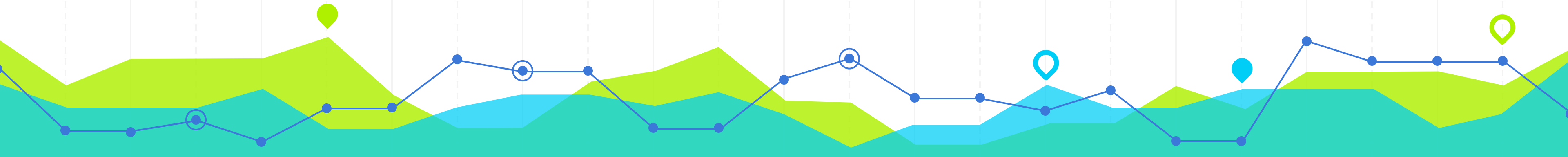
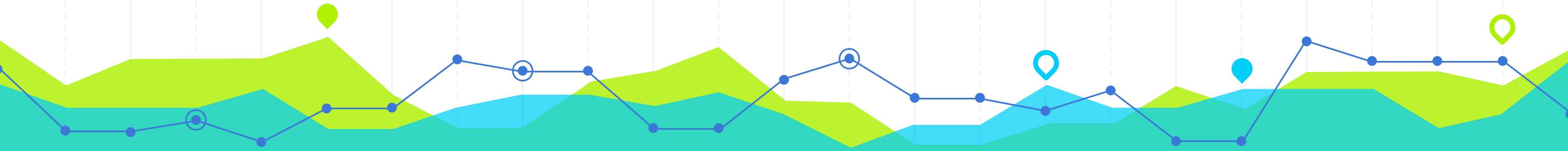


FUNDAMENTALS OF STATE REPLICATION



Definitions and Principles

First things first

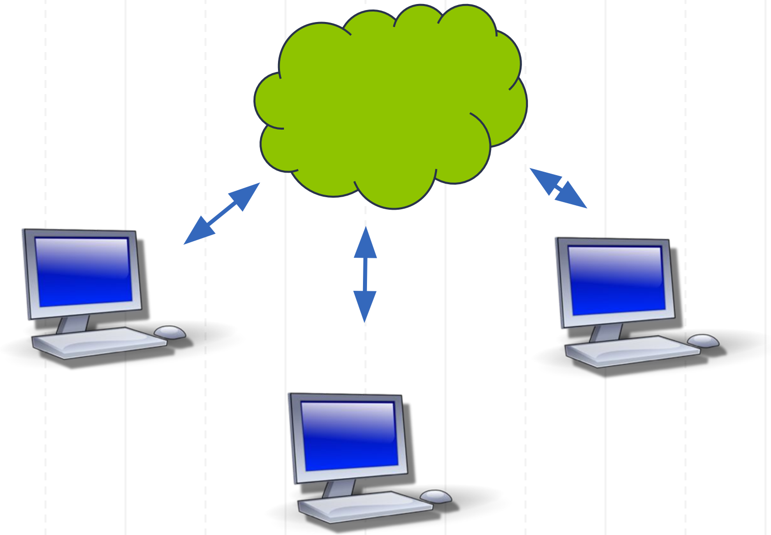
1

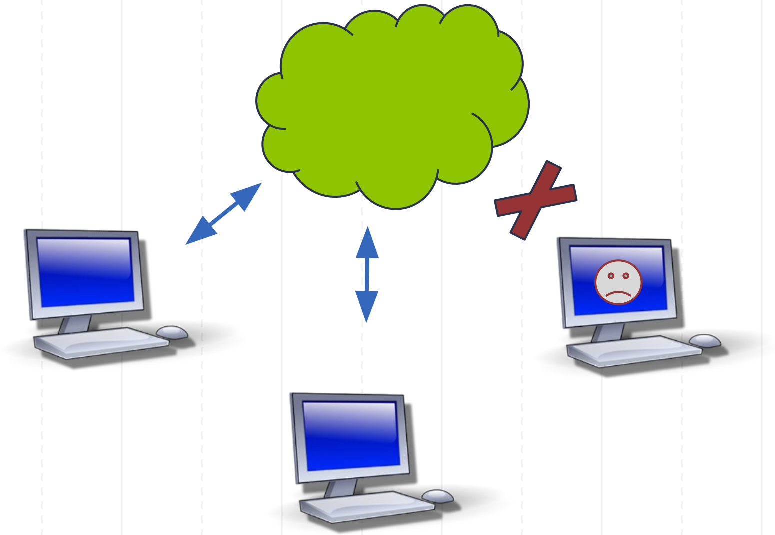
> *A* ***distributed system*** *is one in which two or more processes work together to perform a task.*

