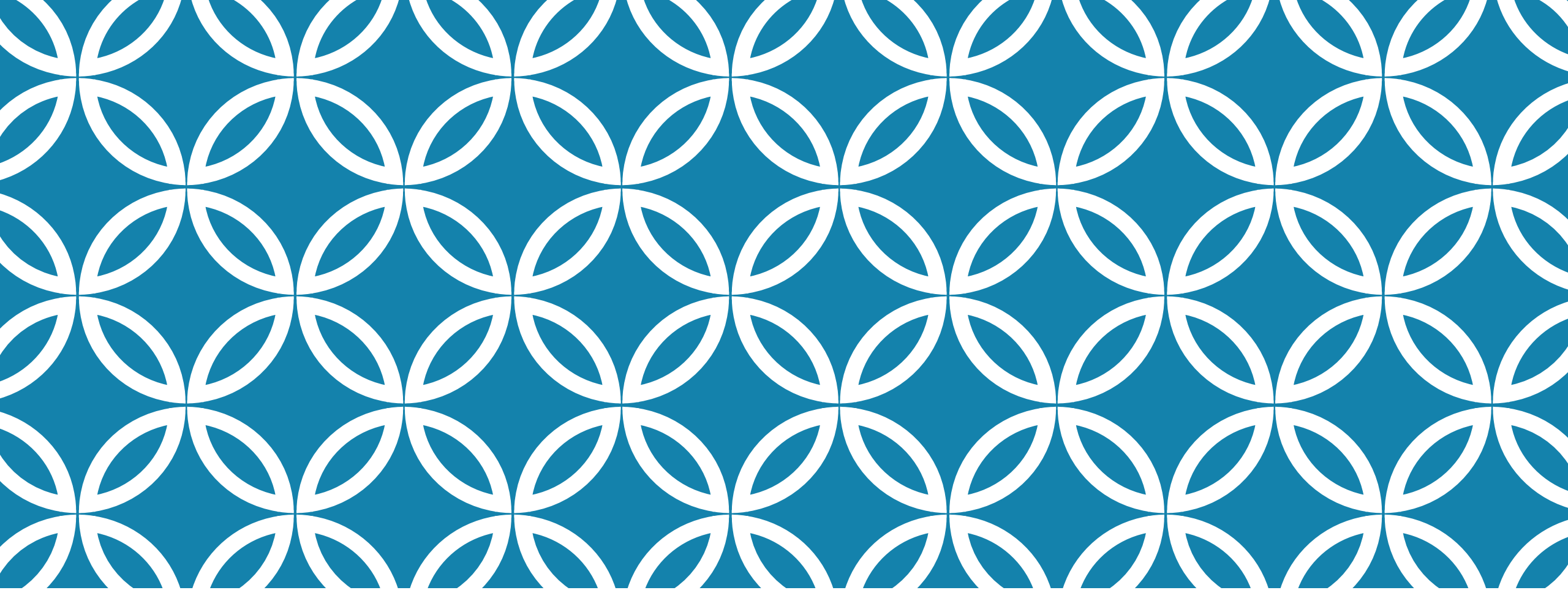


CS5412 / LECTURE 16 BLOCKCHAINS WITH MULTIPLE ORGANIZATIONS

Ken Birman

Spring, 2021

TODAY: BRAINSTORMING ABOUT A REAL BLOCKCHAIN "USE CASE"!

Consider the challenge of running a multi-hospital consortium structure in which one patient might be seen by physicians at more than one different medical center.

Example: In New York City, the "tri-institutional medical consortium" includes Cornell Weill, Sloan Kettering, NYU

A patient with a complex condition could have treatments in all three

EXAMPLE

Mrs. Sally Smith is admitted for abdominal pain at Cornell Weill. Various diagnostic procedures are performed. Cornell starts treating an infection.

She turns out to require surgery, which is done in the gastroenterology surgical unit at NYU.

There is a followup cancer treatment at the Sloan Kettering Medical Center. Meanwhile, she also has infection followups at Cornell, and surgical post-op visits to NYU.

