CS5412/LECTURE 12 THE GEOSCALE CLOUD

Ken Birman
CS5412 Spring 2020

BEYOND THE DATACENTER

Although we saw a picture of Facebook's global blob service, we have talked entirely about technologies used inside a single datacenter.

How do cloud developers approach global-scale application design?

Today we will discuss "georeplication" and look at some solutions.

WHERE TO START? AVAILABILITY ZONES

Companies like Amazon and Microsoft faced a problem early in the cloud build-out: servicing a data center can require turning it off!

Why?

- Some hardware components are too critical to service while active, like the datacenter power and cooling systems, or the “spine” of routers.
- Some software components can’t easily be upgraded while running, like the datacenter management system.
- In fact there is a very long list of cases like these

AVAILABILITY ZONES

So… they decided that instead of building one massive datacenter, they would put two or even three side by side.

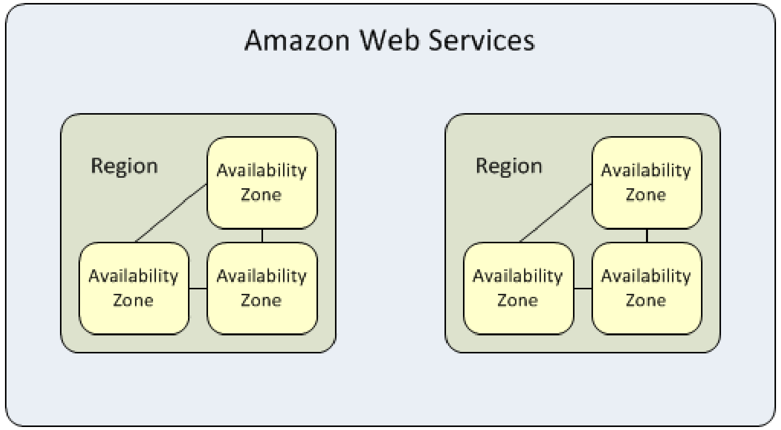
When all are active, they spread load over them, so everything stays busy.

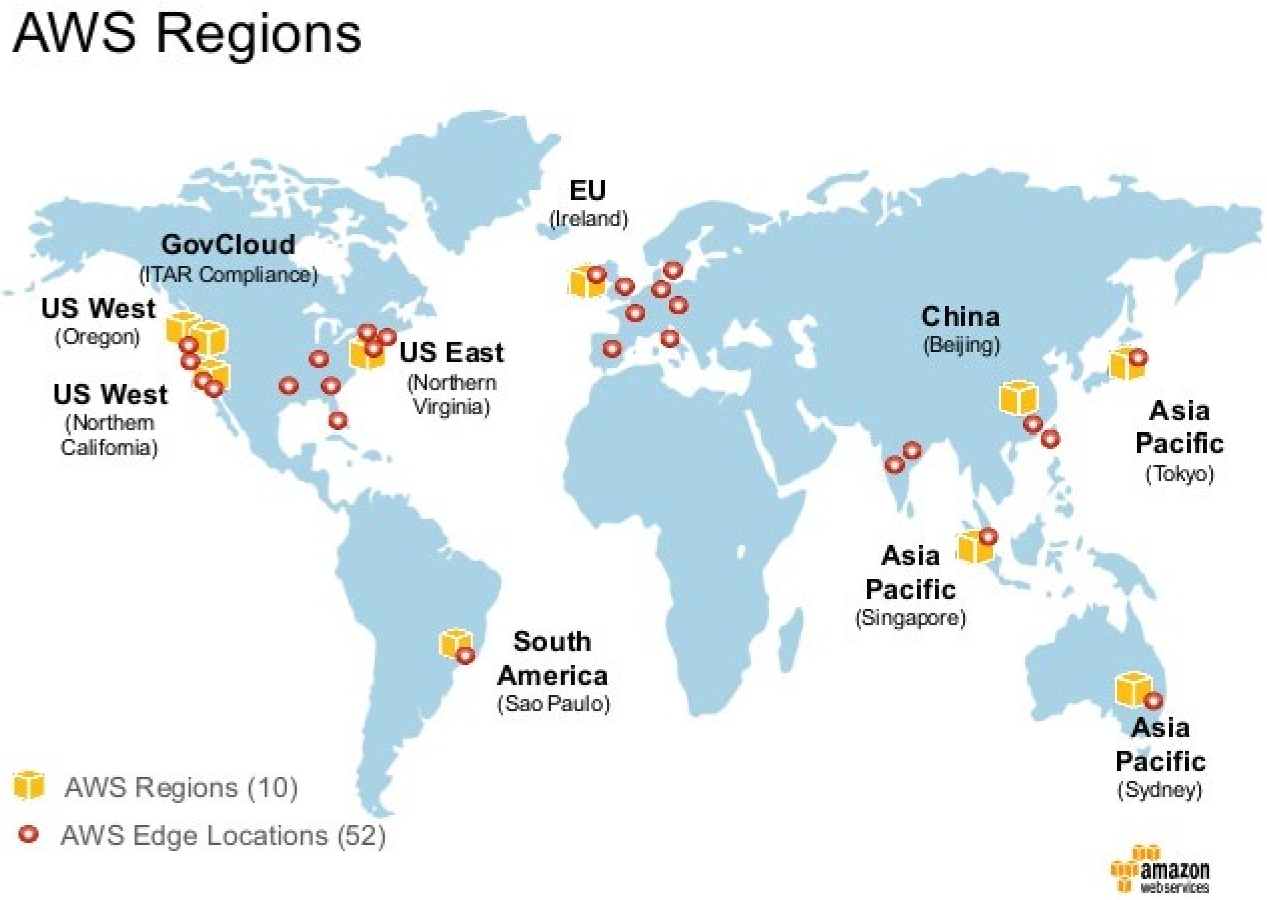
But this also gives them an option for shutting one down entirely to do upgrades (and with three, they would still be able to "tolerate" a major equipment failure in one of the two others).