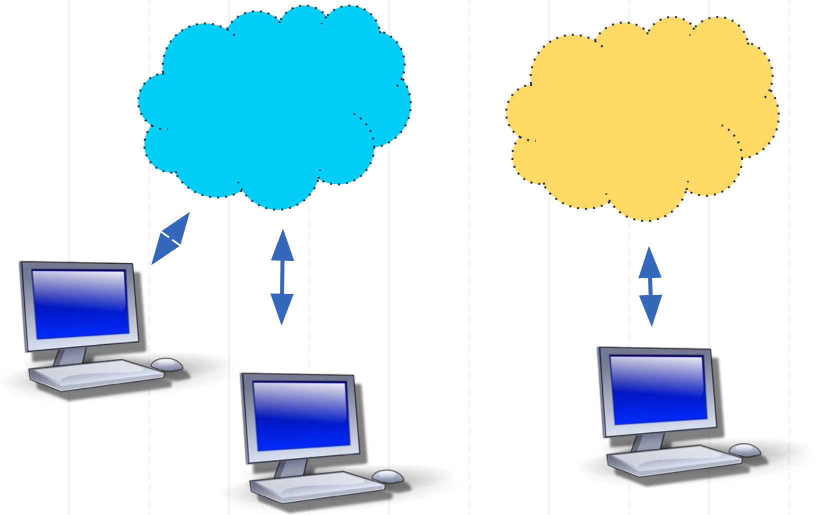
> *The total information processed by the system is known as its* ***state***.

“

Ensuring that all processes in the system have the same view of the state is known as
replication
or (less precisely)
synchronization







TWO CONFLICTING GOALS

Consistency
A distributed system is **consistent** when all its processes agree on its state.

Availability
A distributed system is **available** when all its processes are allowed to modify the state.

WE CAN'T HAVE BOTH.
ALL WE CAN DO IS PICK WHERE WE FALL BETWEEN THEM:

C - - - - - - - - - - - - ? - - - - - - ? -

- - - - - - - - ? - - - - - - - - - - A

TWO COMPETING CHALLENGES

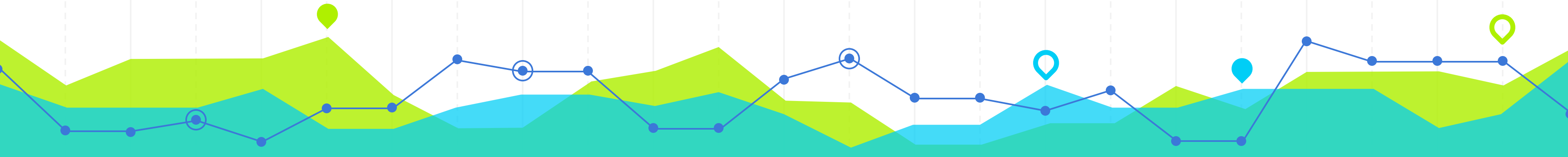
Uncertainty
When a distributed system is **consistent**, we have eliminated its uncertainty.

Time
When a distributed system is **available**, it is functioning as if time were not a factor.

WE CAN'T AVOID BOTH.
ALL WE CAN DO IS PICK WHERE WE FALL BETWEEN THEM:

C - - - - - - - - - - - - ? - - - - - - ? -

- - - - - - - - ? - - - - - - - - - - A

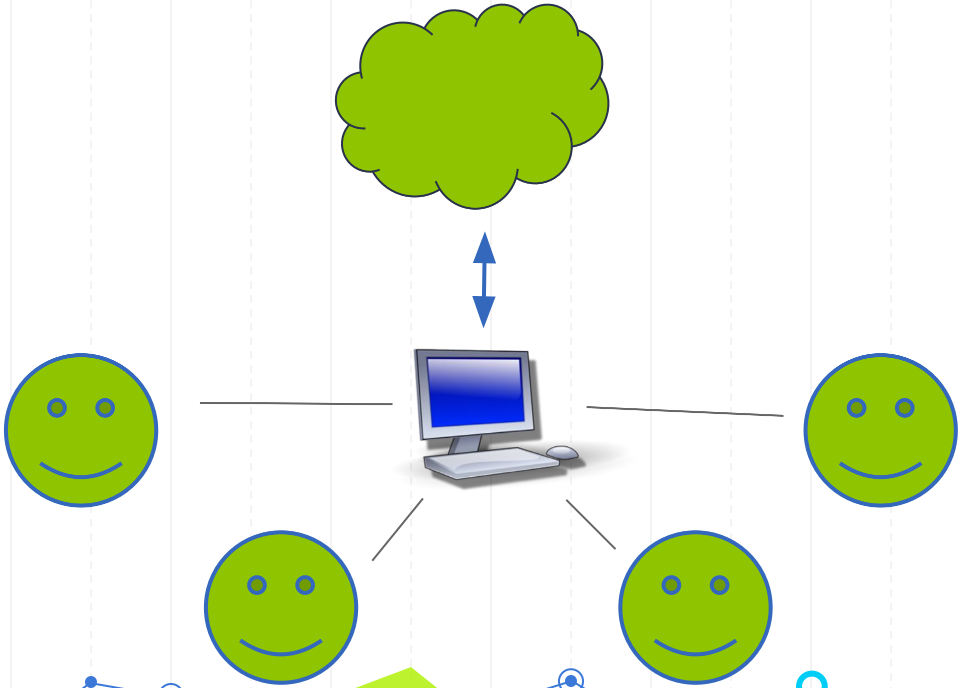


2

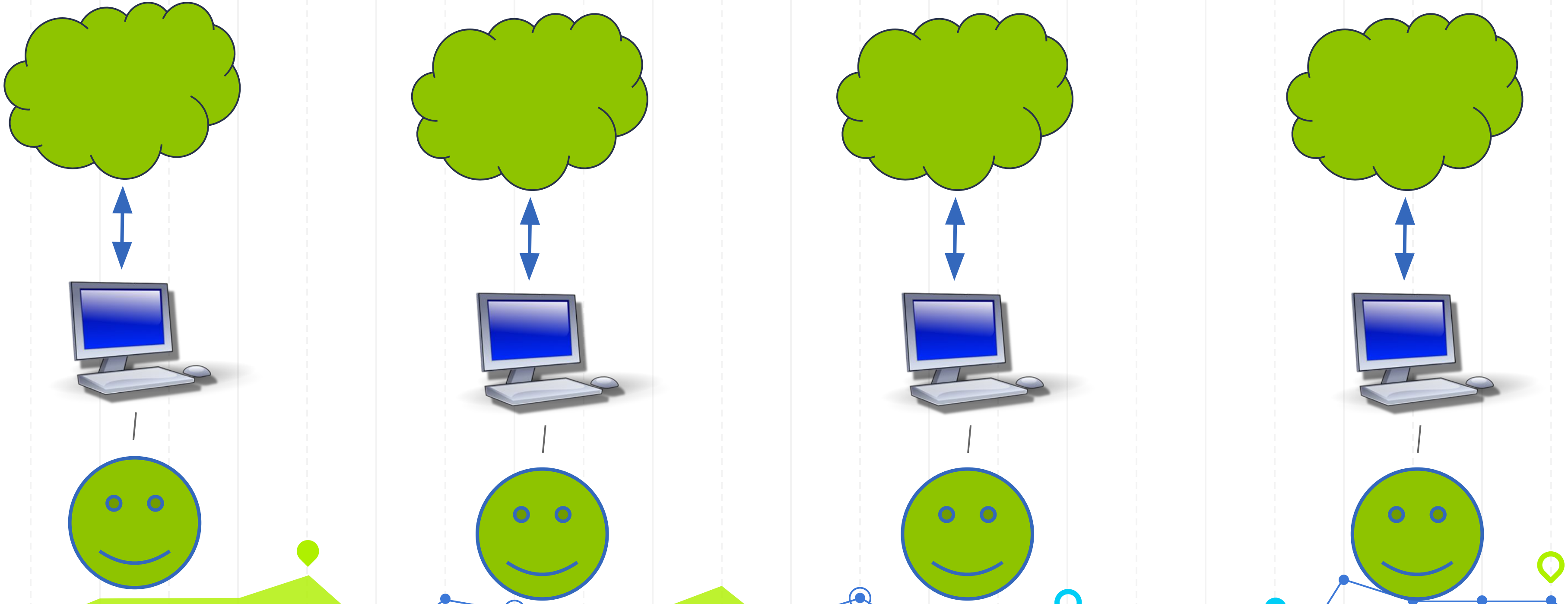
Standard Approaches

Don’t reinvent the wheel