THE ISSUE? 3 SETS OF MEDICAL RECORDS!

Each of these organizations has a distinct electronic medical record management system, which for legal reasons (HiPPA) must be managed by the individual hospital.

Yet each also needs a way to see the records created by the others, in order to ensure that the complex condition Mrs. Smith is being treated for is correctly managed.

HiPPA doesn't deal very well with this form of sharing.

MEDICAL RECORDS AND THE CLOUD

What roles can the cloud play in medical record management?

In early EMR systems, the answer was very simple: *none*. For a long time, medical records systems were treated as "on your own premises only".

But over time, this became more and more expensive and unworkable. Eventually, a form of hybrid cloud emerged.

TERM: HYBRID CLOUD

We have seen this mentioned in past lectures.

A hybrid cloud is any system that combines elements from the home system and the cloud, or even from multiple distinct cloud providers.

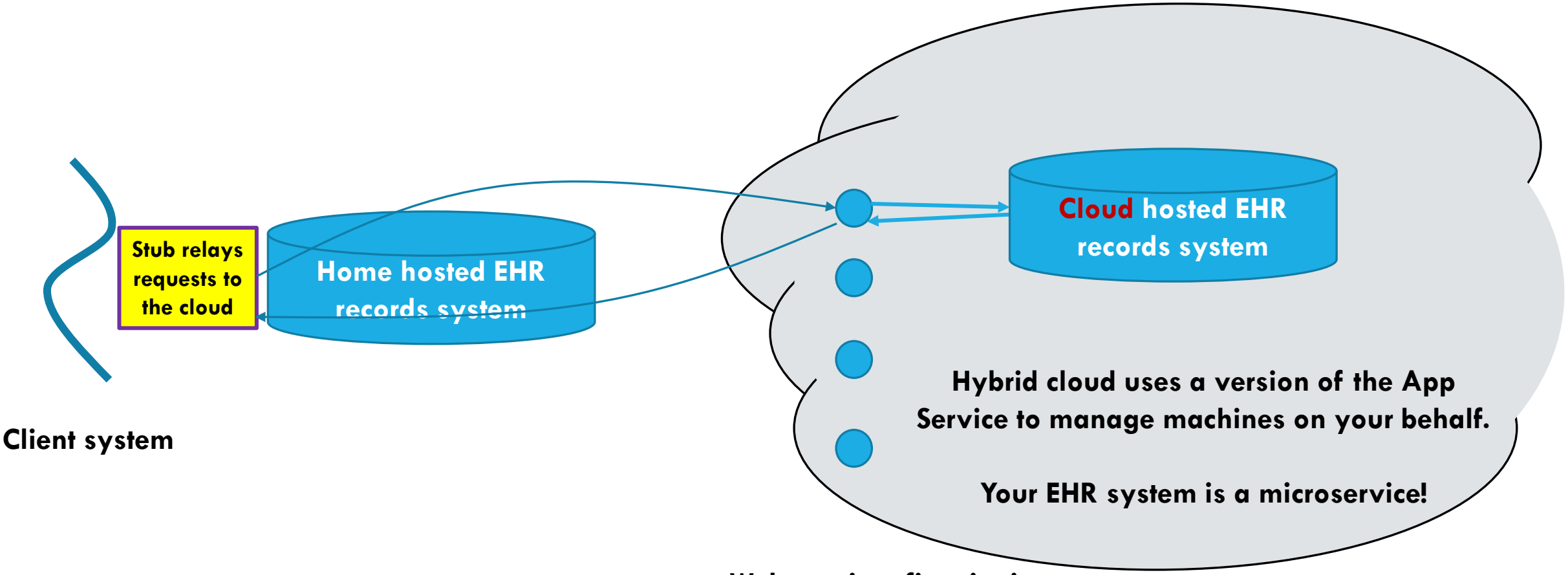
TERM: HYBRID CLOUD

A hybrid cloud is any system that combines elements from the home system and the cloud, or even from multiple distinct cloud providers.

