.364.778 port 921

Amazon AWS Cloud (Data Center)

Could use a different route for traffic in return direction

ROLES PLAYED BY FIREWALLS

They block unexpected "incoming" traffic. ***Configurable policy.***

Permit "outgoing" traffic, such as http (TCP) connections to remote web services and other permitted services. Those use a known port no.

Apply network address translation, if desired.

* Map any outgoing messages to show that the sender is using the IP address of the router itself. Assign a new port number if this is a TCP connection request.

* Fix the packet checksum, to match these modified field. Later, incoming traffic to this IP address/port will be mapped back

By default, the policy often blocks everything! You need to configure the firewall policy to enable connectivity

ROLES PLAYED BY FIREWALLS

They block unexpected "incoming" traffic. ***Configurable policy.***

Permit "outgoing" traffic, such as http (TCP) connections to remote web services and other permitted services. Those use a known port no.

Apply network address translation, if desired.

- Map any outgoing messages to show that the sender is using the IP address of the router itself. Assign a new port number if this is a TCP connection request.
- Fix the packet checksum, to match these modified field. Later, incoming traffic to this IP address/port will be mapped back

TUNNELS

Trainload of IP packets enters a tunnel….

Sometimes it is convenient to send data through some domain without that domain "seeing" the packet headers.

A tunnel is used in such cases. A connection is is still needed, but then packets are sent through it as "pure data".

On the far side, they exit the tunnel and get routed "normally"

A CRAZY PATCHWORK! BUT IT WORKS…

Internet routers are *fast* and speed of light is quite fast too.

… Should you care that your data went via Delaware?
… For the ISP, this route could be cheaper to operate, or use faster links and routers. The physically shortest path may be slower!

At each stage, a packet is routed to whatever router is next in the route for the particular IP address is happens to carry at that stage.

MISTAKES DO HAPPEN, BUT RARELY

Internet routers use “routing tables” that tell them where to send packets. They adapt to route around outages.

Sometimes, mistakes can happen, but this is rare.

Goal of routing: Pick the most cost-effective, reliable, highest performing path available.

MISTAKES DO HAPPEN, BUT RARELY

The Switch

How the Chinese Internet ended up in Cheyenne, Wyoming

The building on Pioneer Ave. that houses Sophidea, the company that received a deluge of Chinese Internet traffic Tuesday. (Google Streetview)

An ISP operator (somewhere very remote from China) accidentally miscoded one digit of a "fixed" route.

This disrupted global routing.

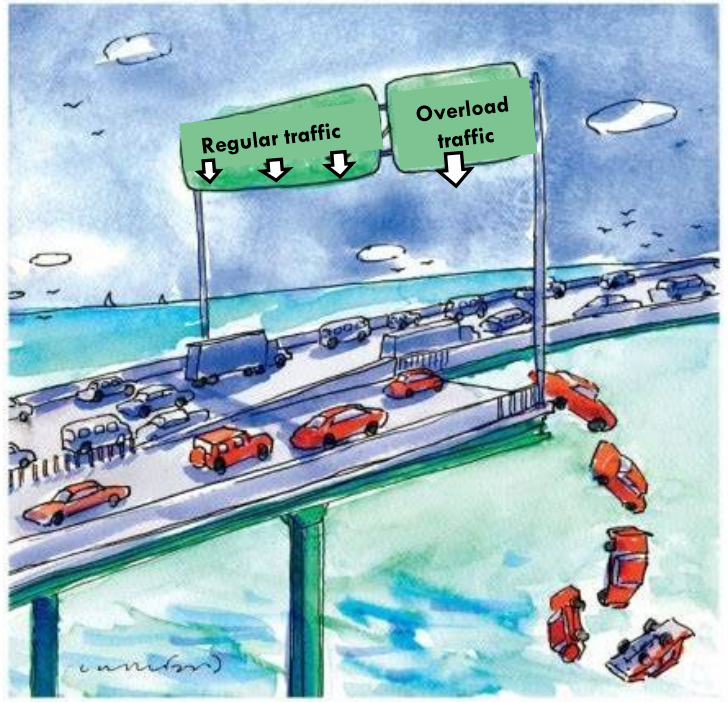
Traffic intended for a centralized router in Shenzhen was "redirected" to a small cafe in Wyoming, near Yellowstone Park…

… all the traffic, for much of China

WHAT DID THE CAFE DO WITH CHINA'S INTERNET TRAFFIC?