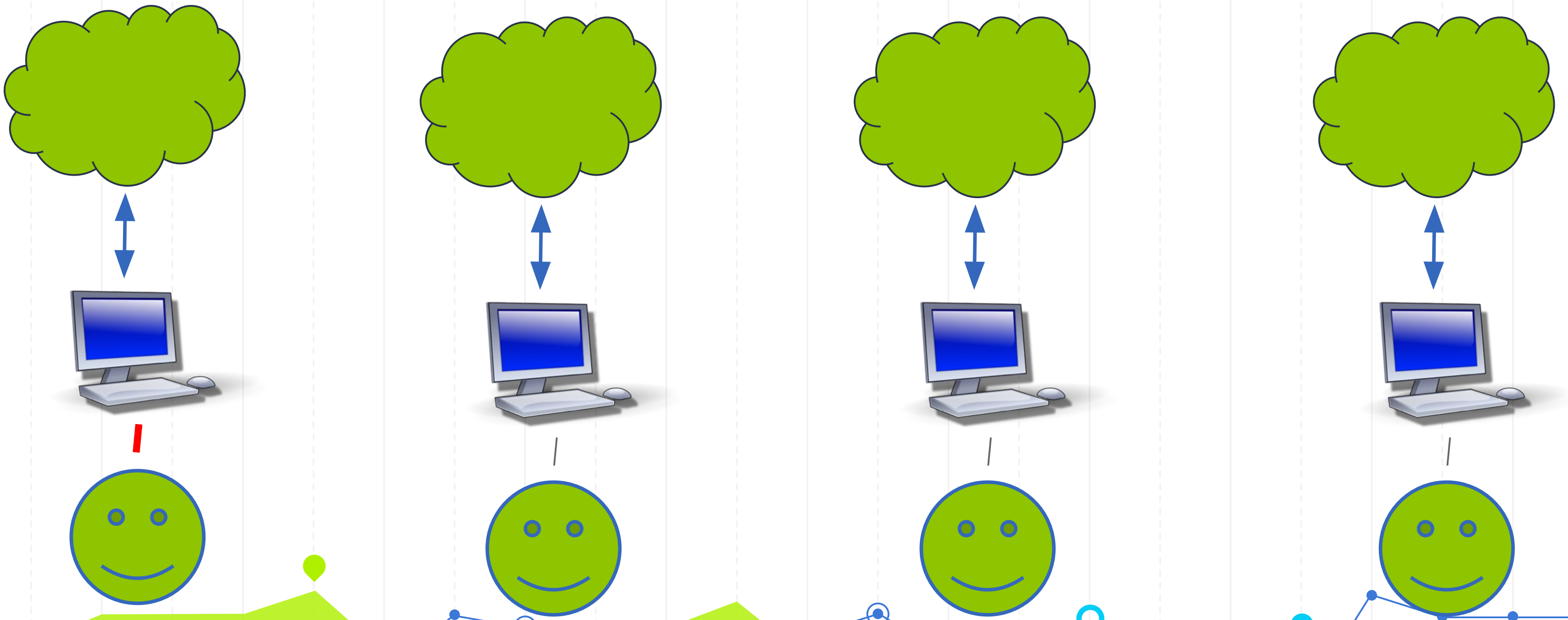
Local Multiplayer



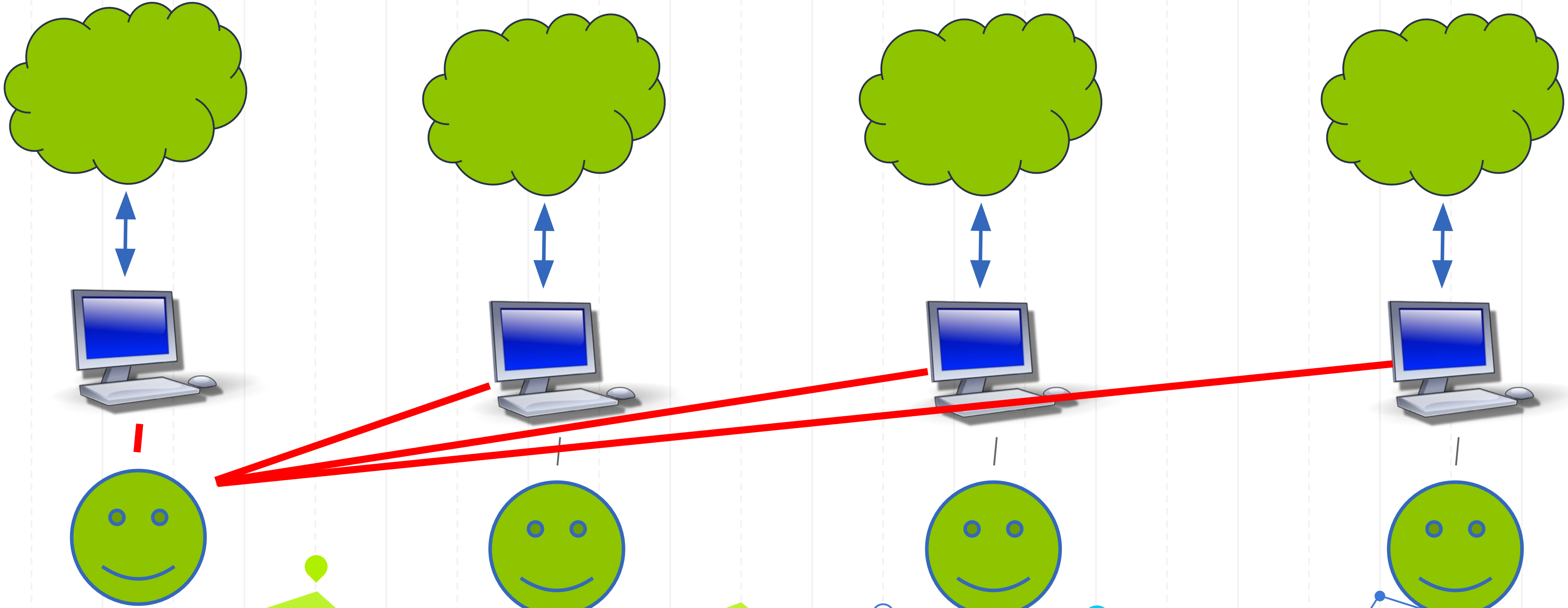
Lockstep Synchrony



Lockstep Synchrony



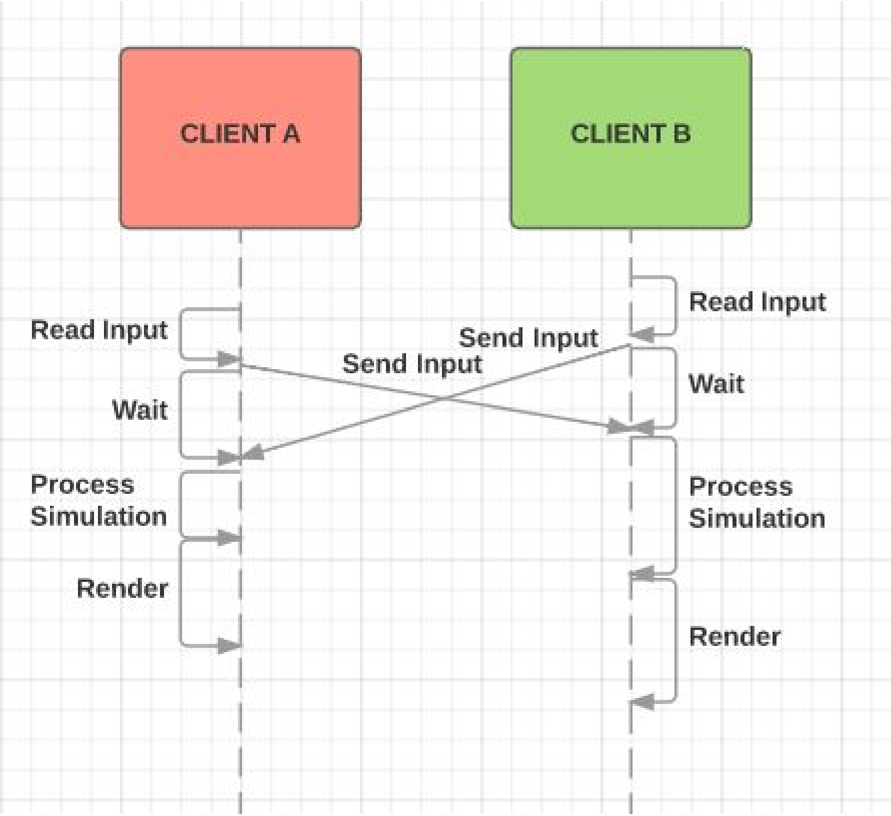
Lockstep Synchrony



Gotchas