These could run on the home system or be migrated to the cloud.

For example, a company could take a database that used to run on its own servers and "port" it to run on AWS. Hybrid cloud allows them do this in a very transparent way.

PORTING AN ENTERPRISE SYSTEM TO A CLOUD



PORTING AN ENTERPRISE SYSTEM TO A CLOUD

We end up "redirecting" what used to be remote procedure calls from clients to the home-hosted server. Now they become web-service requests forwarded to the cloud, and the answers come back in the same web-service format.

But this can be hidden in the library used by clients to talk to the service.

In principle, you just recompile and things work as they did previously!

WHY WAS THIS BENEFICIAL?

Notice first that it could actually be slower than the original solution!

But modern networks are fast, and network delay isn't the only concern.

The cloud has:

- Cutting edge hardware,
- Automated management tools
- Sharing (amortized costs)
- Huge storage and compute capacity

SO…

By adopting the cloud, the hospital can change the computer room into some other use of the space, like a surgical suite.

The IT staff can become more effective and won't need to do as much hands-on management of the servers.

Upgrades are done by the cloud vendor, not by the IT staff.

DOWNSIDES?

Dennis… the evil hacker

The hospital can't access its own EHR records unless the network is up and the cloud provider is operating normally.

Security and privacy aspects of HiPPA: What if Amazon has an evil employee who plans to sell data about celebrity patients?

What if the disk on which data is stored gets upgraded and the old one isn't wiped clean?

SOME REASSURING CLOUD FEATURES

The web connections use TLS (https): all the data is encrypted

The cloud itself uses the Virtual Private Cloud concept we discussed. Your microservice lives in an isolated "security environment".

All the files stored by your application are automatically encrypted by the file system before being written, and the keys are kept at your home system, not on the cloud. So if the disk were stolen, it can't be deciphered.

TECHNOLOGY REALLY CAN HELP

By splitting secret keys among multiple repositories, we can guard against failure that could otherwise destroy a key, while also ensuring that k out of N portions are needed to reconstruct the actual key.

In a secure cloud, keys aren't kept in memory for more than a few microseconds. Instead, any key we will use for a long period is "mapped" to a different key managed by the hardware keys built into the trusted computing hardware module (TCB). This temporary key can only be used on a particular machine, with the help of the hardware itself.

MORE REASSURING FEATURES

There are also human-factors steps taken by the big vendors.

They ensure that any person who has access to secure keys doesn't have access to the physical data center, and that any person who has a way to modify the operating system code can't access keys or hardware.

There are also steps taken to ensure that connectivity from the home site to the cloud is redundant, from multiple ISPs (tunneling over mTCP).

WHERE DOES THIS GET US?

Individual organizations, like hospitals, have some elements of their electronic health records on the cloud, and the trend is to move more and more functionality over time. HiPPA concerns slow this down, but it is advancing even so.

But even if they happen to be on the same cloud, they can't just share records with one-another other than by creating specialized mirroring.

HOSPITAL Y TREATS A PATIENT FROM HOSPITAL X

In today's approach, each pair of cooperating hospitals establishes a form of secure replication channel.

For example, X might send records about Mrs. Smith to Y, specifically needed for Y to carry out this treatment.

Y might then share back some of its records of the treatment, so that X has the needed data to track her overall plan of care.

X AND Y MUST AUDIT ALL SHARING

Each hospital has a legal obligation to track which records have been shared (outgoing or incoming), and precisely why.

This extends to other organizations too, such as laboratories that do testing, private practices where physicians might see some patients in an office setting, etc.

It is natural to think of Blockchain as a technical tool for such uses.

BLOCKCHAIN: A "PARTIAL" MATCH

The tamperproof audit trail aspects are a good fit here.

We can track these EHR record transfers in a blockchain, and it gives us a proof of which records moved from institution to institution.

We can also track individual accesses to those records. Now we know that Dr. Marshall accessed the records of Mrs. Smith from hospital Y.

THE REAL PROBLEM IS THAT A BLOCKCHAIN ISN'T A DATABASE!