They were "dropped" (discarded, silently)

In fact, this is a *feature* of the Internet!



The Internet works like a network of highways and bridges, but this kind of bridge can just toss cars off if a traffic jam forms!

WHY DROP PACKETS? END-TO-END PRINCIPLE

In the early days of the Internet a debate arose: should the Internet be reliable, or is it ok to drop packets?

The "end to end" principle was this: **The Internet doesn't need to be reliable, as long as it is blindingly fast and *mostly* reliable.** The endpoints (sender, receiver) can provide reliability and flow-control.

Packet drop is common even though router/link failures are rare.

WHY DROP PACKETS? CONGESTION SIGNAL!

A second reason is to warn the sender that some router or link is overloaded on the route (network path) it is using.

Loss is a congestion signal. The sender finds out after a timeout.

It waits for acknowledgement but doesn't get one. Or some subsequent packets get through, and the receiver complains.

TCP STREAMS EMBODY THE END-TO-END RULE

Like a pipe, TCP sends a stream of bytes from A to B. Its jobs is to hide losses or data corruption or out-of-order data.

TCP has two "end points". The "end to end" principle makes these endpoints responsible for reliability and security.

TCP uses a *protocol* to achieve them: A scripted exchange of messages with a format that TCP imposes.

DATA LIVES INSIDE THESE TCP PACKETS

TCP has a header of its own, with information used by the end-to-end protocol operated by the TCP module in the Linux kernel.

In the kernel, data you send on a TCP stream is chopped into 1400-byte segments and carried as data within the IP packets it sends.

On arrival, TCP will reassemble and deliver the data, but you won't see the TCP headers: it removes them in the kernel protocol stack.

TCP STARTS WITH A SPECIAL THREE-WAY HANDSHAKE

TCP has some very basic security built in: the initial connection involves a "three way handshake"

1. Hello B. I am A, trying to connect to you. [SEQ=1234]
2. A, I would love to connect. [SEQ=9876]
3. B, this is A again. Success! [SEQ=1234,9876]

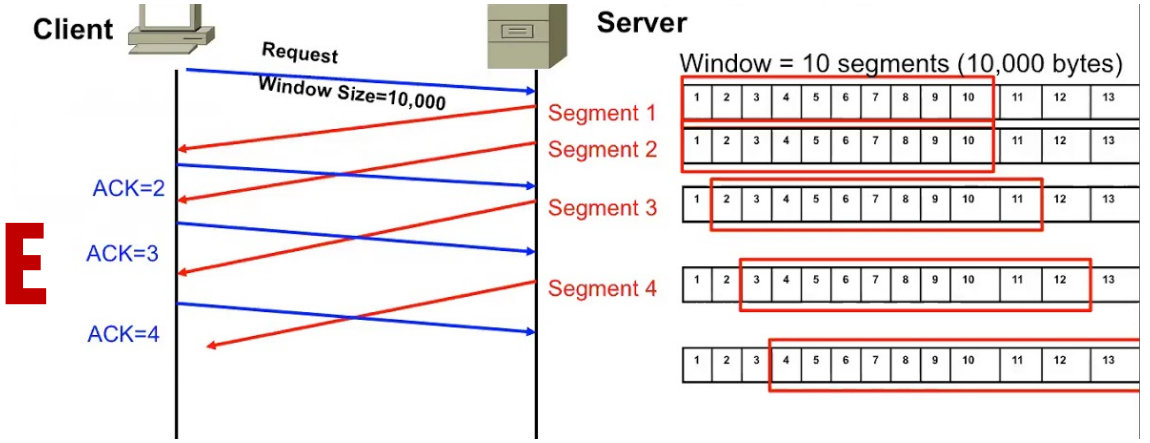
SSL is a standard for creating stronger security on TCP connections. It additionally establishes a session "key" used to encrypt all data.

NAT BOXES TEMPORARILY ALLOW HANDSHAKE RESPONSES THROUGH

Normally, a NAT box only allows packets back for established connections.