- Simulation must be completely deterministic… so be careful with the random number generator
- Object references can’t use pointers

THIS IS GOOD ENOUGH*



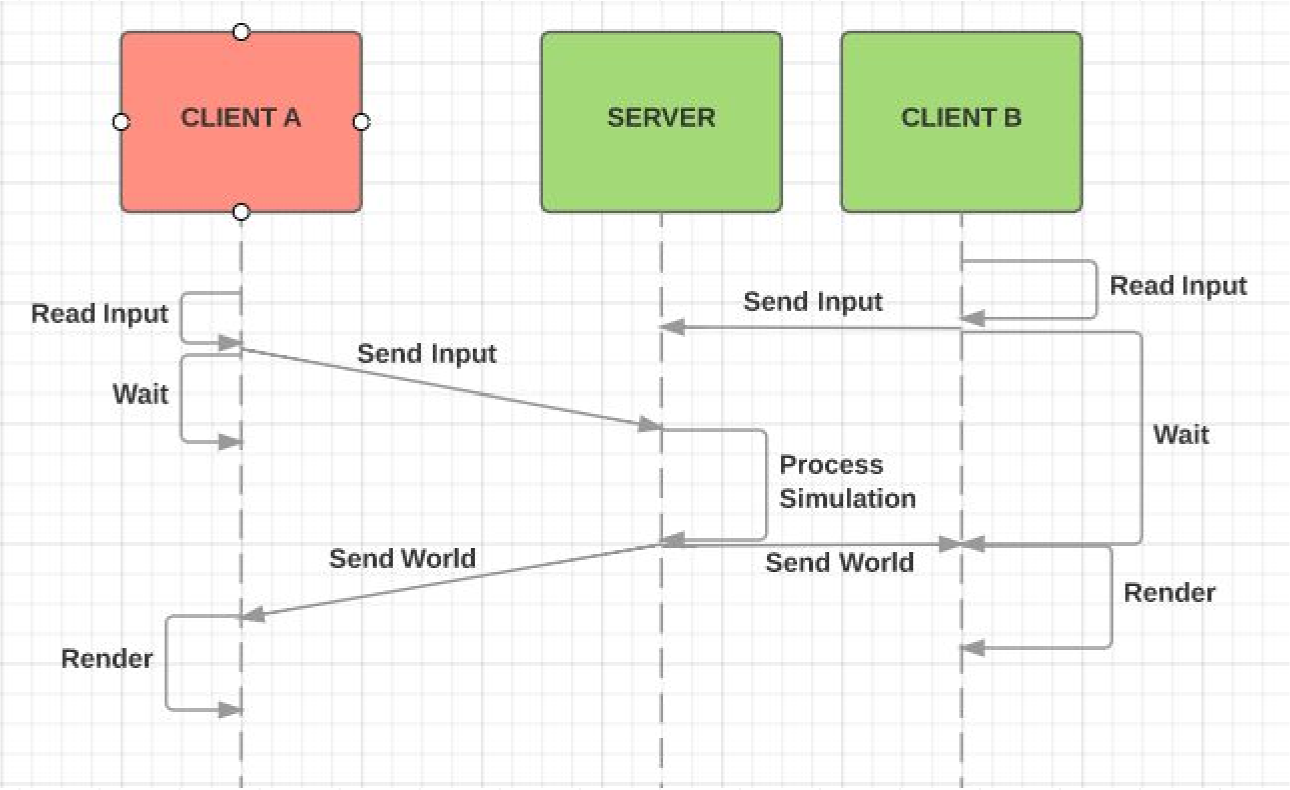
Lockstep Synchrony Sequence Diagram

Lag

- Direct control tasks: Noticed at 50ms, annoying at 100ms, fatal at 350ms+
- Indirect control tasks: Noticed at 250ms, annoying at 1,000ms, fatal at 5,000ms+
- “Ping time” is round trip; one way should *generally* be half that