AVAILABILITY ZONES – AMAZON AWS

AWS Edge is less capable

The AWS “region” has an availability zone structure

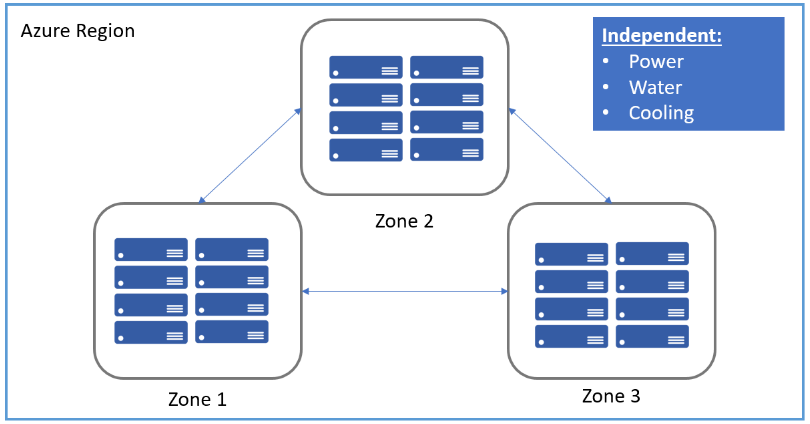




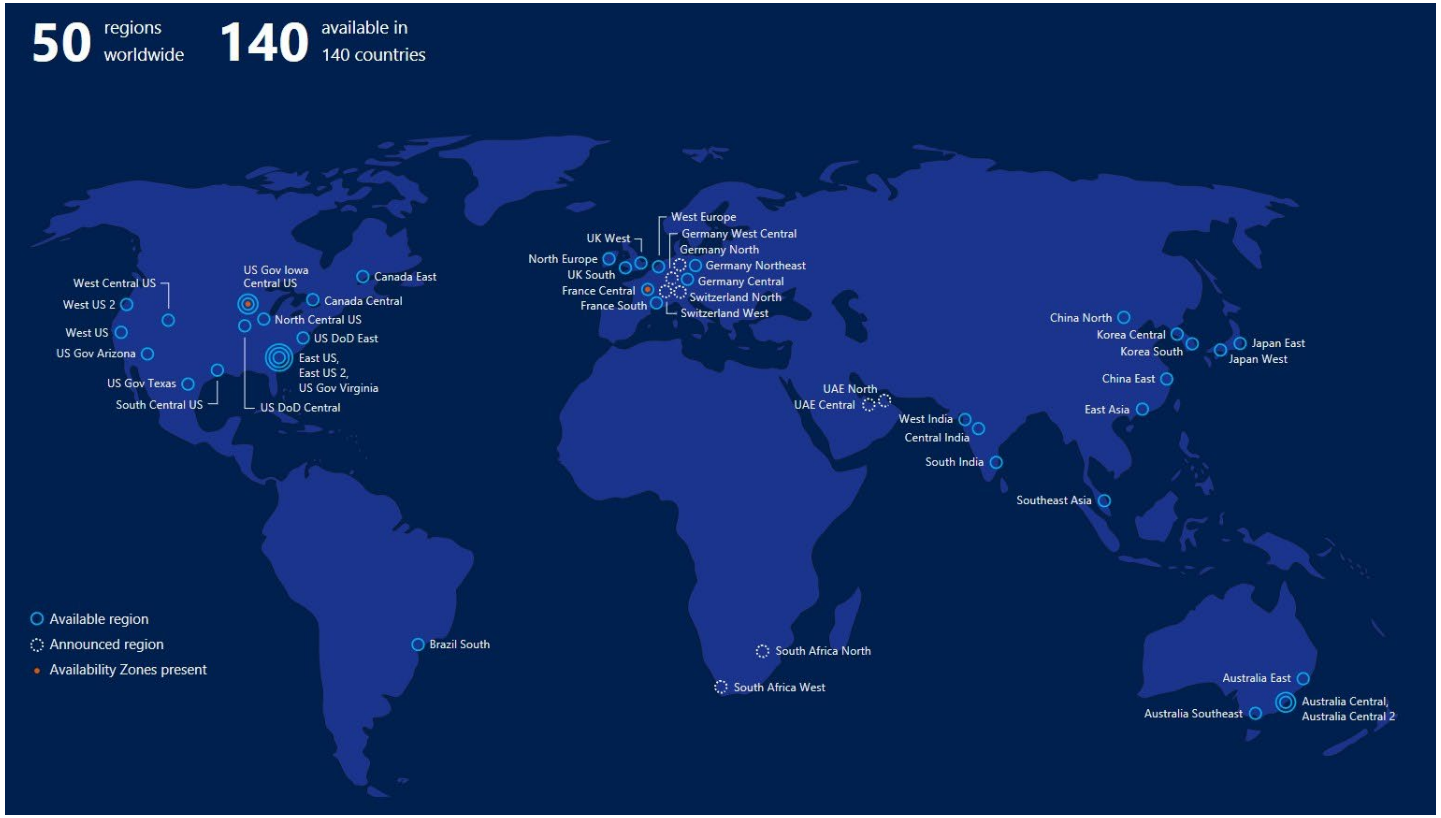
AVAILABILITY ZONES – MICROSOFT AZURE

Notice the blue circles.

Those are regions
with three zones.



Azure Availability Zones are separate buildings 10's of miles apart within a Region



MODELS FOR HIGH-AVAILABILITY REPLICATION

Within a datacenter you can just make TCP connections and build your own chain replication solution, or download Derecho and configure it.

But communication between datacenters is tricky for several reasons. Those same approaches might not work well, or perhaps not at all.

CONNECTIVITY LIMITATIONS

Every datacenter has a firewall forming a very secure perimeter: applications cannot pass data through it without following the proper rules.

Zone-redundant services are provided by AWS and Azure and others to help you mirror data across zones, or even communicate from your service in Zone A to a “sibling” in Zone B.

Direct connections via TCP would probably be blocked: it is easy to connect into a datacenter but hard to connect *out* from inside!

WHY DO THEY RESTRICT OUTGOING TCP?

The modern datacenter network can have millions of IP addresses inside each single datacenter.

But these won't actually be unique IP addresses if you compare across different data centers: the addresses *only make sense within a data center.* In fact, many IP addresses only make sense *within your own "private cloud"!*

Thus a computer in datacenter A often would not have an IP address visible to a computer in datacenter B, blocking connectivity!

HOW DOES A BROWSER OVERCOME THIS?