But during the three way handshake, responses from the server will reach the sender's TCP "stack" (kernel code).

TCP SLIDING WINDOW: USED AFTER HANDSHAKE



The TCP sliding window. Animation available here

Mile high summary:

When TCP sends data, the sender numbers the packets. The receiver acknowledges receipt ("ack") or if it notices a gap, asks for a retranmission ("nack").

This is done using a form of bounded buffer: the sliding window.

LIMITATIONS

NAT box + firewall

Recall that firewalls often allow outgoing TCP connection requests but block every form of unexpected incoming packet.

… they also will frequently create a whole networking enclave of their own, with IP addresses like

So, without help from the firewall you may be unable to connect from applications to external services!

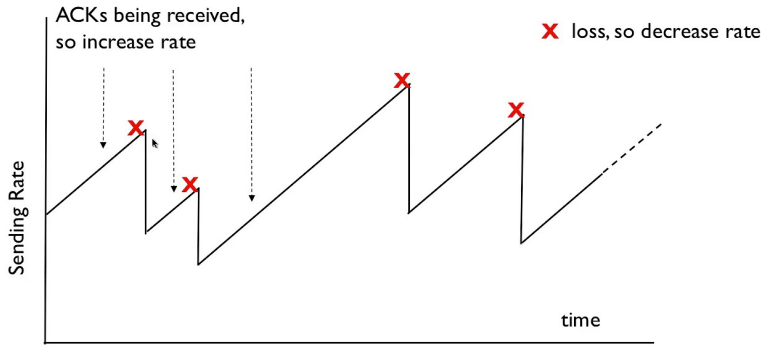
ASYMMETRIC CONNECTIVITY!

A process on my computer, in my home, can connect to sites like Netflix.com or Amazon.com or Azure.com.

- TCP does a three-way handshake
- Once the session is established, messages can flow both ways.

Yet that same Azure.com server wouldn't be able to *initiate* a connection to my computer: traffic would be blocked!

TCP ALSO DOES RATE CONTROL



TCP varies the transmission rate to match the speed of the connection. In effect, the *bounded buffer size* changes *dynamically*.

TCP steadily speeds up until packet drops occur: some router got overloaded. Then it slows down drastically… then speeds up.

Called an *additive increase, multiplicative decrease* scheme.

HOW DO WE MAKE THESE CONNECTIONS?

Consider a client on my machine, perhaps a web browser.

The server is on some other machine, perhaps at Netflix.com

- In fact, Netflix.com is **hosted** by Amazon (rents machines on their cloud)
- So if you talk to a Netflix.com server, your TCP connection will actually be to a machine in an Amazon AWS data center.
- This is an aspect of modern cloud computing: “rent don’t own”.

POSIX SOCKET OPERATIONS: SERVER

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
// logic to initialize serv_addr struct not shown!
bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
listen(sockfd, 5);
int connfd = accept(sockfd);
int bytesread = receive(connfd, buffer, nbytes);
send(connfd, buffer, nbytes);
```

POSIX SOCKET

Create a connection endpoint. This will be used for a TCP connection, but more initialization is required.

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
// logic to initialize serv_addr struct not shown!
bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
listen(sockfd, 5);
int connfd = accept(sockfd);
int bytesread = receive(connfd, buffer, nbytes);
send(connfd, buffer, nbytes);
```

POSIX SOCKET OPERATIONS: SERVER

```
int sockf
// logic to init... serv_addr struct not shown!
bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
listen(sockfd, 5);
int connfd = accept(sockfd);
int bytesread = receive(connfd, buffer, nbytes);
send(connfd, buffer, nbytes);
```

Associate an IP address and port number with it. These come from "gethostbyname"

POSIX SOCKET OPERATIONS: SERVER

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
// logi
bind(so                                    rv_addr));
listen(sockfd, 5);
connfd = accept(sockfd);
int bytesread = receive(connfd, buffer, nbytes);
send(connfd, buffer, nbytes);
```

Tell Linux this server will accept up to five simultaneous client connections via the TCP protocol.

POSIX SOCKET OPERATIONS: SERVER

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
// logic to initialize serv_addr struct not shown!
bind(sockfd, (struct                                    ));
listen(sockfd, 5);
int connfd = accept(sockfd);
int bytesread = receive(connfd, buffer, nbytes);
send(connfd, buffer, nbytes);
```

