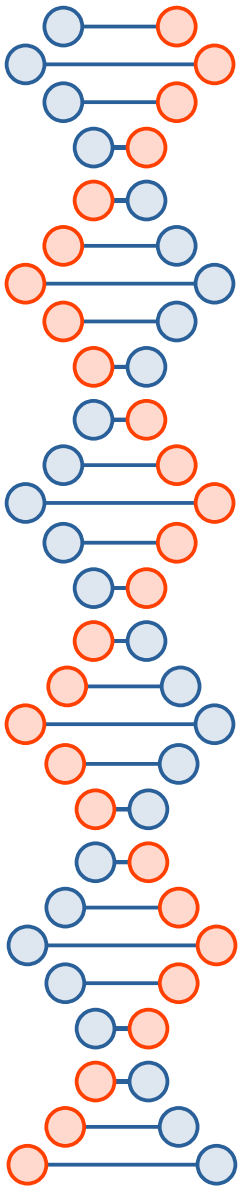


More about distributed systems…

jedimaestro@asu.edu

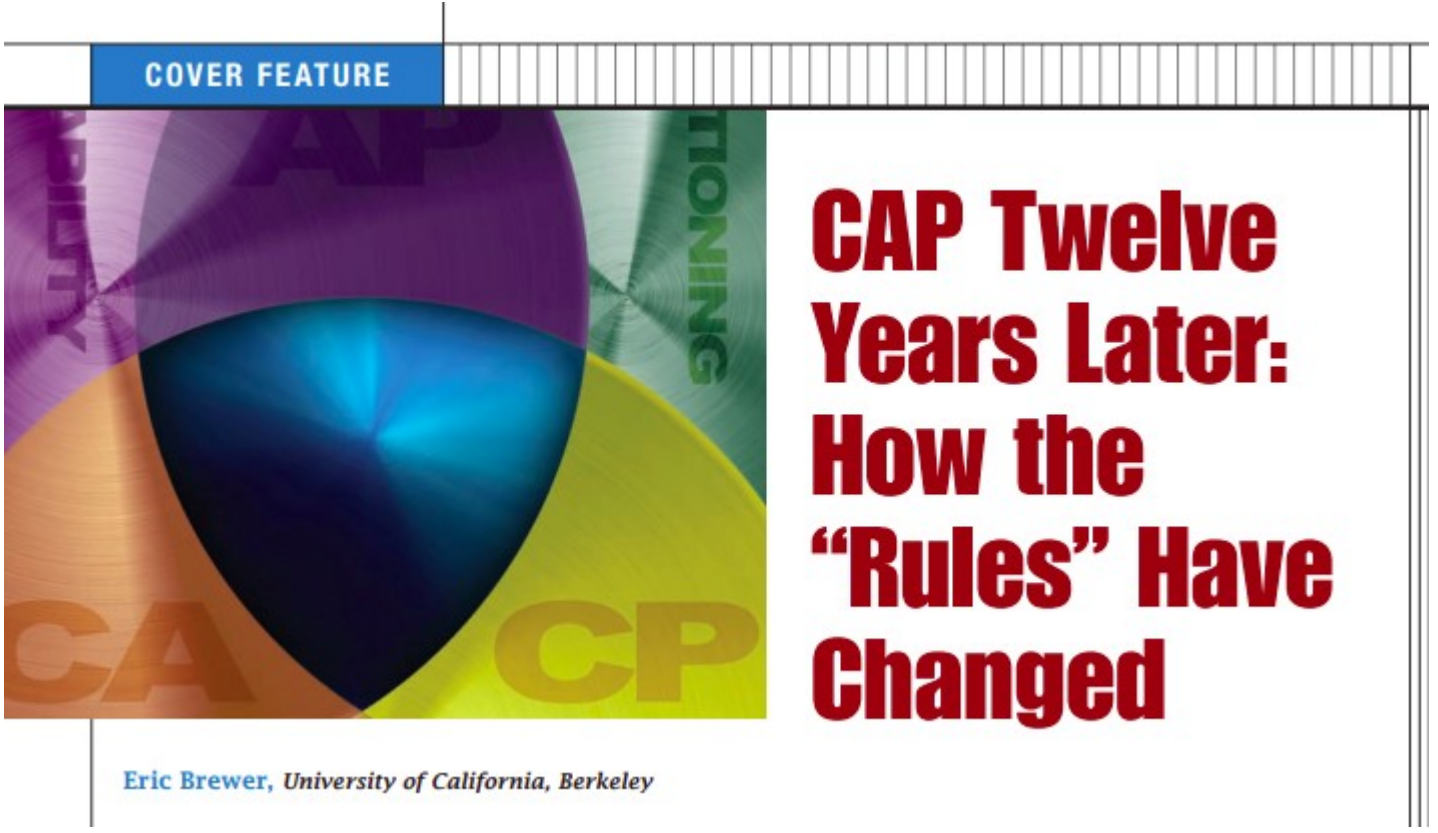
CSE 536 Fall 2026



In these slides...

- CAP revisited
- PACELC
- The FLP Impossibility Result
- Some more real-world examples
  - Discord or ride service for message passing?
  - Netflix uses gRPC and message passing

https://sites.cs.ucsb.edu/~rich/class/cs293b-cloud/papers/brewer-cap.pdf



Richard Feynman's core philosophy on testing ideas is that if a theory disagrees with experiment, it is wrong, regardless of its beauty or the author's intelligence. The key is to never fool yourself, as you are the easiest person to fool. Ideas must be validated by questioning, experimentation, and comparing results with nature. Speakola +5

Key Feynman Quotes on Testing Ideas

- **The Ultimate Test:** "If it disagrees with experiment, it is wrong. In that simple statement is the key to science".
- **Avoiding Self-Deception:** "The first principle is that you must not fool yourself — and you are the easiest person to fool".
- **Questioning Answers:** "I'd rather have questions I can't answer than answers I can't question".
- **Intellectual Honesty:** "...if you're doing an experiment, you should report everything that you think might make it invalid — not only what you think is right about it".
- **On Understanding vs. Memorization:** "Without using the new word which you have just learned, try to rephrase what you have just learned in your own language".
- **The Role of Doubt:** "The question of doubt and uncertainty is what is necessary to begin".
- **Challenging Authority:** "Have no respect for authority". Speakola +6

The CAP theorem states that any networked shared-data system can have at most two of three desirable properties:

- *consistency* (C) equivalent to having a single up-to-date copy of the data;
- *high availability* (A) of that data (for updates); and
- tolerance to network *partitions* (P).

WHY "2 OF 3" IS MISLEADING

The easiest way to understand CAP is to think of two nodes on opposite sides of a partition. Allowing at least one node to update state will cause the nodes to become inconsistent, thus forfeiting C. Likewise, if the choice is to preserve consistency, one side of the partition must act as if it is unavailable, thus forfeiting A. Only when nodes communicate is it possible to preserve both consistency and availability, thereby forfeiting P. The general belief

As the "CAP Confusion" sidebar explains, the "2 of 3" view is misleading on several fronts. First, because partitions are rare, there is little reason to forfeit C or A when the system is not partitioned. Second, the choice between C and A can occur many times within the same system at very fine granularity; not only can subsystems make different choices, but the choice can change according to the operation or even the specific data or user involved. Finally, all three properties are more continuous than binary. Availability is obviously continuous from 0 to 100 percent, but there are also many levels of consistency, and even partitions have nuances, including disagreement within the system about whether a partition exists.

CAP-LATENCY CONNECTION

In its classic interpretation, the CAP theorem ignores latency, although in practice, latency and partitions are deeply related. Operationally, the essence of CAP takes place during a timeout, a period when the program must make a fundamental decision—the *partition decision*:

- cancel the operation and thus decrease availability, *or*
- proceed with the operation and thus risk inconsistency.