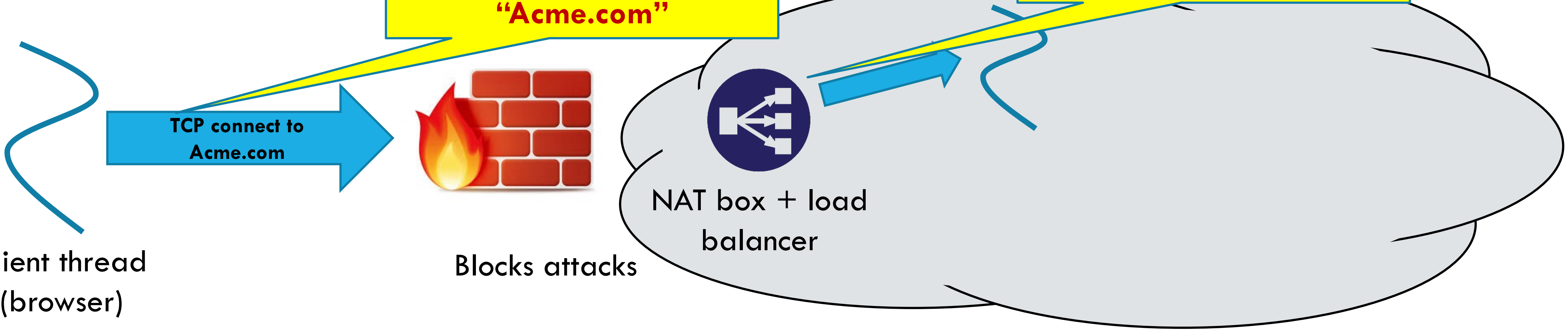
Here at Cornell, your browser is on the *public* internet, not internal to a datacenter.

So it sends a TCP connect request to the datacenter over one of a small set of datacenter IP addresses covering the full datacenter.

- AWS, which hosts for many other sites, has a few IP addresses per site.
- Some systems mimic this approach, others have their own ways of ensuring that traffic to Acme.com gets to Acme's servers, even if hosted on their framework.

ARRIVING TRAFFIC: NAT BOX

On arrival, packets are scanned for possible BOT traffic or DDoS attacks, then the IP addresses are translated to “load balance” work over the proper servers. This would be costly, but high-spee re is used to do the need es.


IP address looks like “myIP”, connecting to “Acme.com”
IP address looks like “NatBox”, connecting to “server17.Acme.com”
TCP connect to Acme.com
Client thread (browser)
Blocks attacks
NAT box + load balancer

HOW DOES IT PULL THIS TRICK OFF?

The NAT box maintains a table of "internal IP addresses (and port numbers) and the matching "external" ones.

As messages arrive, if they are TCP traffic, it does a table lookup and substitution, then adjusts the packet header to correct the checksum.

Thus "server17.Acme.com" cannot be accessed directly and yet your messages are routed to it, and vice versa.

BUT WITH GEOREPLICATION, THIS BLOCKS YOUR CODE FROM CONNECTING TO ITSELF

If you have machine A.Acme.com inside AWS or Azure, and then try to connect to B.Acme.com, it works inside a single datacenter.

- In fact you will be running in an "enterprice VLAN" or "VPC": what seems to be a private cloud.
- If you were to launch Ethereal or a similar sniffer you only see traffic from your own machines, not traffic from other datacenter tenants.

But if B was in a different datacenter, the connection simply won't work.

- Both A and B are behind NAT boxes, so neither can see the other!

COULD THIS BE SOLVED?

Actually, *yes.*

Because two NAT boxes are employed here, Amazon or Azure actually could allow connectivity using some form of load-balancing scheme.

But they don’t do so because they don’t want uncontrolled connections.

OPTIONS?

Some vendors (not AWS or Azure) do offer ways to make an A-B connection across datacenters in the same region (availability zone).

You need to use a special library they provide and otherwise, the connection would fail. And they charge for this service.

With AWS and Azure, you use an existing “Zone-aware” service

ZONE AWARE SERVICES ON AZURE

Linux Virtual Machines

Windows Virtual Machines

Virtual Machine Scale Sets

Managed Disks

Load Balancer

Public IP address

Zone-redundant storage

SQL Database

Event Hubs

Service Bus

VPN Gateway

ExpressRoute

Application Gateway

A FEW WORTH NOTING

Zone-aware storage is a storage mirroring option.

Files written in zone A would be available as read-only replicas in zone B. B sees them as a read-only file volume under path /Zone/A/…

This is a very common way to share information between data centers.

A FEW WORTH NOTING

The **Azure Service Bus** in its “Premium” configuration

- This is a message bus used by services within your VPC to communicate.
- Azure offers a configuration that automatically transfers data across zones under your control.
- Again, you need to follow their instructions to set it up. They charge but the performance and rate have historically favored this model.
- Basically, two queues hold messages, and then they use a set of side by side TCP connections to shuttle data in both directions. Very efficient!

A FEW WORTH NOTING

Azure’s **zone-redundant virtual network gateway**

- Used when you really do want a connection of your own, via TCP
- Setup is fairly sophisticated but they walk you through it.
- In effect, creates a special “pseudo-public” IP address for your services, which can then connect to each other. Not visible to other external users
- But performance might be balky: this isn’t their most performant option. And they charge by the megabyte transferred over the links.

QUEUING VERSUS MESSAGE BUS MODEL

We mentioned these earlier.

A message queue is a store-and-forward scheme, like email. Useful when doing batched processing ("give me all the emails from the past hour")

A message bus is used when you want minimal latency on a per-event basis. Data is sent without any persistent storage, like a text message.

HEAVY TAILED LATENCY

A big concern with georeplication is erratic delays.

Within an availability zone, the issue is minimal: the networks are short (maybe blocks, maybe a few miles), so latencies are tiny.

But with global WAN links, latencies can be huge and variation grows.