Accept returns the file descriptor of an established connection, and now the server can read bytes from it

POSIX SOCKET OPERATIONS: SERVER

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
// logic to initialize serv_addr struct not shown!
bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
listen(sockfd, 5);
int connfd = accept(sockf
int bytesread = receive(connfd, buffer, nbytes);
send(connfd, buffer, nbytes);
```

Read a message

POSIX SOCKET OPERATIONS: SERVER

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
// logic to initialize serv_addr struct not shown!
bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));
listen(sockfd, 5);
int connfd = accept(sockfd);
int byt
send(connfd, buffer, nbytes);
```

Send a response message

POSIX SOCKET OPERATIONS: CLIENT

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
… client does not need “bind” or “listen” or “accept”!
int connfd = connect(sockfd, (struct sockaddr *) &serv_addr,
                                                 sizeof(serv_addr));
send(connfd, buffer, nbytes);
int bytesread = receive(connfd, buffer, nbytes);
```

HOW DID THE CLIENT GET THE IP ADDRESSES?

For the remote service, it calls **gethostbyname,** which returns the address struct required for the **connect** operation.

- The server's port number is returned by gethostbyname().
- Some port numbers are standard, like 80 for Web servers.

It gets its own IP address by calling **gethostbyname** on **localhost.**

- The sender's port number can be automatically assigned by Linux.

GETHOSTBYNAME