Retrying communication to achieve consistency, for example, via Paxos or a two-phase commit, just delays the decision. At some point the program must make the decision; retrying communication indefinitely is in essence choosing C over A.

https://martinfowler.com/articles/patterns-of-distributed-systems/two-phase-commit.html

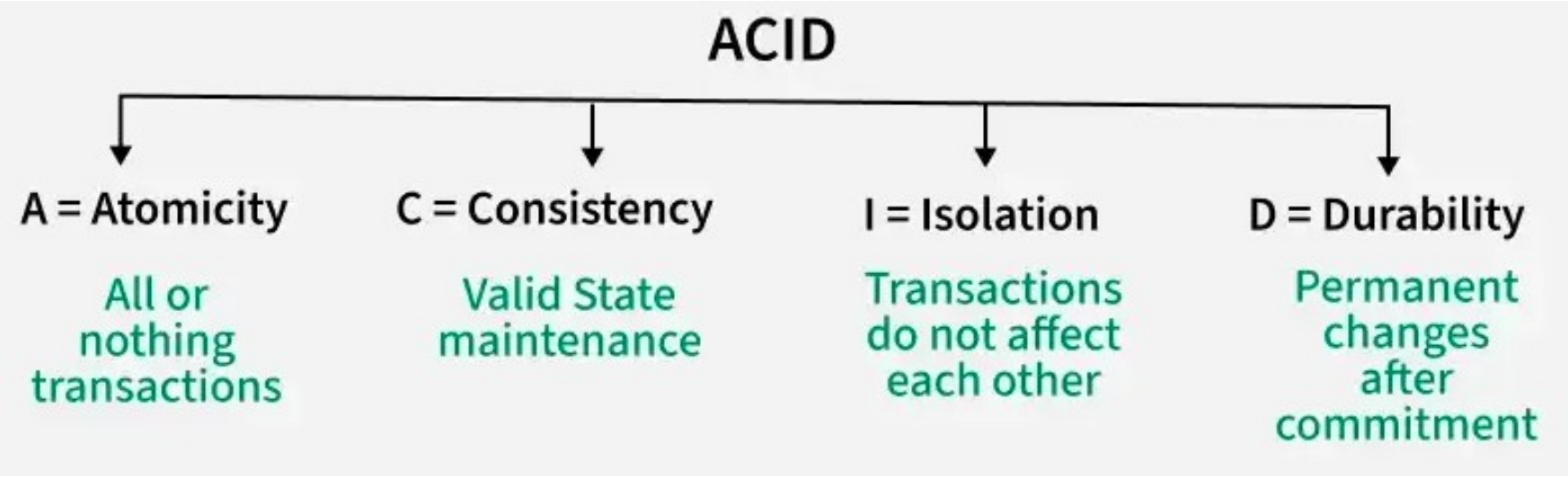
The essence of two-phase commit, unsurprisingly, is that it carries out an update in two phases:

1. The prepare phase asks each node if it can promise to carry out the update.
2. The commit phase actually carries it out.

Wikipedia: “The greatest disadvantage of the two-phase commit protocol is that it is a blocking protocol.”

Thus, pragmatically, a partition is a time bound on communication. Failing to achieve consistency within the time bound implies a partition and thus a choice between C and A for this operation. These concepts capture the core design issue with regard to latency: are two sides moving forward without communication?

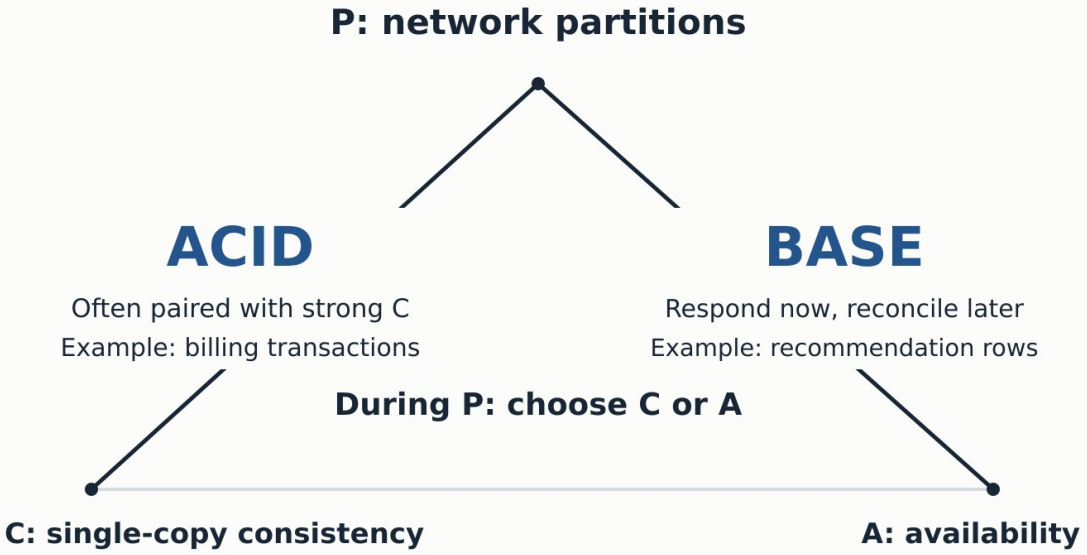
https://www.geeksforgeeks.org/dbms/acid-properties-in-dbms/



Although both terms are more mnemonic than precise, the BASE acronym (being second) is a bit more awkward: Basically Available, Soft state, Eventually consistent. Soft state and eventual consistency are techniques that work well in the presence of partitions and thus promote availability.

CAP, ACID / BASE, REST / SOAP

Data guarantees and service interfaces are separate choices



ACID: Atomicity, Consistency, Isolation, Durability

BASE: Basically Available, Soft state, Eventually consistent

How services communicate

REST

Representational State Transfer

Architectural style for resources

Stateless requests

```
GET /users/sam/recommendations
```

SOAP

XML messaging protocol

Structured envelopes and faults

```
<GetRecommendations user="Sam"/>
```

Either can front ACID or BASE.

RPC, SOAP and REST

Application-specific actions versus resources with standard operations

RPC: invoke a named procedure in another process

SOAP used for RPC

Name the action

RPC-style operation

```
GetRecommendations(user="Sam")
```

Application-defined operation + parameters

SOAP carries the call in an XML envelope.

SOAP also supports document-style messages.

SOAP is a messaging protocol.

REST

Name the resource

HTTP request

```
GET /users/sam/recommendations
```

Resource URI identifies the thing.

HTTP method supplies the verb:

`GET, POST, PUT, DELETE`

REST is an architectural style.

Both can use request/reply and must handle network failures.

Using HTTP or JSON alone does not make an API RESTful.

CAP vs. PACELC

PACELC adds a latency tradeoff even when the network works.

CAP

IF a partition occurs (P)

A vs. **C**

Availability
Reply on both sides
Data may disagree

Consistency
One-copy order
Some requests wait



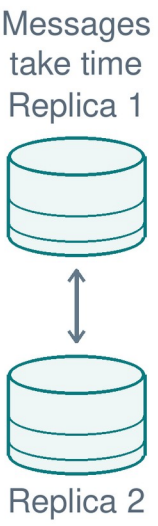
PACELC adds

ELSE, no partition (E)