For higher level machine-learning tools that could ask "is it appropriate for Dr. Marshall to make this access?", we would need more of a database representation!

In fact, in general, both human and computer users of this data will want to query it.

But a blockchain isn't a database!

THERE IS A LOT OF WORK TO CLOSE THE GAP

Many companies offer SQL interfaces so that their blockchains can be viewed as if they were databases.

This isn't always very performant, but by creating secondary indices (like our B+ tree from hw2), we can ensure that common query patterns are executed very rapidly.

Then an administrator could, for example, check to see whether Dr. Marshall made this access as part of a treatment activity.

WOULD THIS DEFEAT ATTACKERS?

Remember Dennis Nedry… he wrote the code!

Dennis wants you to *think* everything is being audited and tracked, but *actually* he plans to steal Mrs. Smith's health records and sell them without being caught.

If the records themselves are encrypted, he might be blocked, but a legitimate medical access might be delayed or blocked too…

DENNIS THE EVIL ATTACKER

Dennis could even modify the audit-query code to try and fool you.

"Chief security officer: Check that all EHR record accesses are properly authorized."

"System: Confirmed, no problems detected."

A SKEPTIC MIGHT WANT MORE PROOF

… but here things get very difficult.

Would we show the skeptic the entire audit trace?

Or could we somehow construct a machine-checkable proof that the query response is a correct and complete one?

WAN MIRRORING CAN HELP!

Suppose that as we generate our blockchain, we also make a lot of mirror copies on other datacenters. They "countersign" the replicas.

Thus if WAN datacenter X logs event A, a skeptic can actually fetch the signed log from X but can also get countersigned mirror copies at Y, Z…

Much harder for Dennis to hack this system: one a record is logged he won't easily be able to prevent the skeptic from noticing any tampering!

OTHER SETTINGS WITH SIMILAR NEEDS

Farm to table tracing of the food chain: we might want to track all the main events between when a load of spinach was harvested to when it reached the consumers.

If an e-Coli outbreak occurs, we would want to show that "on the Smith family farm, these cases were transported on a pickup truck that previously was used to transport a load of cow manure."

But it can be extremely hard to capture enough state to do that!

MORE SUCH SETTINGS

Inter-bank transactions

Supply chain tracking for industrial activities such as manufacturing plants

Tracking the maintenance of hardware in the field, for things like power grid technology with very sensitive roles

MANY SUCH CASES NEED EVEN MORE!

The smart contracts concept from Ethereum can appear to be a great fit for banking transactions or other complex kinds of records.

But here we trust that the Ethereum "code" describing the event is correct and unambiguous.

The potential of a blockchain to roll back (instability of the last records) is especially worrying.

EXAMPLE OF HOW ROLLBACK CAUSES ISSUES

Suppose that Dan sells chewing gum to Frank for $1.

Our blockchain record may record that "upon receipt of $1 from Frank, Dan will hand Frank the chewing gum."

But now Frank decides to earn some money. He washes Sally's car and for this she will pay him $3, and he will use $1 of this to buy the gum. The record is added to the blockchain.

… WE CAN CREATE ENDLESS DEPENDENCIES!

… actually, Sally wrote him a check. Frank needs to deposit it.