Gethostbyname is an interface to a massively distributed service called the domain name service (DNS). The ISP hosts a version, and many companies do too.

The job is to map from a string like "cs.cornell.edu" to **132.236.207.53** and the associated port number.

The DNS is tree structured, and can be used dynamically: this allows Netflix to route client requests to different cloud datacenters.

OK… A IS NOW CONNECTED TO B!

A TCP stream is just a stream of bytes.

To make a request with arguments:

- A builds a header identifying the requested operation and serializes the arguments into a byte array.
- B reads the header first, allowing it to learn how much data to read.
- The request-id will be used later to pair the result with a waiting thread.

GOOGLE GRPC: A POPULAR PACKAGING OF THESE MECHANISMS

A library that can be used from C++, Java, Python, etc. Runs over TCP, like REST or CORBA but more modern, faster.

Fairly easy to use, but we won't get into the details in CS4414.

GRPC allows a server to associate an object with a connection. The client can call methods in a type-checked way.

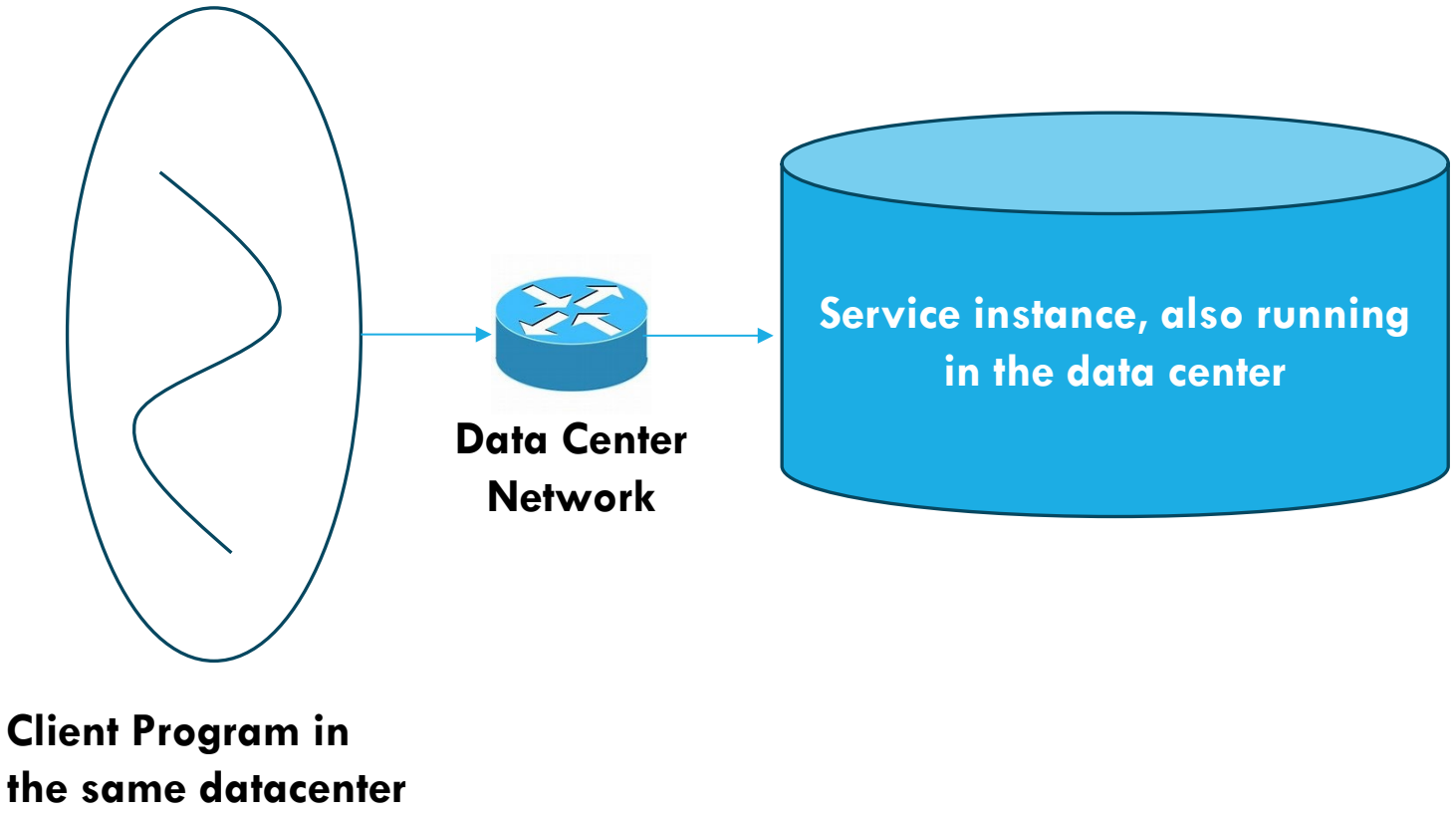
GRPC OVERHEADS?

GRPC has a fast "serialization" method. When using a secure TCP session (with SSL), there is a small extra encryption/decryption delay.

Network delays are the main cost:

- From Ken's lake cottage in Trumansburg to Cornell: 1.5ms (routed via Syracuse).
- Between two machines in a data center, they are as low as 50-100us.
- On the public Internet, bandwidth of 10MB is excellent.
- But inside a data center, rates can reach 10 GB: 1000x faster!.

GRPC EXAMPLE: HELLO WORLD



GRPC EXAMPLE: HELLO WORLD