- Mean delay from New York to London: 90ms
- Mean delay from New York to Tokyo: 103ms

HEAVY TAILED LATENCY

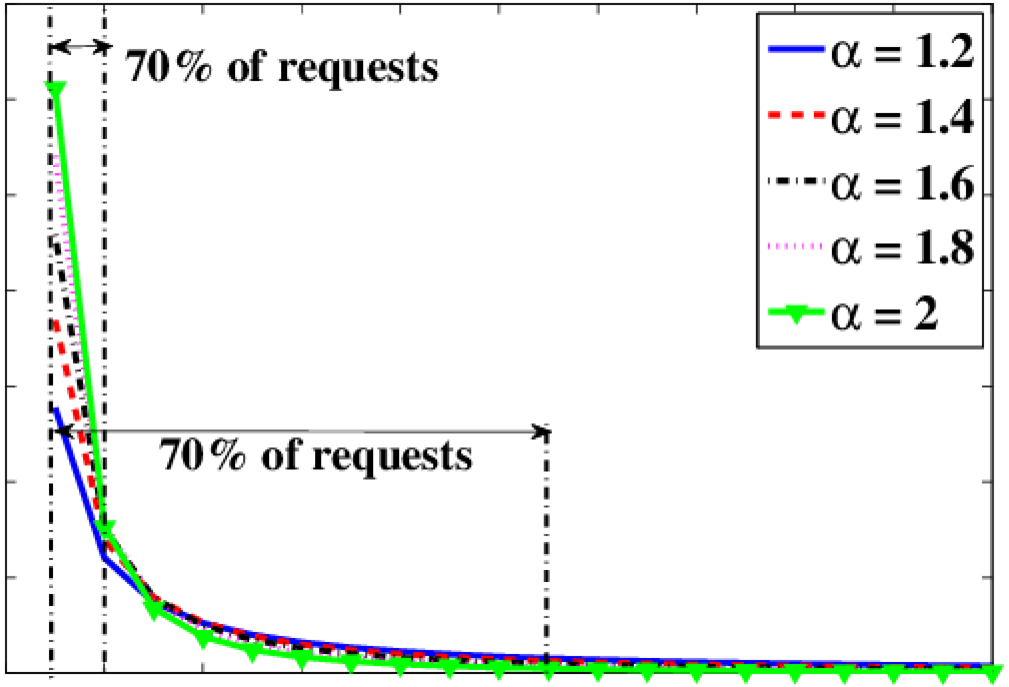
The number of events in the i'th popularity bin is proportional to $1/i^{\propto}$

Zipf's law

Studies reveal "Zipf" latency distributions

Most traffic gets through very quickly

But some traffic takes extremely long. We say that these distributions are heavy-tailed if the "very slow" traffic adds up to a substantial portion of the total traffic.



ISSUE THIS RAISES

Should we want to wait for *all* sites to respond, or just a quorum?

Quorum: a subset of the sites large enough to include a majority. Any two quorums are certain to overlap. Call this Q: we require Q + Q > N

- If task A updates a quorum and task B later reads the quorum, then B is certain to overlap with A on some server. So A "sees" B's update.
- One can get fancier by introducing separate quorums for writes and for reads, such that $Q_w + Q_w > N$, and $Q_w + Q_r > N$

ISSUE THIS RAISES

Seemingly, a quorum is far better.

On the other hand, perhaps we would want all sites within an availability zone, but then wouldn't need to wait for geo-replicas to respond?

Leads to a three-level concept of Paxos stability.

- Locally stable in datacenter… Availability-zone stable… WAN stable

WHAT HAVE EXPERIMENTS SHOWN?

Basically, Paxos performs poorly with high, erratic latencies.

A big issue is that delay isn't symmetric:

- The path from Zone A to Zone B might be slow
- Yet the path from Zone B to Zone A could be fast at that same instant.

So the outgoing proposals experience one set of delays, and replies from Paxos members experience different delays. You end up waiting "for everyone"

GOOGLE SPANNER: BACKGROUND ON CHUBBY

Google was one of the first to use a service built from Paxos in real datacenter settings. It was called Chubby, and was created by Mike Burrows with a Cornell PhD graduate, Tushar Chandra.

In a single data center, Chubby worked well, although it was dramatically slower than Derecho.

But in a WAN setting, Google struggled to try and build a global version of Chubby. Basically, they couldn't pull it off!

GOOGLE SPANNER INNOVATION: TRUETIME

Google uses actual time as a way to build an asynchronous totally ordered data replication solution called Spanner.

Google **TrueTime** is a global time service that uses atomic clocks together with GPS-based clocks to issue timestamps with the following guarantee:

- For any two timestamps, T and T’
- If T’ was fully generated before generation of T started,
- Then T > T’

WHY SUCH A TORTURED DEFINITION?

With the guarantee they offer, if some operation B was timestamped T, and could possibly have seen the effects of operation A with timestamp T', then we can order B and A so that A occurs before B.

The full API is very elegant and simple: TT.now(), TT.before(), TT.after(). TT.now() returns a range from TT.before() to TT.after().

The basic guarantee is that the true time is definitely in the range TT.now().

TT.before() is definitely a past time, and TT.after() is definitely in the future.

IMPLEMENTING TRUETIME

Google starts with a basic observation:

- Suppose clocks are synchronized to precision δ
- It is 10:00.000 on my clock, and someone wants to run transaction X.
- What is the largest possible timestamp any zone could have issued?

My clock could be slow, and some other clock could be fast.

So the largest (worst case) possible will be T+2 δ

MINIMIZING δ IN TRUETIME

Google uses a mix of atomic clocks and GPS synchronization for this. They synchronize against the mix every 30s, then might drift in between.

GPS can be corrected for various factors, including Einstein's relativistic time dilation, and they take those steps.

In the end their value of δ is quite small (0-6ms). So at actual time 10am, transaction X might get a TT.after() timestamp like **(10:00.006, zone-id, uid).**

- The zone id and uid are to break ties.

SPANNER'S USE OF TRUE TIME

Spanner is a transactional database system (in fact using a key-value structure, but that won't matter today).

Any application worldwide can generate a timestamped Spanner transaction.