Lockstep Synchrony

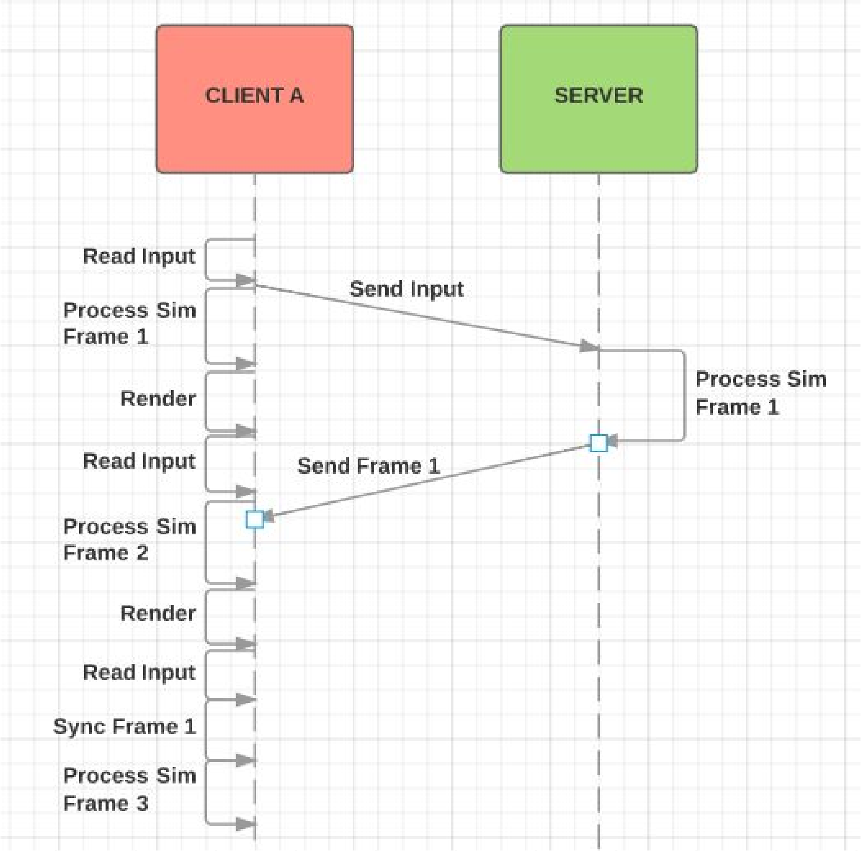
- Lag is one-way trip time (or roughly ½ ping)
- Clients are coupled; sim runs at speed of slowest client
- Rejoining difficult to impossible
- Consistency > Availability when operating normally
- Availability > Consistency when partitioned



Dumb Client Synchrony Sequence Diagram

Dumb Client Synchrony

- Lag is full ping time
- Clients are decoupled…
- ...but this is potentially unfair
- Rejoining easy
- Consistency > Availability when operating normally
- Complete unavailable when partitioned



Optimistic Synchrony Sequence Diagram

Optimistic Synchrony

- Effectively no lag on the player’s action
- But the player is always acting in the future
- Availability > Consistency in normal operation…
- …in fact *never* consistent

Failure cases:

- Sometimes the player’s action is rejected
- The player’s always aiming behind targets

Optimistic Synchrony: Player Snap-Back

1. Remember player's sequence of moves
2. Snap back to divergent frame
3. Replay sequence of moves from now-corrected state
4. Display this to the player in some suitable manner

Optimistic Synchrony: Target Prediction

1. Track momentum for all remote entities
2. Project remote entities forward in time from last-known frame to current frame
3. When errors arise, they will be in the past. Rewind, integrate correct state, and project forward to the present again
4. Display this to the player in some suitable manner
5. For perceived fairness, server must evaluate critical actions from the known historical perspective of the player

Where’s the simulation?