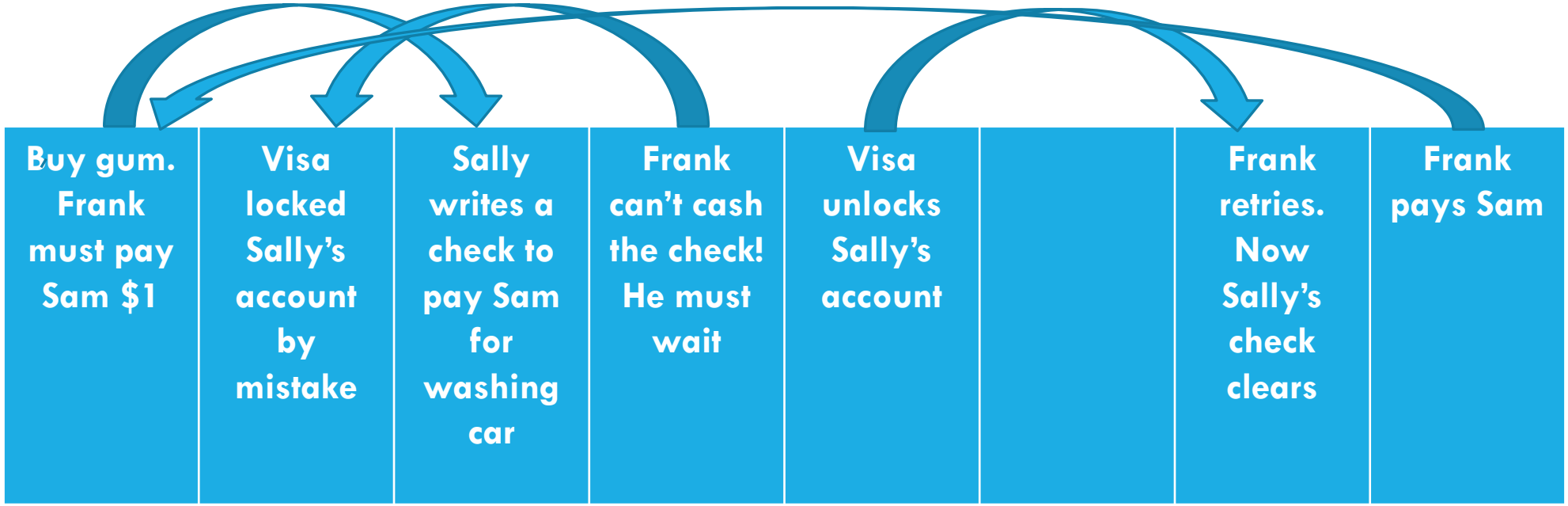
… the bank needs to clear the check.

… Sally's bank account has plenty of money, but is blocked by her visa card, which is trying to collect what they claim is an unpaid bill.

… the Visa card company actually made an error. Sally did pay the bill (someone else with the same first and last name is in arrears). They agreed to fix this, but haven't done so yet.

… WE CAN CREATE ENDLESS DEPENDENCIES!

Visualizing these dependencies. Remember: the blockchain can roll back at any time, erasing the last few records!



CASCADING EFFECT OF ROLLBACKS

The last 5 or 6 blocks of records are considered unstable, meaning they could be overwritten by other records.

Blockchain becomes stable after about an hour, but in Frank's case, that won't be enough. He needs the last record in the chain to stabilize!

Even worse: what if the amount being paid depends on some future thing?

FUTURE DEPENDENCY

Frank, Sally and Dan are creating a startup company. It will revolutionize newspaper stands in big cities, and they will split the insane profits.

They call it "stand". Even the name will be worth ~~millions~~ billions.

The blockchain record of the agreement might say: "when stand.com is sold, Frank and Dan will receive 30% of the proceeds each, after deducting costs, and Sally will receive 40%."

… BUT THE "AMOUNT" COULD VARY!

Depending on the records added to the chain in the future, the costs could change, and hence the amounts they will be paid could change.

What are the semantics for "costs"?

When should costs be evaluated? When can the transaction finalize?

CAN IOT HELP?

On the positive side, we can capture records from our IoT devices and store them into the blockchain, too. We can even have them signed by the devices to "prove" that this is really what the device detected.

And we could have enough sensors to make it hard to conceal things.

But ultimately, if Dennis Nedry understands how the system works, he won't have any real difficulty evading it.

BIGGEST CONCERN OF ALL

What Rumsfeld called the “Unknown unknowns”.

It ain’t what you know that’ll get you. It’s what you know that just ain’t so.

-- Attributed to Samuel Clemens (Mark Twain)

PATTERNS OF TRUST

One big area of discussion relates to situations where there are multiple blockchains managed by different organizations and you don't have equal trust in all of them.