These are relayed over the Google version message services. Their service ***delivers messages from Zone X to Zone Y in timestamp order.***

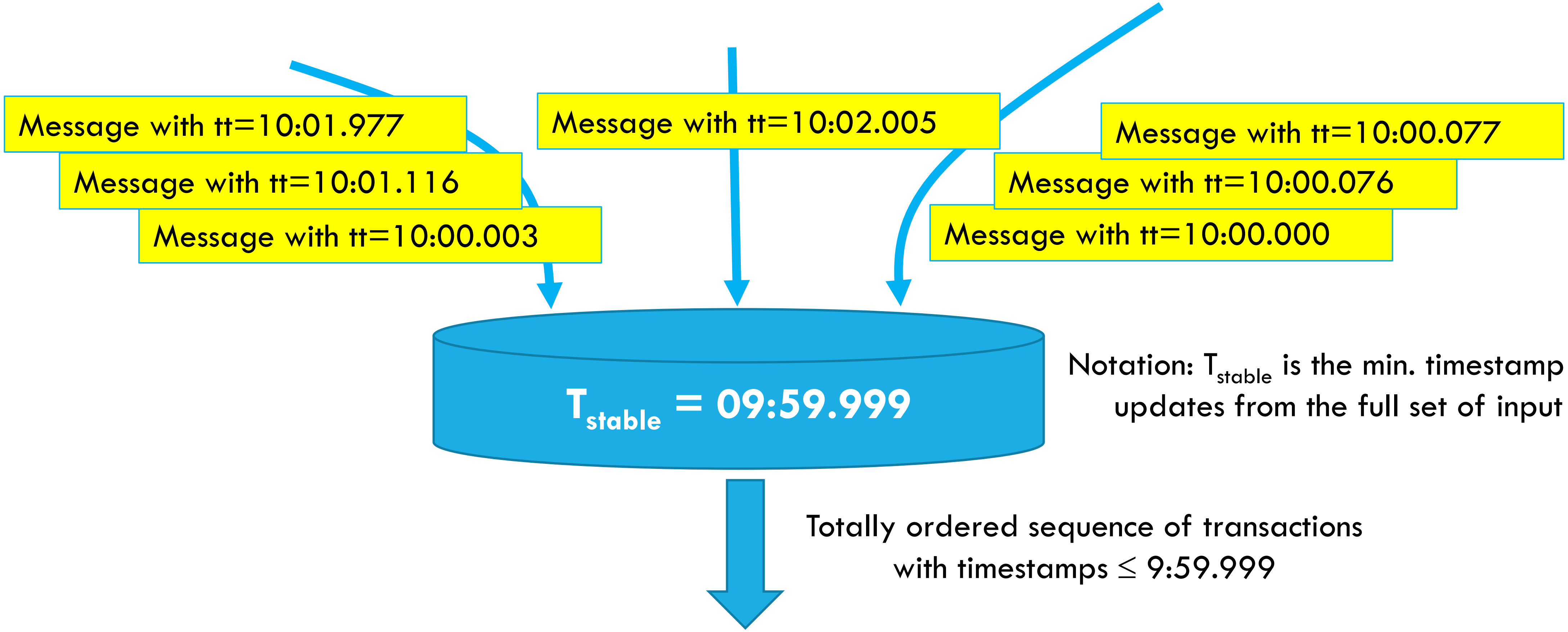
LIFE OF A SPANNER INSTANCE

Spanner has connections to perhaps hundreds of remote zones.

Transactions flow in on these connections, and it stores them until every zone has sent some transaction with a timestamp of value T_{stable} or larger.

Then it can apply all transactions with timestamps $\leq T_{stable}$, in order.

LIFE OF A SPANNER INSTANCE



Message with tt=10:01.977

Message with tt=10:01.116

Message with tt=10:00.003

Message with tt=10:02.005

Message with tt=10:00.077

Message with tt=10:00.076

Message with tt=10:00.000

T_{stable} = 09:59.999

Notation: T_{stable} is the min. timestamp for incoming updates from the full set of input channels.

Totally ordered sequence of transactions with timestamps ≤ 9:59.999

A NICE FEATURE OF SPANNER

Think back to how CASD took worst-case assumptions and then created a **slow** protocol: It always delays until the worst-case delay has passed.

But Spanner can safely assume that the links to all its data centers are working, and it just waits to hear from all of them. If a data center is taken offline, Spanner is told, so then it won't wait for that link.

Thus Spanner can make ordering decisions "as soon as possible".

WHY DOES THIS WORK?

If Spanner has received messages with timestamps > T_{stable} from every zone, *it has seen every transaction with* timestamp ≤ T_{stable}!

- This is because the connections deliver messages in order, by timestamp.
- If an earlier transaction were to show up, it would violate this rule.

So, if it now places those transactions into timestamp order (breaking ties by (zone-id, uid)), they can be applied to the global database in total order.

WHAT LIMITS SPANNER?

One limiting factor is that although the zone-to-zone data transfers run over large numbers of parallel TCP connections, the messages need to be put into order to obtain this “monotonicity” property.

A second limiting factor is Zipf-like latency with heavy tails: Spanner will often have to wait for “laggards”.

At global scale the effect can be significant (many seconds).

PAXOS ALL OVER AGAIN?

With the classic implementation of Paxos, we have a back-and-forth interaction *that traverses every link several times in both directions.* *<u>Paxos experiences delays repeatedly.</u>*

With Google Spanner, there are global asynchronous flows but no back-and-forth coordination of this kind. *<u>So Spanner experiences delays once.</u>*

Intriguing observation: Derecho's version of Paxos "behaves" like Spanner.

WHAT LIMITS SPANNER?

Google researchers report that the most frustrating issue is that a transaction cannot even be processed *locally* without waiting this way!

In some situations, a speculative result may be acceptable: "If there are no conflicts, my transaction would run and you would win the auction!"

But in other settings, consistency is absolutely needed, so there is no choice.

TRICKS TO MINIMIZE IMPACT

If a zone hasn't been sending any transactions, it should "pause".

- Send a special "End of sequence" message
- Cease to send new transactions
- Other zones no longer need to wait.

Later, to resume, it will have to get permission to restart:

- Send a "resume request"
- Every other zone must acknowledge this before new transactions can start.

OTHER OPTIONS?

With a “primary zone” model, we can shard our database and assign each shard a primary owner (only the owner zone can update that shard).

Then you can make the rule that the primary owner can always do transactions on shards that it owns, without waiting.

But if *any* transaction would need to access shards for which it isn’t primary, than *all* must use the Spanner ordering mechanism (you can’t mix methods).

WHAT LIMITS THE PRIMARY ZONE METHOD?

In some situations it is hard to know what shards a transaction would access before the transaction code actually executes.

For example, the keys a transaction will touch might be a function of the data it reads in some initial step. So until it executes, we don't know the key set, and can't know if those will all be local shards.

Spanner is really aimed at cases like these.

MORE REMARKS ABOUT IP ADDRESSES

We have seen that with modern cloud systems, you could connect to Acme.com and be routed to Amazon.com instead.