L vs. **C**

Low latency
Read a nearby copy
Data may be stale

Consistency
Coordinate replicas
Pay network delay

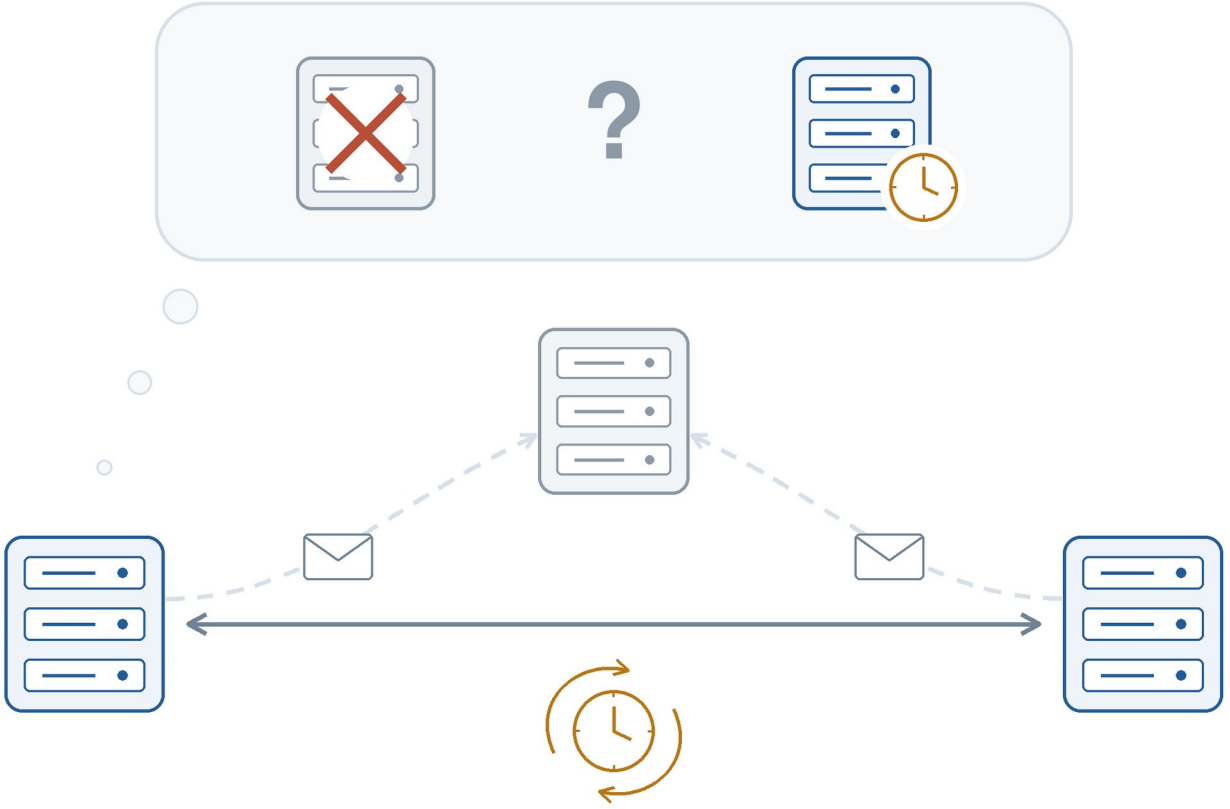


PACELC = IF P: A vs. C ELSE: L vs. C

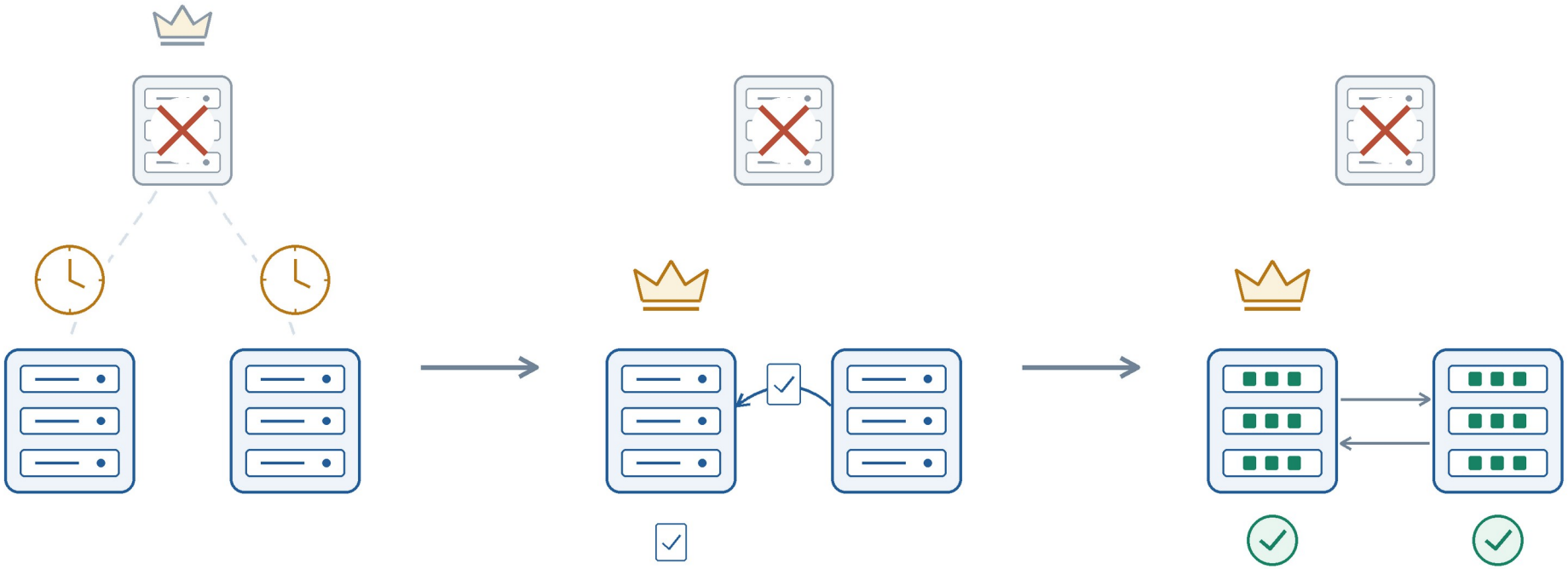
C = linearizability: operations act on one up-to-date logical copy.

Brewer, “CAP Twelve Years Later” (IEEE Computer, 2012).
Abadi, “Consistency Tradeoffs in Modern Distributed Database System Design” (IEEE Computer, 2012), doi:10.1109/MC.2012.33.

Fischer–Lynch–Paterson (FLP): With unbounded delays and even one possible crash, no deterministic algorithm can guarantee consensus will finish.

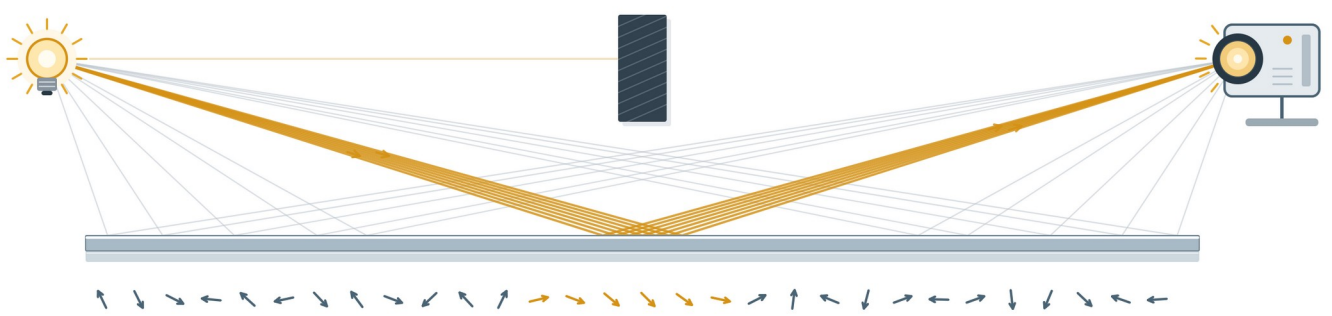


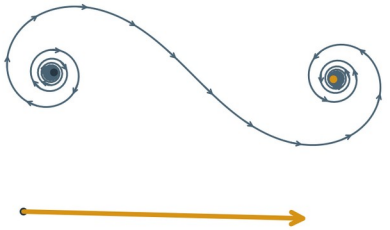
Raft / PAXOS work despite the FLP result