```
// The greeting service definition is used by both the
// Client and the server.  It contains virtual methods
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
message HelloReply {
  string message = 1;
}
```

Code running in the client

```
class GreeterServiceImpl final : public Greeter::Service {
  Status SayHello(ServerContext* context, const HelloRequest*
                  request, HelloReply* reply) override {
    std::string prefix("Hello to my favorite client!");
    reply->set_message(prefix + request->name());
    return Status::OK;
  }
};
```

Code running in the server

NOTE ABOUT THIS EXAMPLE

What I showed you is incomplete! Normally the client and server would have a *lot* more application-specific code, too

GRPC requires a specific installation (it isn't hard).

Compiling a client or server also requires various include files, and a special "initialization" must be called from main.

WOULD A WEB BROWSER USE GRPC?

The web has its own encoding for messages: XML. GRPC is not used in this setting and would be blocked by firewalls.

This HTML encoding is less efficient than the one GRPC uses. It works well and is universal but is slow to compute and "bulky".

- Every message is encoded as a web page, and every reply
- Data is printed in ascii text format.
- Uses layers of standards: SOAP on HTML on XML… SSL would add a layer of encryption to this.

… SO, WHAT WOULD A WEB BROWSER DO?

You launch your web browser and type in Netflix.com

The first step is to look up the IP address for Netflix.com. This is done by calling gethostbyname, which uses the DNS service.

You get an IP address for Netflix.com (in fact, the IP address of a nearby AWS data center: Netflix is hosted by Amazon).

NEXT STEPS

Your browser initiates an http connection (insecure: https uses SSL, but http isn't encrypted in any way).

AWS selects some machine within the Amazon cloud with the Netflix web server process already running on it.

A three-way handshake is performed.

NEXT STEPS

Your Netflix Cookie!

Your web browser first sends Netflix a “cookie” with your user information. It uses the old web services model, so the cookie is actually encoded as a small web page.

Next Netflix sends back an initial welcome web page. The browser reads this back as a byte stream, deserializes it, and renders the page on your screen.

NEXT STEPS

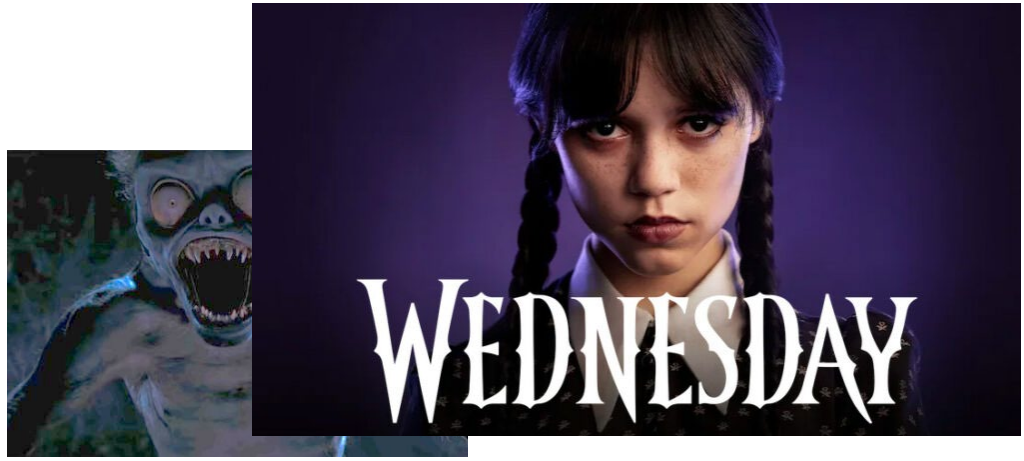
Your Netflix Cookie!

Yuexing Wan, account number #25441. Most recently streamed Cookies of Damnation III, but paused at 01:11:15

Y ...with your user
i ...so the cookie is
actually encoded as a small web page.

Next Netflix sends back an initial welcome web page. The browser reads this back as a byte stream, deserializes it, and renders the page on your screen.

YOU SELECT A MOVIE…



"For the record, I don't believe that I'm better than everyone else. Just that I'm better than you."

Netflix recommends Wednesday.

You click the tile… it is associated with a URL link to the movie.

The browser sends an HTTP request to Netflix to "open" the URL. Netflix streams the movie back.

WHAT IF THIS WAS AN INTERACTION WITH CLAUDE OR CHATGPT OR COPILOT?

Everything is the same!

The modern trend is to view the cloud as a collection of *agents* that might be AIs, and might be cloud services like we just saw

Agentic AI is all about having a browser or a local AI that sends requests to a cloud where other services or AIs run. They could "do things", then send back data that the initiator requested.

VPN AND VPC

Many of us have access to special computers at Cornell.

These aren't available for public use, so they live inside cs.cornell.edu and aren't even visible from outside Cornell.