For example, maybe an Ithaca resident only trusts credit union banks, but not big banks. Meanwhile, a hard core Wall Street capitalist only trusts really big banks and would never trust CFCU or AFCU.

So how can the Ithaca person get a mortgage that the Wall Street person is actually issuing? Somehow their different blockchains need to talk to each other!

VEGVISIR WON'T SOLVE THIS

The model in Vegvisir is basically "one blockchain we all share".

So there is work at a company called Stellar to create a different model with many blockchains. The key idea is to use a version of a state machine replication protocol called RAFT to replicate blockchain records across the set of blockchains.

If person P trusts {X,Y,Z} then any record associated with person P is required to be stored by a quorum (majority) of the set {X,Y,Z}.

STELLAR WITH TWO PEOPLE

What if a record involves this mortgage that P is basically obtaining from Q? Then we take their two trust sets: P trusts {A,B,C,D,E}. Q trusts maybe {X,Y,Z} (or they can share some trust, this is fine too).

Stellar asks for a majority from *each of the trust sets*. So we end up putting the mortgage record into a quorum from {A,B,C,D,E} and also a quorum from {X,Y,Z}. Now anyone P "cares about" would be able to see the record. And similarly for anyone Q trusts.

ISSUES WITH STELLAR

If the trust sets are static, the idea is pretty simple and it is obvious that it will work in a simple way.

But suppose that trust varies over time. Maybe CFCU and AFCU merge, or perhaps that big banker decided to not trust Deutsch Bank anymore.

So now the quorum sets aren't constant. We have to "re-replicate" to preserve our invariant. With concurrency and failures it gets messy!

COULD WE USE VIRTUAL SYNCHRONY?

Reminder: Derecho is using a model called virtual synchrony in which first we maintain the membership of a system, as an initial background task.

Then we run protocols during epochs while the membership is constant.

The protocols are simplified, but this implies agreement on a system-wide membership protocol, which probably must tolerate attackers ("Byzantine" model) because after all, the Wall Street guy doesn't trust anyone else.

CAN WE ACTUALLY CONCLUDE ANYTHING?

Mostly, the conclusion here is that the enthusiasm for blockchain is way ahead of the reality of the standard systems!

Everyone is going wild dreaming up fancy use cases. The founders of Ethereum are worth billions! Yet Ethereum has fairly low levels of revenue.

Bottom line: the actual technology doesn't match the fancy scenarios.

ONE OF KEN'S "TALES OF WOE"

We take client-server computing for granted… but it was nearly on one of those gravestones!

In the earliest days, Digital Equipment Corporation invented client-server with the introduction of their VaxClusters architecture. A huge advance!

The market was exponential – DEC was on track to become the largest computer company ever. *Then it suddenly imploded and was acquired!*

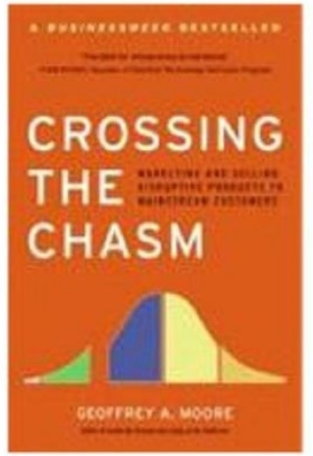
WHY DID EARLY CLIENT-SERVER FAIL?