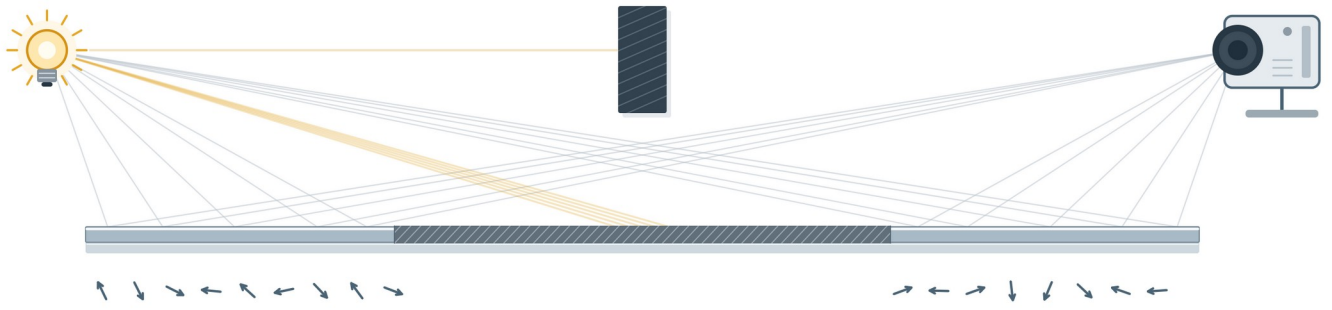


E.g., Raft preserves agreement during delays and makes progress once a live majority can elect and maintain a leader with timely communication.

A short digression, just to point out that in this class we’re assuming classical physics...

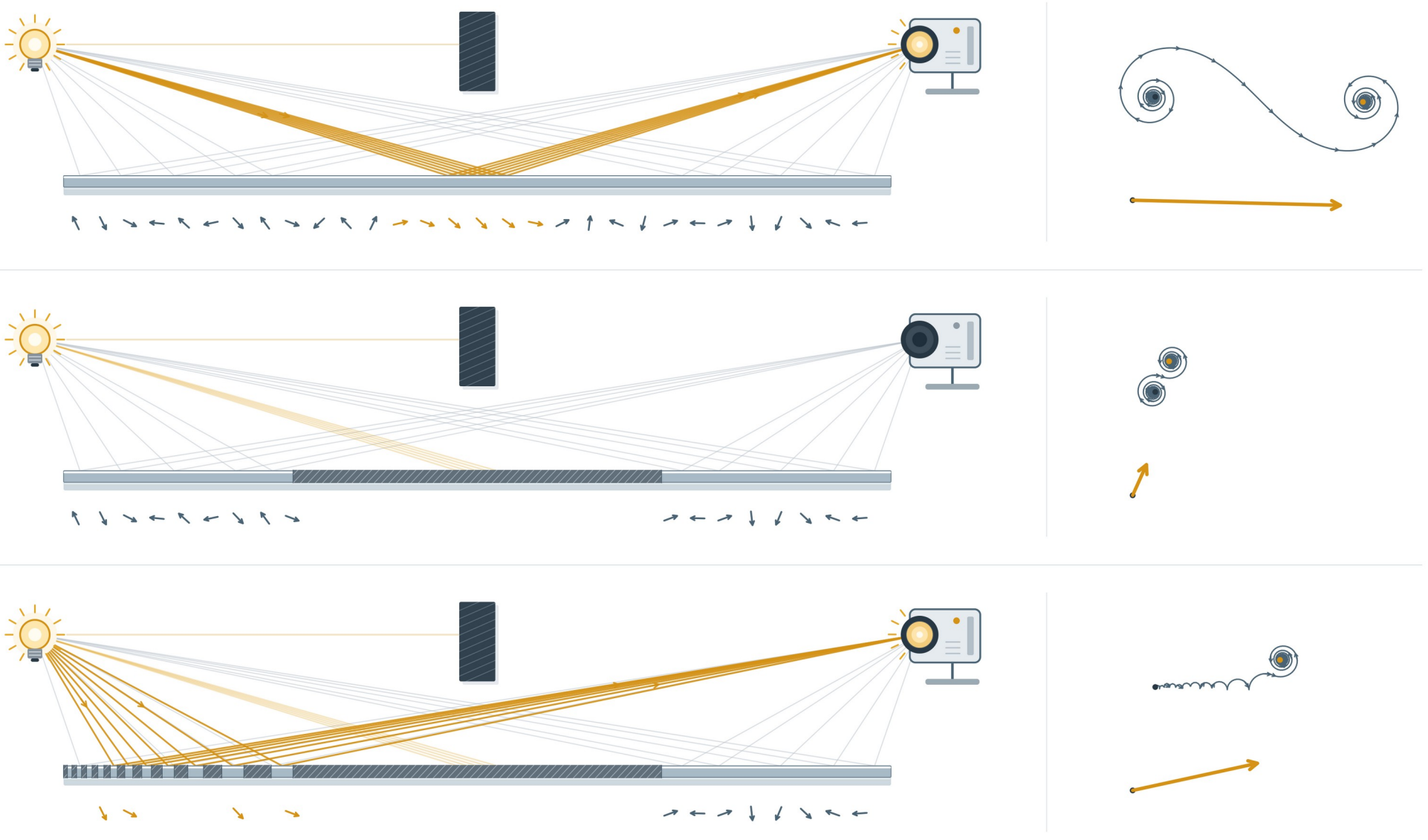








Can you get light to reach the detector again
by scratching away more of the mirror?



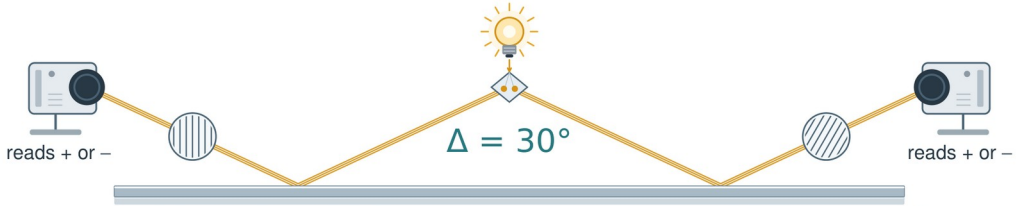
1. Unentangled photons

Independent, unpolarized photons. Each detector is 50/50 at every angle.

Left: 0°

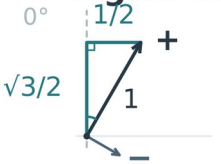


P(+) = ½



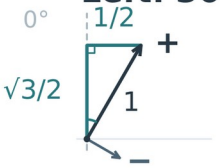
½ × ½ + ½ × ½ = 50% same **50% opposite**

Right: 30°

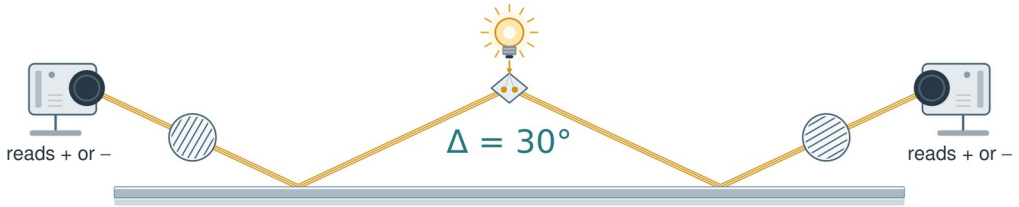


P(+) = ½

Left: 30°



P(+) = ½



½ × ½ + ½ × ½ = 50% same **50% opposite**

Right: 60°

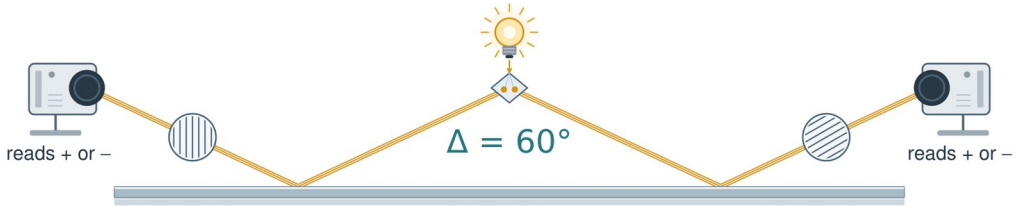


P(+) = ½

Left: 0°



P(+) = ½



½ × ½ + ½ × ½ = 50% same **50% opposite**

Right: 60°



P(+) = ½

Bell’s inequality

D(0°, 60°) ≤ D(0°, 30°) + D(30°, 60°)

Perfect same-angle agreement. Independent settings. No communication during measurement.