With a "virtual private network" you can step inside the Cornell firewall from home, as if you were in Gates Hall!

HOW A VPN WORKS.

Cornell operates a special VPN router. You need yournetid to log into it. They use a product called Cisco AnyConnect.

Once you log in, a TTP SSL connection will exist from your machine to the VPN router.

IP addresses in the Cornell domain, or DNS requests, are automatically sent over this encrypted link.

VIRTUALLY PRIVATE *CLOUD*

VPC is similar in concept, for people who use cloud computing.

Suppose your organization uses 10 machines on Azure or AWS.

Those 10 can be isolated from other cloud users: a VPC creates the illusion of a network composed purely of your machines, plus services the cloud operator provides (like the global file system)

WITH VPC, ALL TRAFFIC IS ENCRYPTED

This way even if some intruder is watching the network, and even if your own applications use GRPC without SSL enabled, your messages are still encrypted.

It happens automatically. AWS and Azure both have special hardware accelerators to do the encryption and decryption.

In fact, with VPC they even encrypt files, using the same technique!

DISTRIBUTED COMPUTING SCENARIOS

Distributed systems typical center on networked protocols. You use the system either as a library (such as Derecho), or by connecting to a cloud hosted service that internally was built using such a library (such as when you use Kafka).

At the lowest level, these libraries often use GRPC or a fancier form of message passing, such as RDMA (which offloads data transfer to network hardware for blazing speeds)

SUMMARY OF LECTURE 16

When applications will run on multiple machines, we use networking to link the processes.

This occurs over TCP, often with SSL encryption. Encryption is not the default but is very common.

Google GRPC is one common library for building networked applications with a C++ object-oriented style of code.

AND NOW… A SELF-TEST ROM-COM MEET CUTE QUESTION

Every course should allude to a 1989 rom com at least once, right?

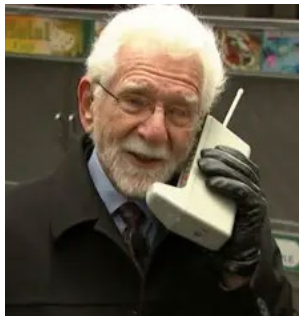
Harry and Sally are both CS geeks, really love logic, and work for the same faceless corporate logic company in New York in a building with almost no windows and no external wifi connections.

This was the 1989. Just lean into it!

HARRY AND SALLY'S LUNCH DATE

Harry and Sally usually meet for lunch in the basement cafeteria but just for a change they decide to meet outside *if the weather is good*.

Sally has a window in her office, but Harry is in a windowless cubicle. So they coordinate by email. The only thing is that both are sometimes pulled into meetings and so might not see email. In 1989 mobile phones and text messages were uncommon.



Can they coordinate to meet outside?

HARRY AND SALLY'S ILL-FATED LUNCH DATE

Sally emails: "Harry, it looks spectacular outside! Lunch outside?"

Harry emails back: "Love to!"

Sally: "Great!"

It is safe for Sally to assume Harry got this last email? What if he didn't?

HARRY AND SALLY'S ILL-FATED LUNCH DATE

Sally emails: "Harry, it looks spectacular outside! Lunch outside?"

Harry emails back: "Love to!"

~~Sally: "Great!"~~

Without Sally's last email, what would Harry do? He would wonder if she got his acknowledgement!

HARRY AND SALLY'S ILL-FATED LUNCH DATE

Sally emails: "Harry, it looks spectacular outside! Lunch outside?"

~~Harry emails back: "Love to!"~~

~~Sally: "Great!"~~

Without Harry's acknowledgment, Sally wouldn't know if Harry realizes the weather is good. And she worries he will go to the cafeteria, like usual!

HARRY AND SALLY'S ILL-FATED LUNCH DATE

~~Sally emails: "Harry, it looks spectacular outside! Lunch outside?"~~