In fact the modern cloud has considerable control over this, and you can influence that layer.

For each external client, when it initially connects, you can program the cloud DNS to route that specific request to a particular data center, and control whether or not the DNS record will be cached.

MORE CONTROL

You can use information in a cookie left on the client to advise the cloud on which of your servers would be the best one to handle a connection.

And you can select between a wide range of elasticity and routing and load-balancing “recipes” that are offered by the vendor.

In the next generation of routers (SDN routers) you’ll be able to even provide packet inspection rules that could route based on values in specific fields of the incoming request.

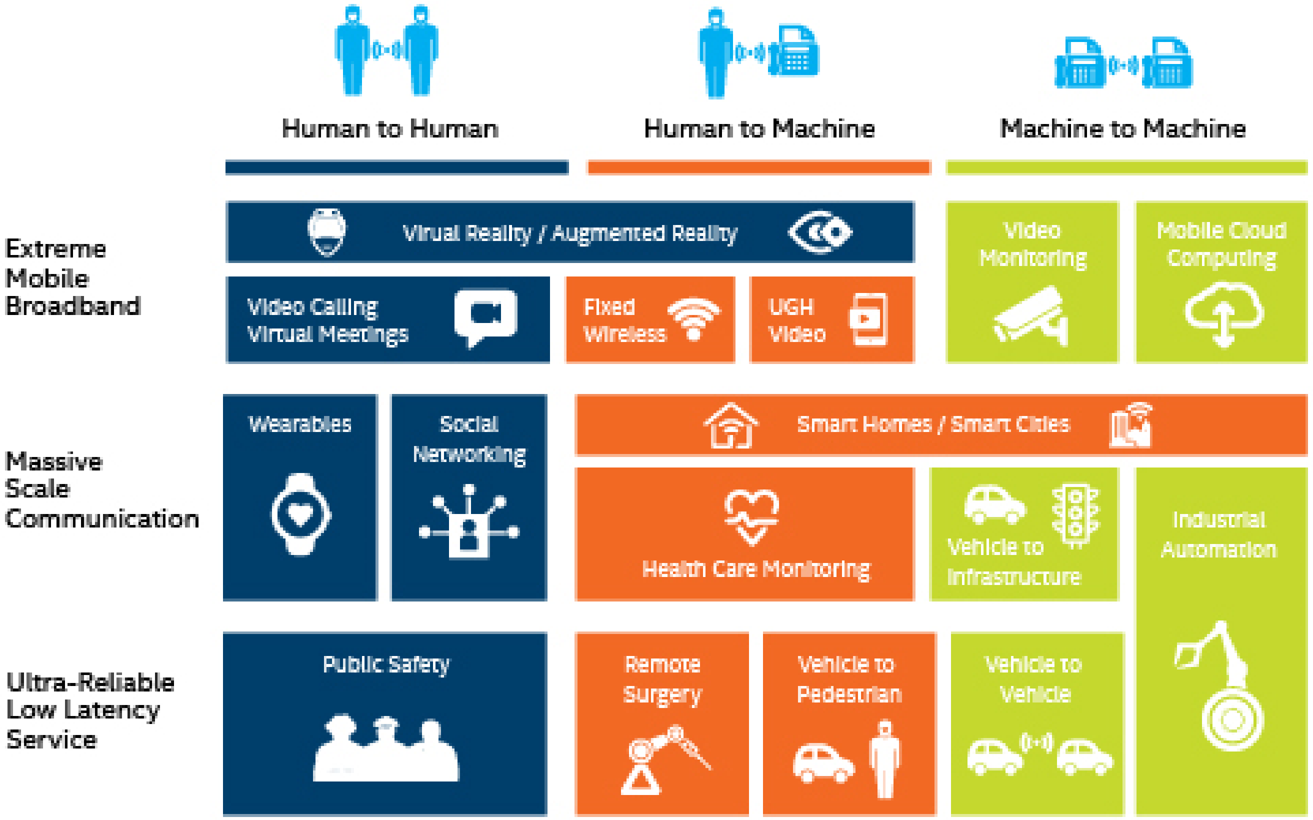
MOBILITY AND GEOREPLICATION

With the advent of 5G, more and more IoT systems will include mobile clients in which the sensors are actually inside a vehicle like a car or plane.

5G hubs are very much like Azure IoT Edge, and in fact Azure IoT Edge may be a leading platform option for 5G services.

With mobile users, the user itself may have a dynamically changing IP address, and might be using multipath TCP to maintain connectivity.

THE PATH TO 5G (SLIDES FROM AN INDUSTRY "VISIONING" EVENT)