D = probability of opposite signs

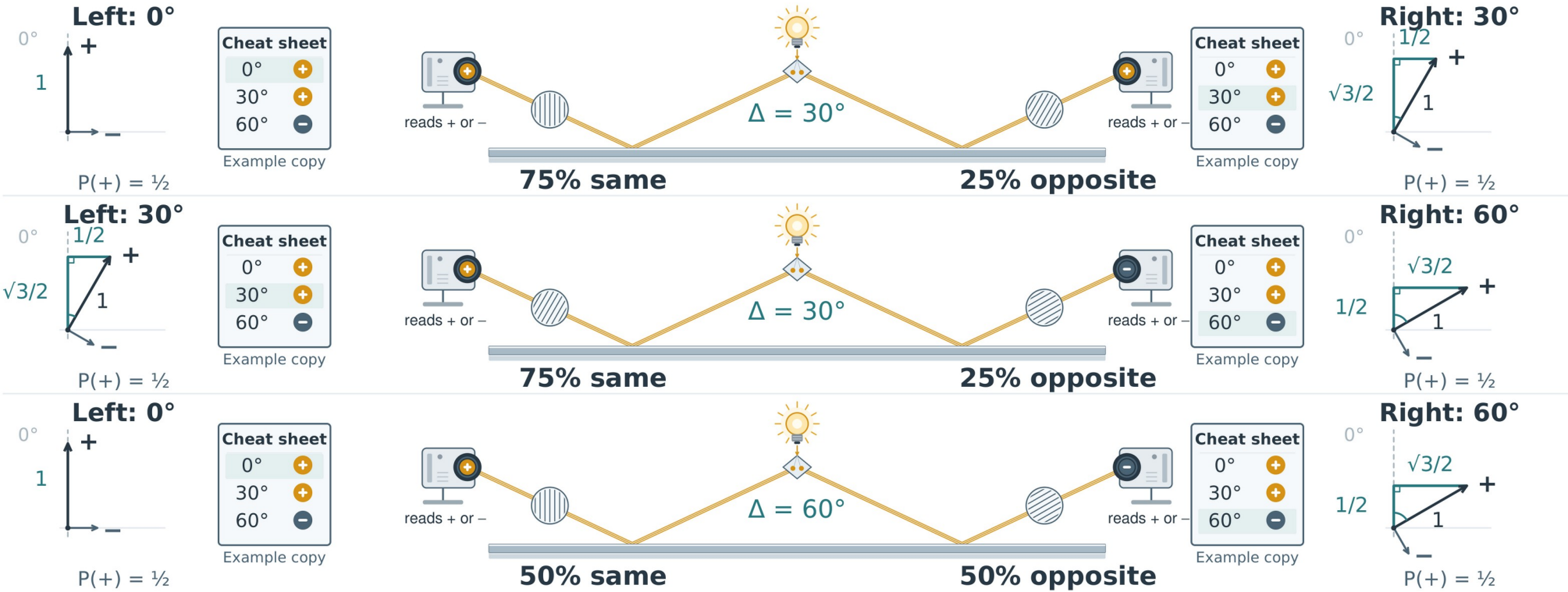
Same angle: only 50% match.
Independent signs explain
50% opposite at every setting.
Perfect matching is missing.

Matching cheat-sheet limit — N/A: same-angle premise fails

Opposite at 0° / 60° — **50%**

2. Unentangled + cheat sheets (local hypothesis)

Matching prewritten signs. Random sign flips keep each detector 50/50.



Bell’s inequality

$$D(0°, 60°) \le D(0°, 30°) + D(30°, 60°)$$

Perfect same-angle agreement. Independent settings. No communication during measurement.

D = probability of opposite signs

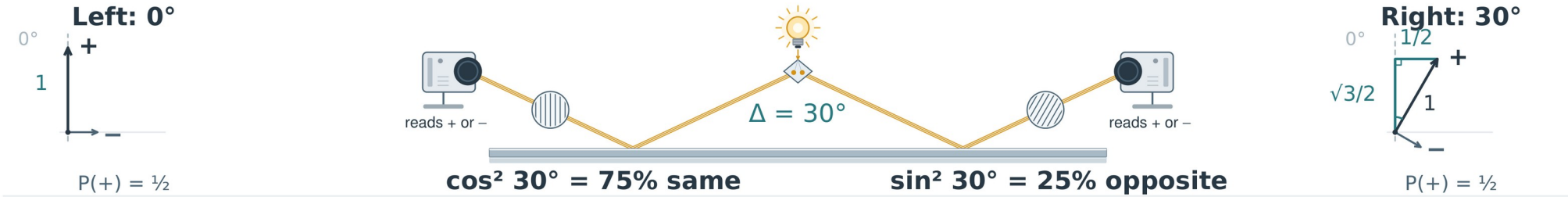
Same angle: 100% match.
50%: +++ or −−−
25%: ++− or −−+
25%: +−− or −++



Each “or” is 50/50. Unlimited memory still obeys this bound.

3. Entangled photons

Each detector is still 50/50. Equal settings always produce matching signs.



Bell’s inequality

$$D(0°, 60°) \leq D(0°, 30°) + D(30°, 60°)$$

Perfect same-angle agreement. Independent settings. No communication during measurement.

D = probability of opposite signs

Same angle: 100% match.
25% + 25% = 50% local maximum.

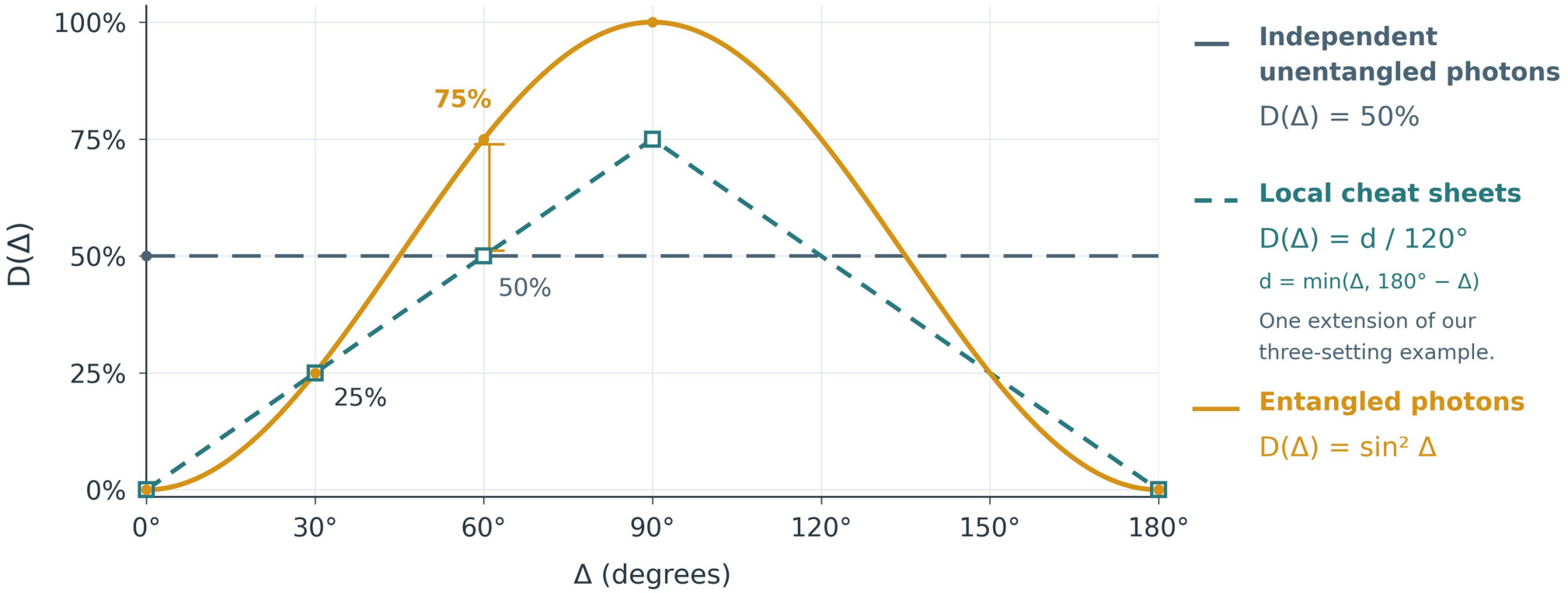
75% > 50%

Bell’s inequality is violated.