~~Harry emails back: "Love to!"~~

~~Sally: "Great!"~~

… they will meet in the cafeteria just like usual! Being logicians they will agree to exchange more messages next time!

HARRY AND SALLY'S ILL-FATED LUNCH DATE

Sally emails: "Harry, it looks spectacular outside! Lunch outside?"

Harry emails back: "Love to!"

Sally: "Great!"

Harry: "Ok, we're on"

Sally: "Yup, looks that way"

Harry: "Whew! But something is worrying me…"

And they meet in the cafeteria again.

FORTUNATELY, LOVE ISN'T LOGICAL!

Whew!

SELF-TEST QUESTION

But programs *are* logical. They can't just do things for love.

Would the Sally and Harry issue arise for GRPC over TCP?

Now think of a typical task such as asking an AI some question. Does the problem arise here, or is it somehow avoiding an issue that caused Sally and Harry to "get trapped" in a cycle?

SELF-TEST QUESTIONS

What does the Sally and Harry scenario tell us about failure detection in cloud data centers?

Recall that Derecho and similar protocols tolerate crash failures. Classic Paxos doesn't even pause… yet ultimately needs to clean up temporary state.

Is failure detection even possible? How does a library like Derecho "work around" the issue?

(READINGS FOR LOGICIANS)

Knowledge and common knowledge in a distributed environment. Joseph Y. Halpern and Yoram Moses. J. ACM 37, 3 (July 1990), 549–587.

Monotonicity and Opportunistically- Batched Actions in Derecho. Ken Birman, Sagar Jha, Mae Milano, Lorenzo Rosa, Weijia Song, Edward Tremel. 25th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems, Oct 2023.

SELF-TEST QUESTIONS

We saw that networks are fundamentally not perfect and might even throw packets away to shed load

This means that retransmission is common: a resend of the same request, with the same unique request id and caller id.

What issues would you need to think about if you were building an AI service and it sometimes would see retransmitted requests?

SELF-TEST QUESTIONS

In earlier lectures we learned that delays can leave hardware idle and pile up, causing LLMs and LRMs to underperform.

What are all the potential sources of delay you can identify in a scenario where a company is offering an LLM-based agentic service to its customers? Think about delay both from the customer to the LLM application, and also about delay internal to the LLM.

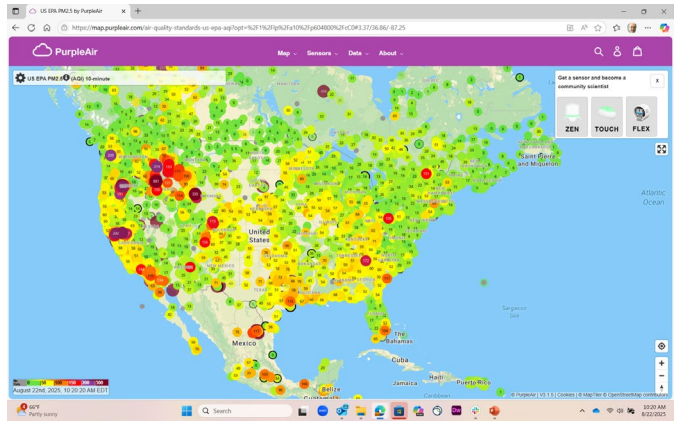
SELF-TEST QUESTIONS

Those clients (on the prior slide) are talking to the company over the internet using something like GRPC.

But the agentic service is helping many clients at the same time, hence seeing thousands of concurrent requests.

We learned that asynchronous streaming of requests and batching can pay off. How would you implement logic to map from individual requests to streaming or batching and back? Where would it run?

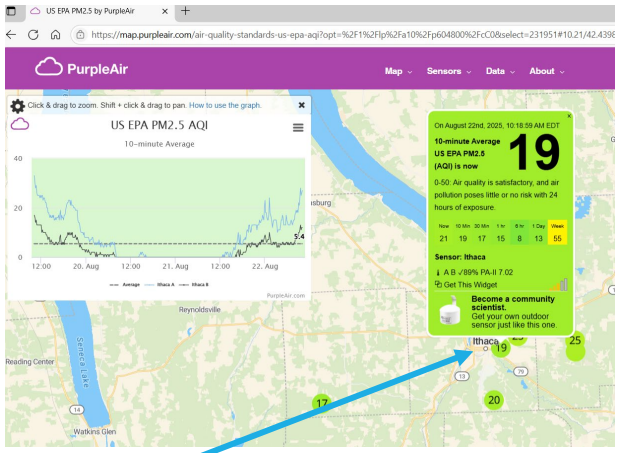
SELF-TEST QUESTIONS



purpleair.com offers a zoomable map showing air quality in the US. Try using it and zoom in and out with your mouse. Now pan and Zoom to Seattle.

You'll see that the interface is sluggish when we pan or rescale. Google maps rescales much more quickly, even if you are in the mode of searching for restaurants or something.

MORE ON PURPLEAIR.COM



Their architecture probably uses a sharded key-value data store on AWS. Ken has a sensor on his garage: Ithaca-A and Ithaca-B. These names are "keys"; the sensor uploads data every few minutes over the internet.

Brainstorm possible ways to improve zoom in / zoom out for better speed. Feel free to propose additional data preprocessing that purpleair.com could implement on the AWS cloud to speed itself up.

What might be an interesting agentic AI service purpleair could offer?