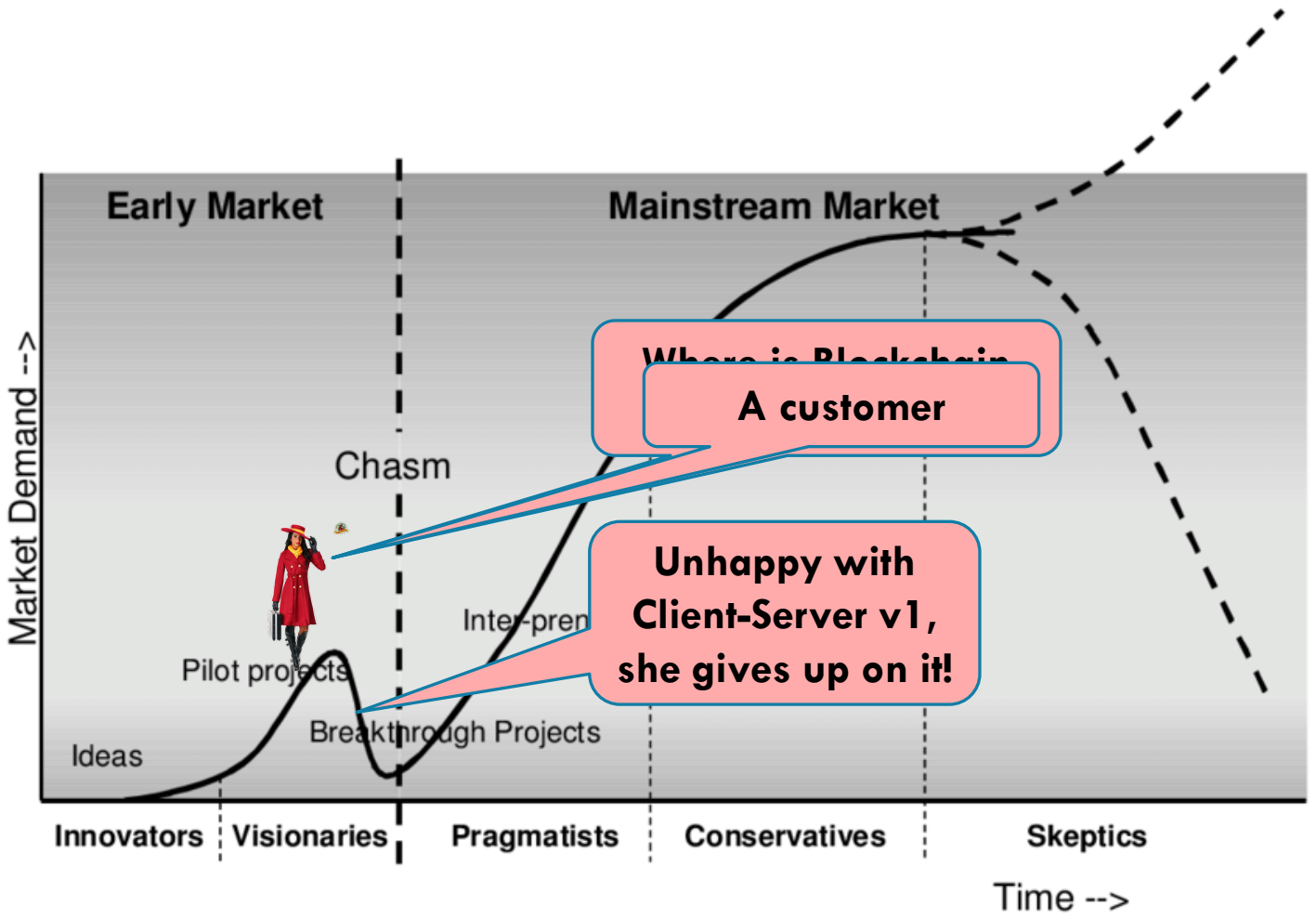
The concept was absolutely right! Yet the customers were unhappy.

The early products felt like a research prototype, not professional.

The “life cycle” of a full client-server deployment had not yet been thought through, hence there were a huge number of missing tools and features.

CROSSING THE CHASM (MOORE, 2009)





FACTORS THAT LIMIT UPTAKE

Early enthusiasm / bleeding edge always moves on to the next better thing, so the first adopters are certain to wander away.

Conservative customers want to be “the first to be last” and wait for the mainstream uptake to occur.

If you move too quickly, you simply overextend and fall into the chasm.

SUMMARY

Real world uses that seem ideal for IoT and Blockchain are very common

Yet it can be a real puzzle to actually build the solutions!

Simply seeing a match doesn't really *solve* anything. The details are hard!