D(Δ): probability of opposite signs

Δ = |left polarization basis angle − right polarization basis angle|



Bell's inequality, when D(0°) = 0: D(60°) ≤ D(30°) + D(30°)

Entangled: 75% > 25% + 25% = 50%

Independent settings. No communication during measurement.

Cheat sheets must obey a triangle inequality

The source chooses the row and the shared coin flip before the photons separate.

Heads (½): M

0°	30°	60°	90°
+	+	+	+
+	−	−	−
+	+	−	−
+	+	+	−

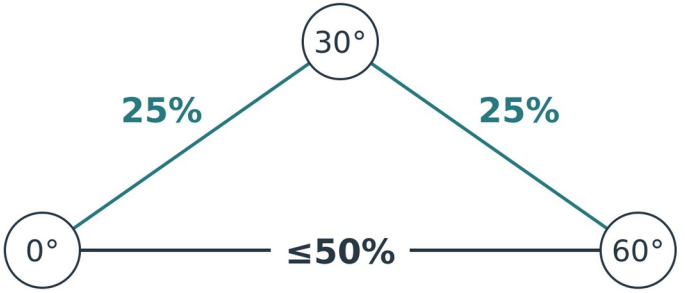
Tails (½): −M

0°	30°	60°	90°
−	−	−	−
−	+	+	+
−	−	+	+
−	−	−	+

Each row: ¼. Both photons carry identical copies.
Each detector then reads its local angle’s entry.
The shared flip keeps each local sign 50/50.

D is a Hamming distance

Fraction of rows with different signs



If the end signs differ,
at least one step must differ.

True for every row. Therefore true for every mixture of rows.

Bell’s inequality: D(0°, 60°) ≤ D(0°, 30°) + D(30°, 60°)

Local: 25% + 25% = 50%

Entangled: 75% > 50%

Perfect same-setting agreement. Independent settings. No communication during measurement.

```
left_sign = f(
    left_polarization_basis_angle,
    cheat_sheet.state_description,
    cheat_sheet.shared_randomness
)

right_sign = g(
    right_polarization_basis_angle,
    cheat_sheet.state_description,
    cheat_sheet.shared_randomness
)
```

Neither cheat sheet can access the other detector’s setting.



Quantum networks will allow some kinds of coordination that aren’t possible in classical networks, breaking some of the assumptions of some of the limitations we’ve been talking about.

(But faster than light communication is still not allowed.)

Netflix’s Ray using gRPC/Protobuf and message passing (not SOAP, but closer to SOAP than to REST)...

What should Sam watch next?

Hacker and pirate-radio movies, recommended across machines

Sam’s viewing history

WarGames

Sneakers

Finished both movies
Gave both a thumbs-up

recommend

Recommended for Sam

01 **Pump Up the Volume**

02 **Hackers**

03 **The Net**

The user sees an ordered row of movie recommendations.

Inspired by Netflix’s published recommendation and Ray training work.

Four actors, four machines

An actor keeps state and receives remote method calls

MACHINE A	MACHINE B	MACHINE C	MACHINE D
Controller actor process	**History** actor process	**Page Builder** actor process	**Ranker** actor process
Knows the other actors Starts the request	WarGames + Sneakers Both watched and liked	Builds the visible row Receives a ranker handle	Model + movie catalog Scores titles for Sam

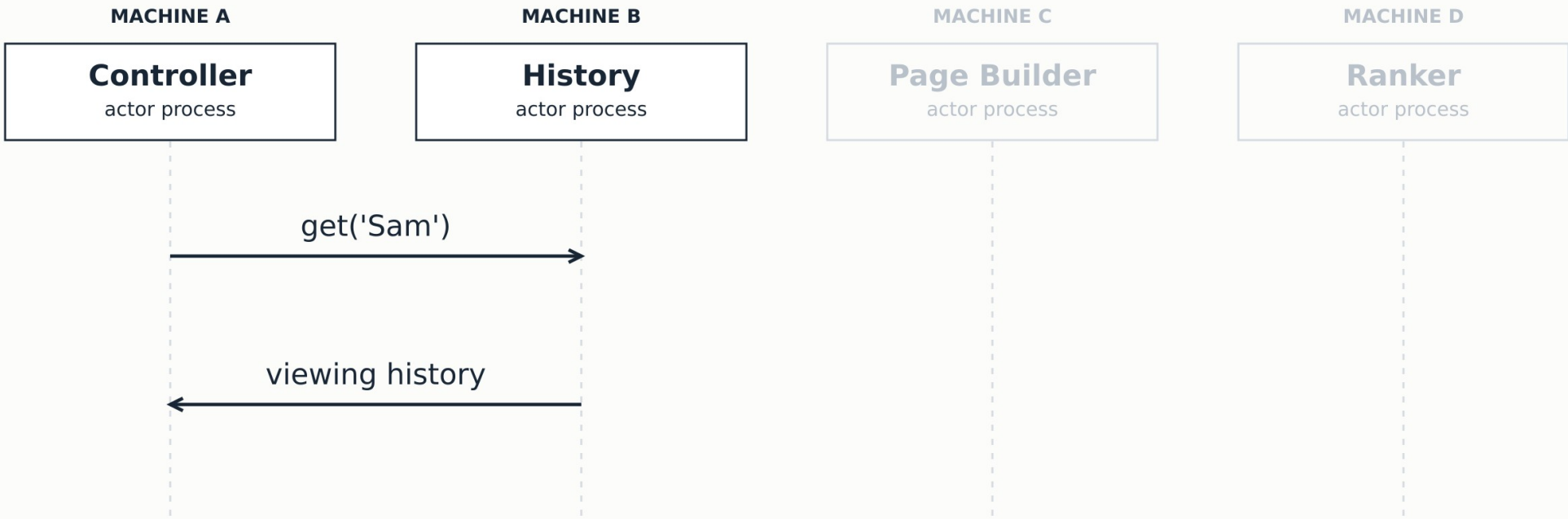
The Controller already holds handles to the other three actors.

A handle is a usable reference to an existing actor.

The actor stays on its machine when another process receives its handle.