Human to Human

Human to Machine

Machine to Machine

Extreme Mobile Broadband

Virual Reality / Augmented Reality

Video Calling Virtual Meetings

Fixed Wireless

UGH Video

Video Monitoring

Mobile Cloud Computing

Massive Scale Communication

Wearables

Social Networking

Smart Homes / Smart Cities

Health Care Monitoring

Vehicle to Infrastructure

Industrial Automation

Ultra-Reliable Low Latency Service

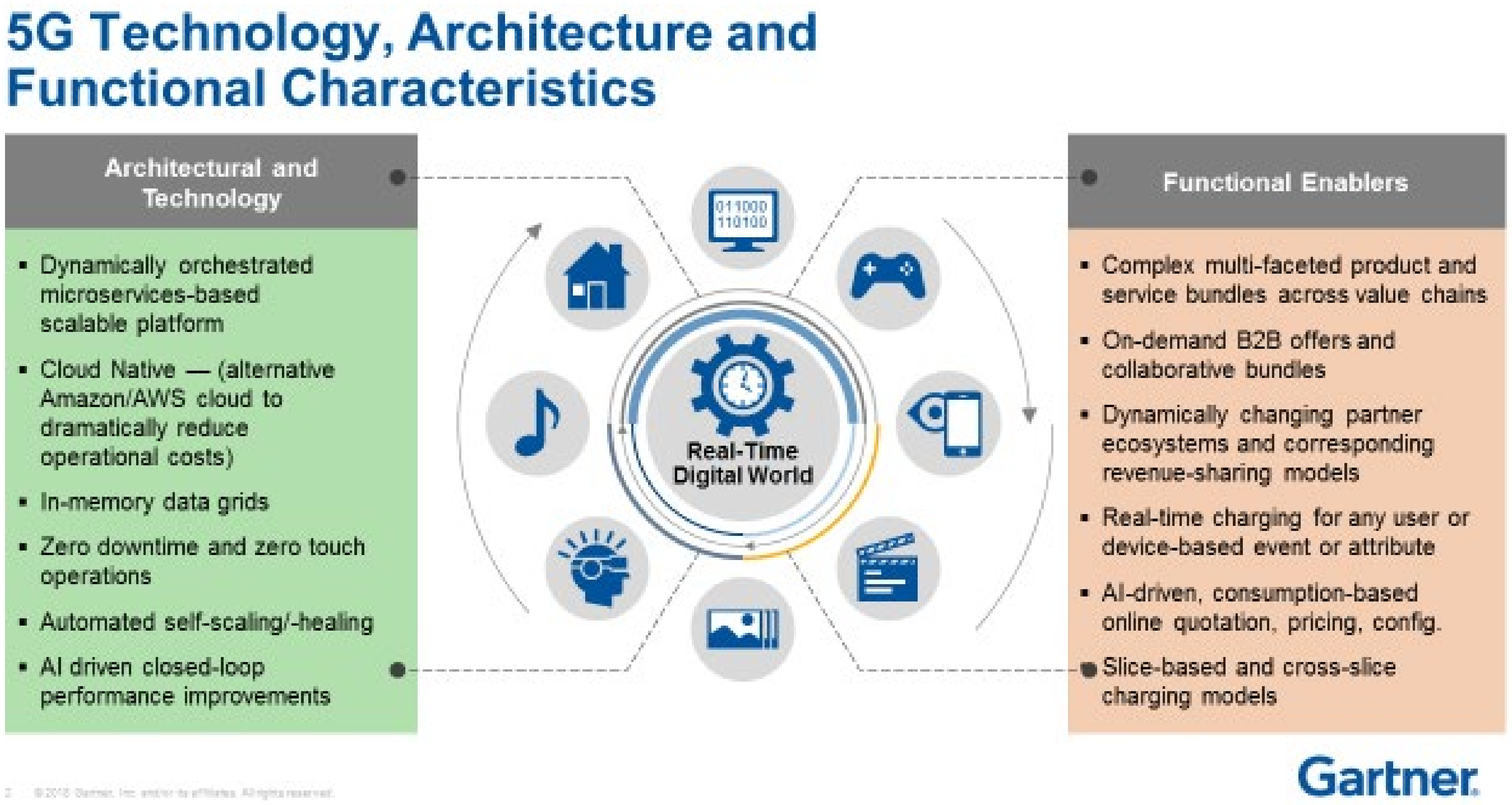
Public Safety

Remote Surgery

Vehicle to Pedestrian

Vehicle to Vehicle

THE PATH TO 5G



5G Technology, Architecture and Functional Characteristics

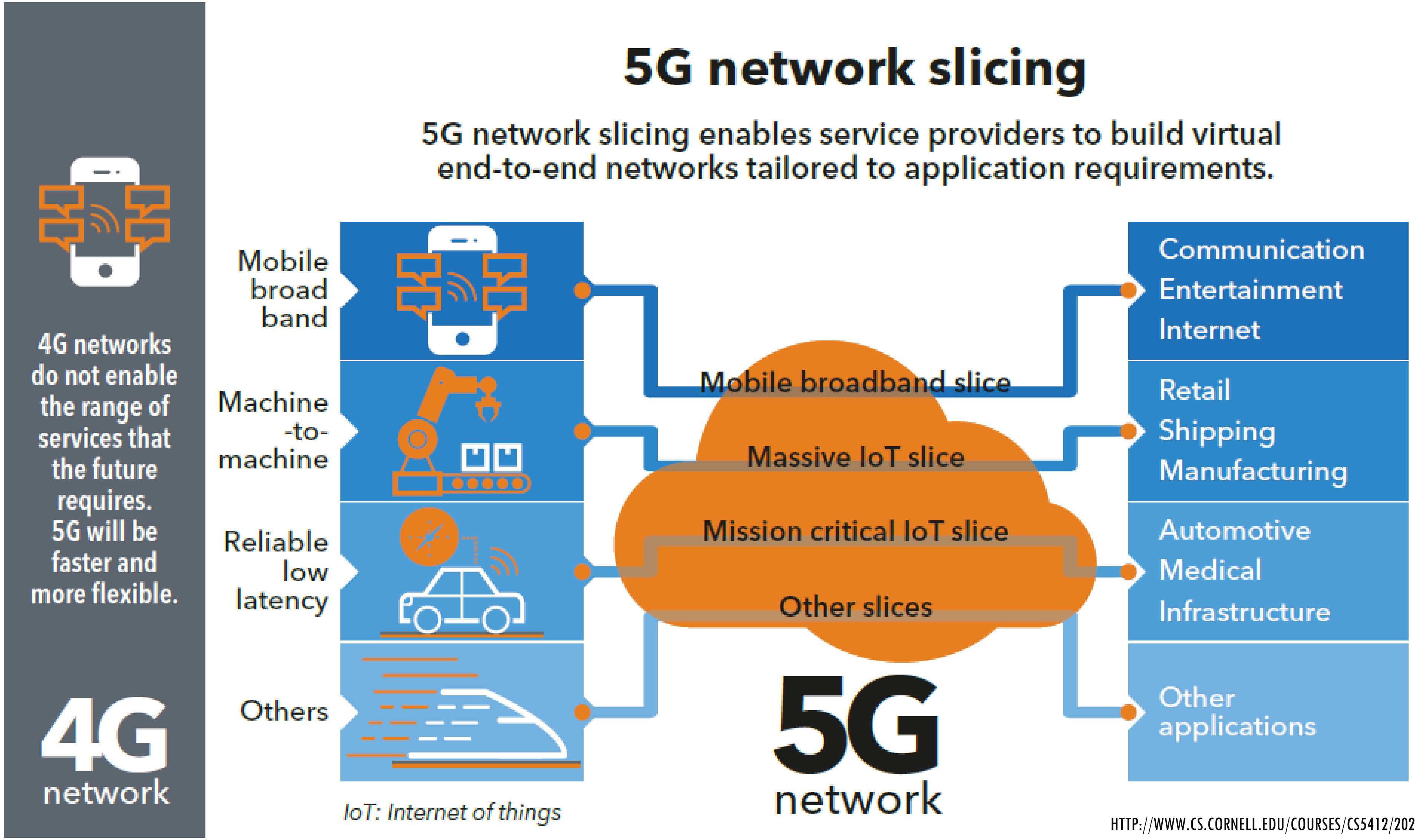
Architectural and Technology

- Dynamically orchestrated microservices-based scalable platform
- Cloud Native — (alternative Amazon/AWS cloud to dramatically reduce operational costs)
- In-memory data grids
- Zero downtime and zero touch operations
- Automated self-scaling/-healing
- AI driven closed-loop performance improvements