1. Fetch Sam’s viewing history

The Controller asks the History actor on machine B

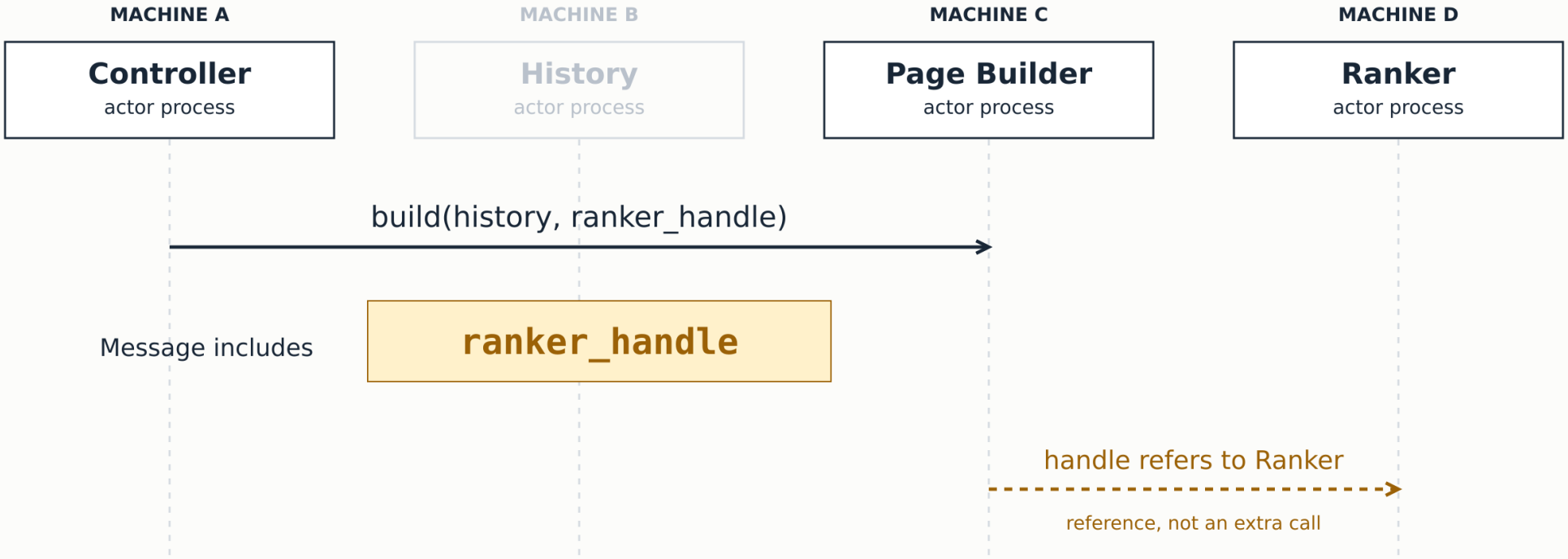


Reply: WarGames + Sneakers, both watched and liked

This message carries data about Sam.

2. Pass a handle to the Ranker

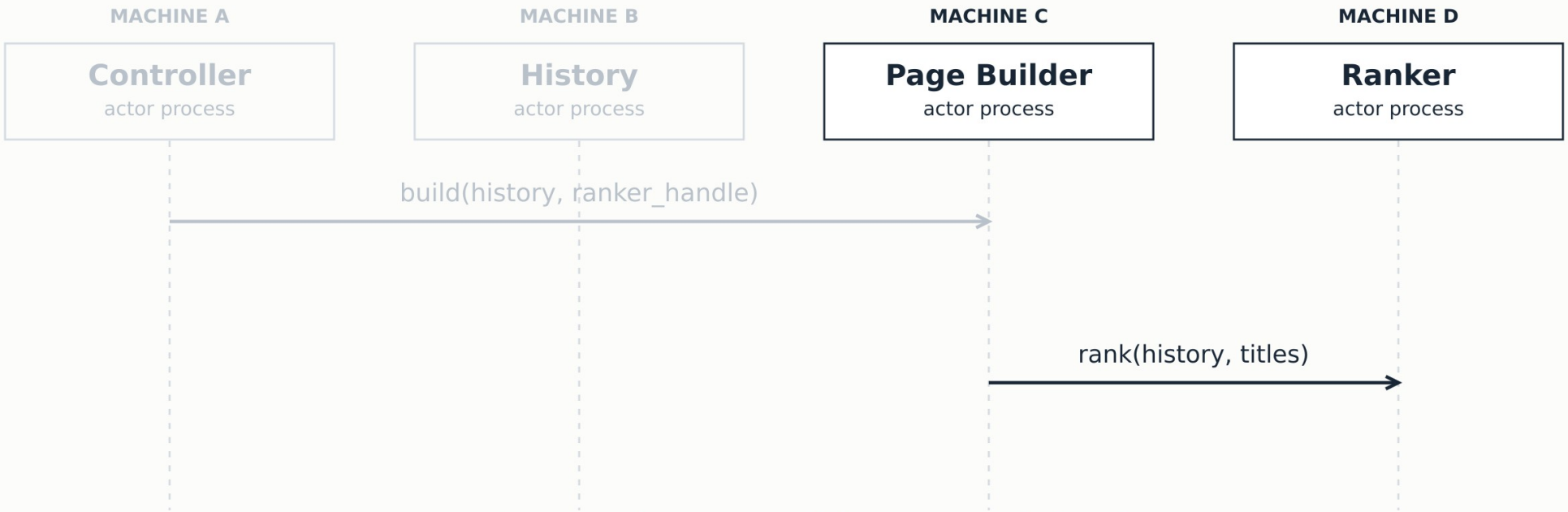
The Controller sends Page Builder both history data and an actor handle



Page Builder now has a way to call the actor on machine D.

3. Page Builder calls the Ranker

The received handle lets machine C contact machine D directly




```
ranker.rank.remote(history, titles)
```

The call returns an ObjectRef immediately, before the result is ready.

4. The Ranker scores the movies

A larger score puts a title earlier in Sam’s recommendation row

Input

WarGames + Sneakers

Both watched and liked

Ranker on machine D

Trained model + catalog



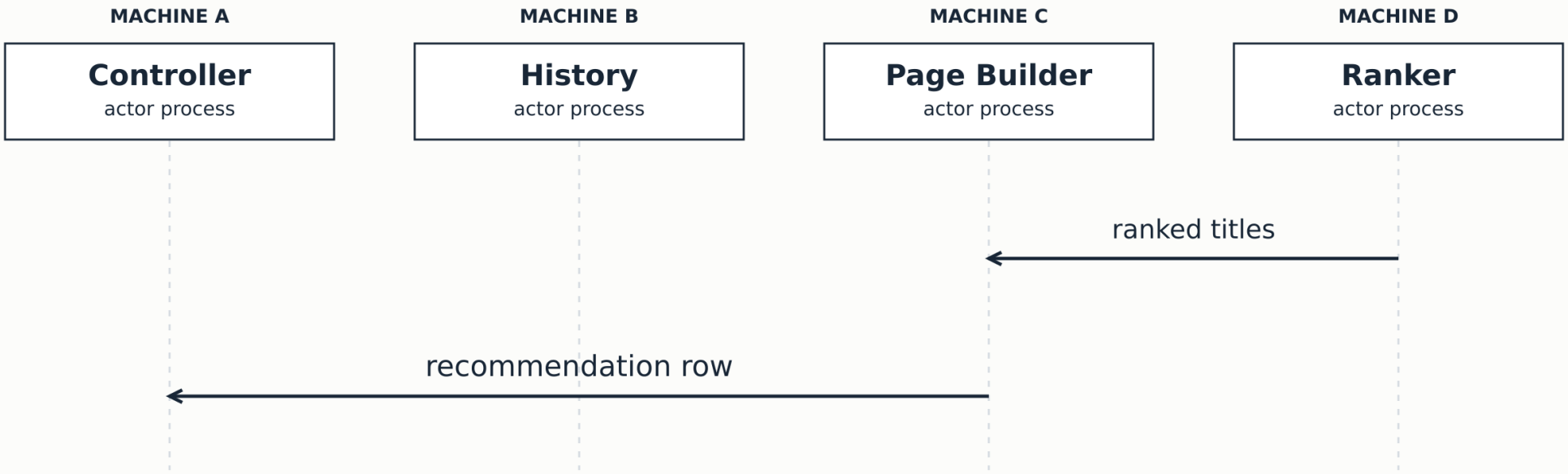
Candidate title	Score
Pump Up the Volume	**3.2**
Hackers	2.7
The Net	0.8

Illustrative scores, not probabilities

A real recommendation model learns from past viewing and feedback.

5. The page returns to Sam

Ranker returns the order, and Page Builder assembles the movie row



Because you liked WarGames and Sneakers

1 **Pump Up the Volume**

2 **Hackers**

3 **The Net**

A message can carry a way to communicate

The connection to Unix sockets, D-Bus and Milner’s π-calculus

A passed socket FD	A passed Ray actor handle
Refers to a local kernel object	Refers to an actor in the cluster
Kernel installs a usable FD	Ray reconstructs a usable handle
Receiver can read() / write()	Receiver can call actor methods
Works between processes on one host	Works between different hosts

The receiver gains a communication endpoint through a message.

The Ranker stays on machine D. A Unix FD remains local to its host.

Sources: Ray docs, Actors / Objects. Netflix TechBlog, GenRec / LLM Post-Training.

A Microsoft Copilot-style assistant using Azure OpenAI, LangChain, a REST API on Azure App Service behind Azure Front Door, and Azure Cosmos DB for NoSQL with active-active U.S.–Europe regions and asynchronous geo-replication illustrates PACELC and distinguishes causal consistency from globally fresh chat history...

Copilot-style means a conversational AI assistant like Microsoft Copilot that uses a language model to answer questions and help with tasks.

LangChain is a software framework for building applications that connect language models to tools, data, and conversation history.

Azure App Service hosts web applications and APIs while Microsoft manages the underlying servers and operating systems.

Azure Front Door is a global web gateway that routes requests to healthy application instances and can cache responses.

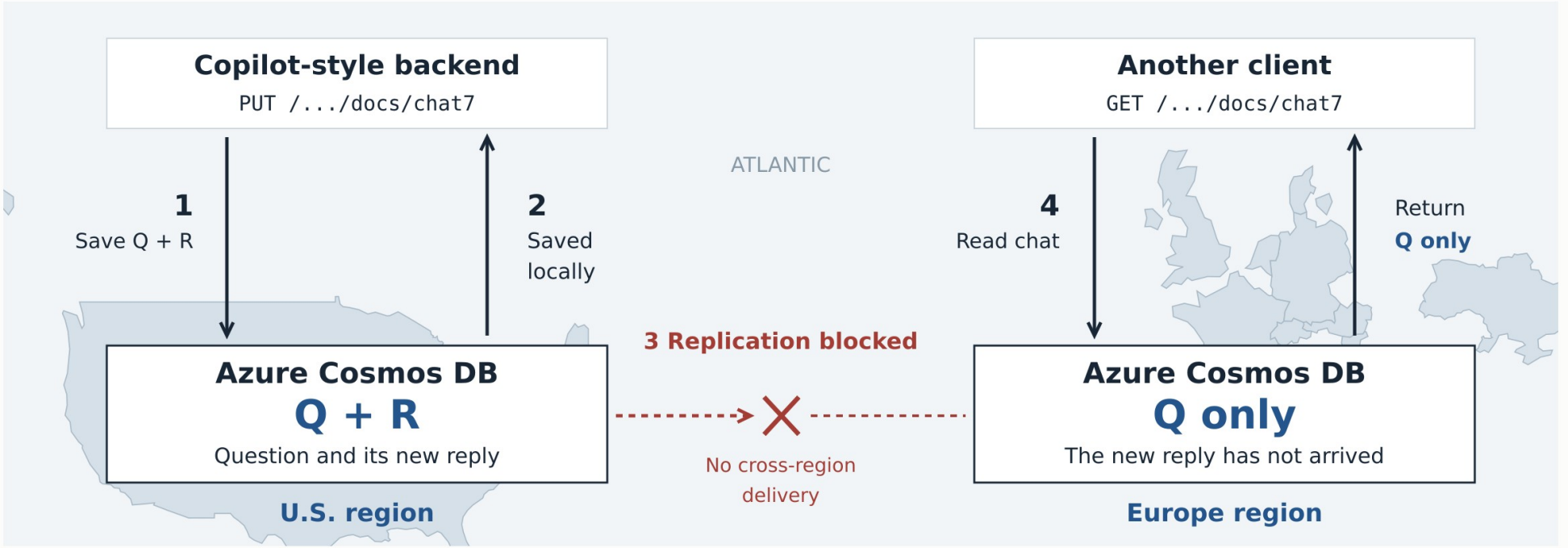
Cosmos DB is Microsoft’s managed distributed database service, which can store JSON documents and replicate them across geographic regions.

NoSQL is a family of nonrelational databases, including document, key-value, and graph databases, some of which support SQL-like queries.

P: the U.S.–Europe link is down

A over global C

Cosmos DB multi-region writes with eventual reads



Q is a question. R is the reply that depends on Q.

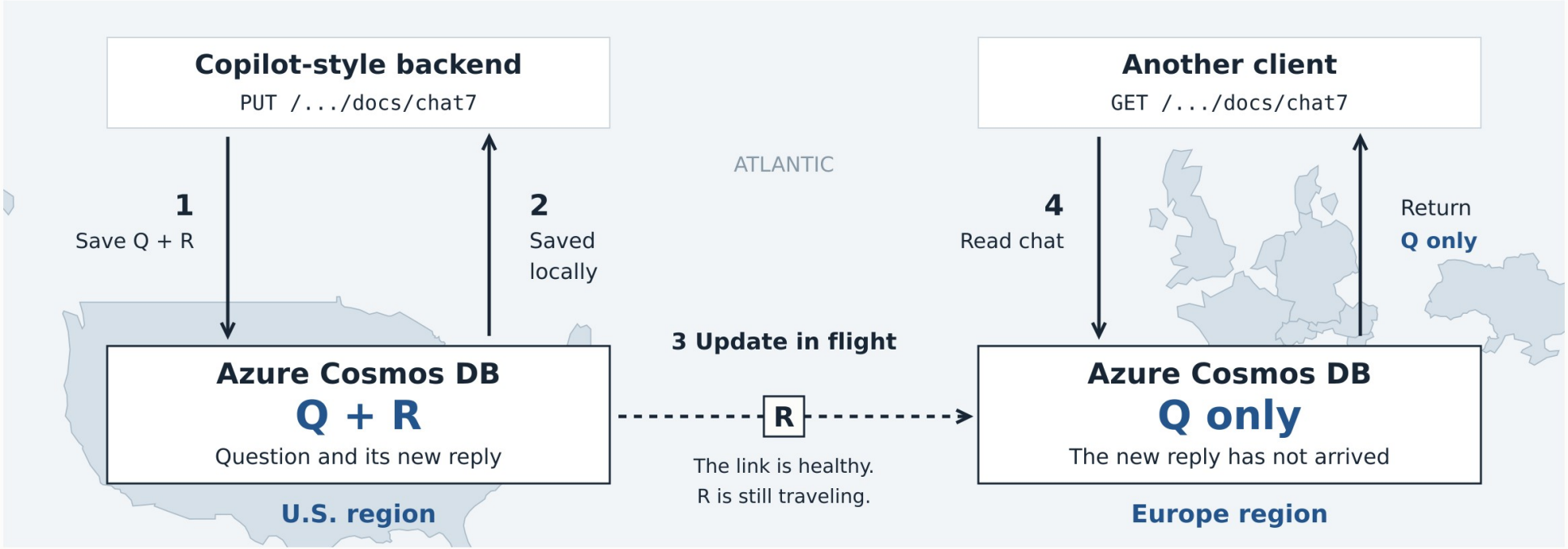
Causal consistency would preserve Q before R and still allow this older Q-only copy.

Europe keeps serving the older chat until the connection returns and replication catches up.

E: the link works, but messages take time

L over global C

Cosmos DB multi-region writes with eventual reads



Q is a question. R is the reply that depends on Q.

Causal consistency would preserve Q before R and still allow this older Q-only copy.

The user gets a quick save, but Europe can show the older chat until the update arrives.