Functional Enablers

- Complex multi-faceted product and service bundles across value chains
- On-demand B2B offers and collaborative bundles
- Dynamically changing partner ecosystems and corresponding revenue-sharing models
- Real-time charging for any user or device-based event or attribute
- AI-driven, consumption-based online quotation, pricing, config.
- Slice-based and cross-slice charging models

THE PATH TO 5G



5G network slicing

5G network slicing enables service providers to build virtual end-to-end networks tailored to application requirements.

4G networks do not enable the range of services that the future requires. 5G will be faster and more flexible.

IoT: Internet of things

THE PATH TO 5G

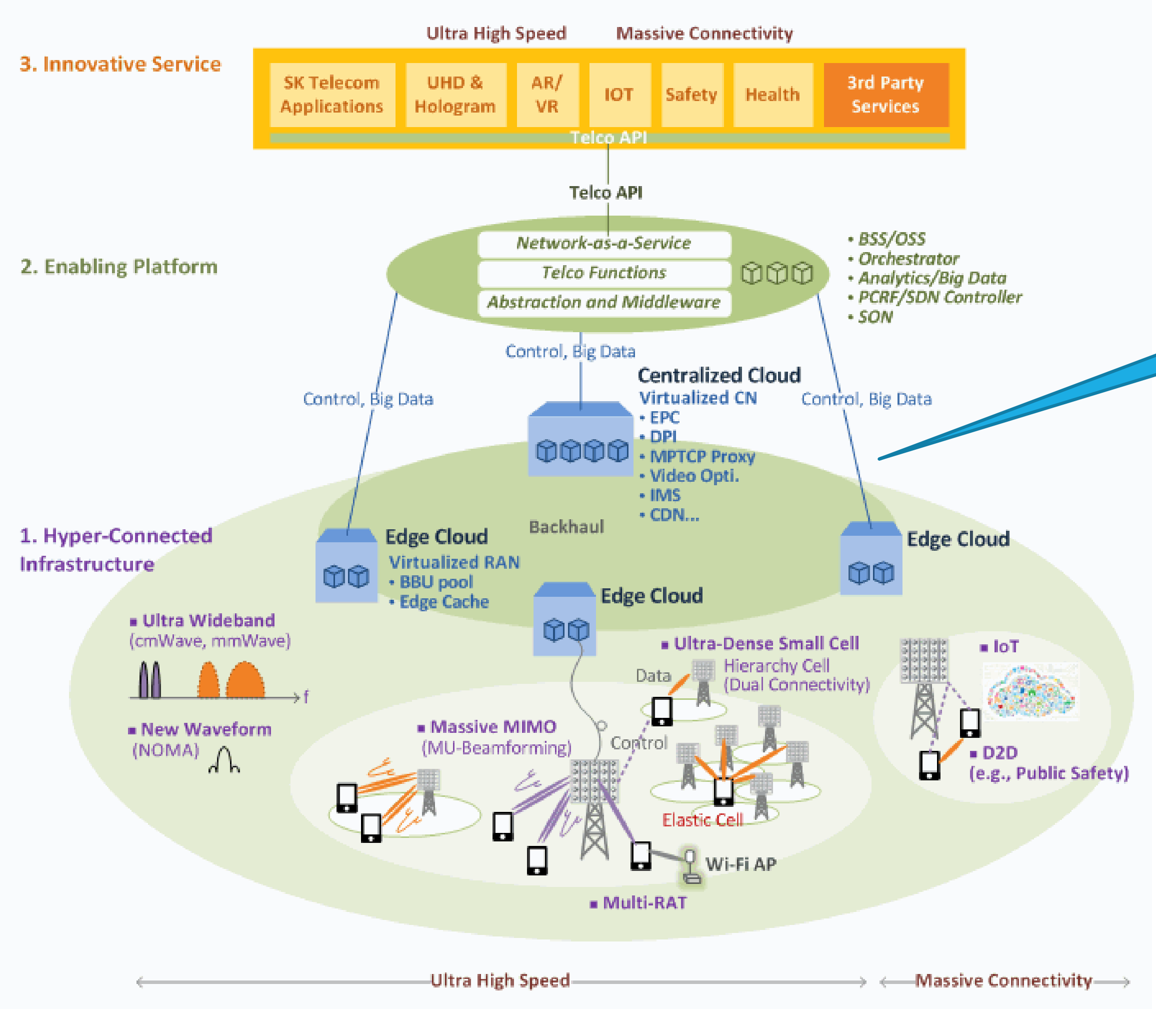


Figure 3. SK Telecom's 5G architecture (Source: SK Telecom's 5G whitepaper - Re-illustrated by Netmanias)

They envision a service infrastructure of their own living on Azure IoT Edge or similar!

Is 5G just an IoT cloud integrated more closely to telecommunications?

MOBILITY AND GEOREPLICATION

A multipath TCP session (like it sounds) is one where a single TCP connection could route over any of several pre-selected paths.

This can have some surprising latency effects, which multipath TCP "hides". Moreover, without "help" from some form of stable intermediary service, a mobile client could end up switching from zone to zone which might expose inconsistencies.

5G services live in a more stable environment, and become the mobile user's cache, and a gateway to the full cloud, much like the IoT sensor proxies we discussed.

SUMMARY

"...he doth bestride the narrow world/Like a Colossus..."

GeoReplication is best viewed as having two scales:

- Availability Zone: Just neighboring data centers. With some cost, you can use TCP and build "normal" protocols. Derecho should work this way.
- True global scope: Here, techniques like Google Spanner are best. Researchers have experimented with Paxos at global scope, but it performs poorly due to high-latency links.
- The notion of ***wide-area stability*** is useful and might be worth using in other contexts.
- 5G mobility will introduce an additional layer of services