Just on the server:

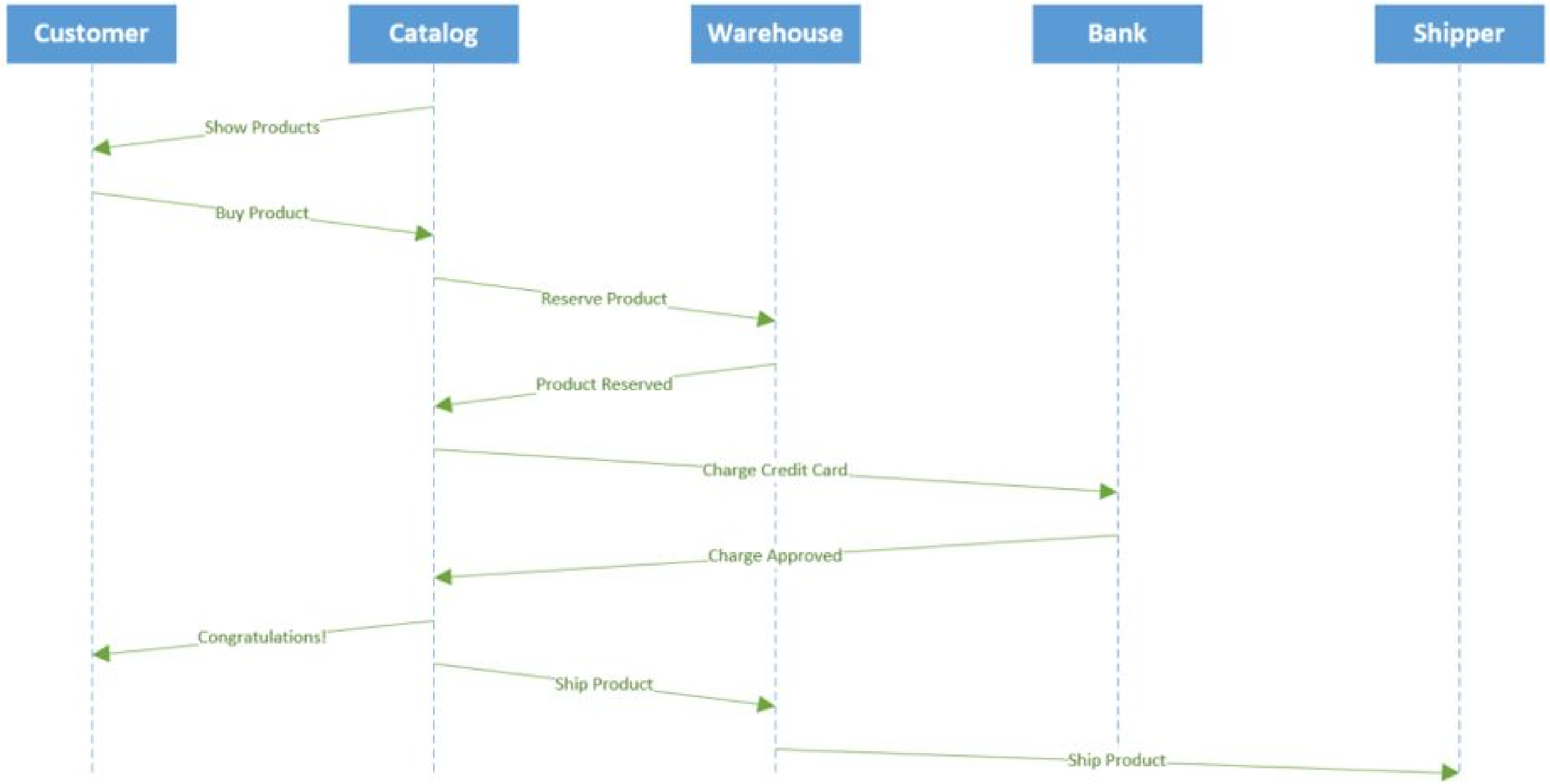
- Simpler to implement
- Highly secure
- Lag on every action
- Very dependent on simulation frame rate

Call of Duty and World of Warcraft work this way

On both client and server:

- Player’s actions are lag-free
- Runs well at lower frame rates
- Physics interactions between controlled objects (e.g. players) still subject to full lag

HALO, Destiny and Assassin’s Creed work this way



Customer

Catalog

Warehouse

Bank

Shipper

Show Products

Buy Product

Reserve Product

Product Reserved

Charge Credit Card

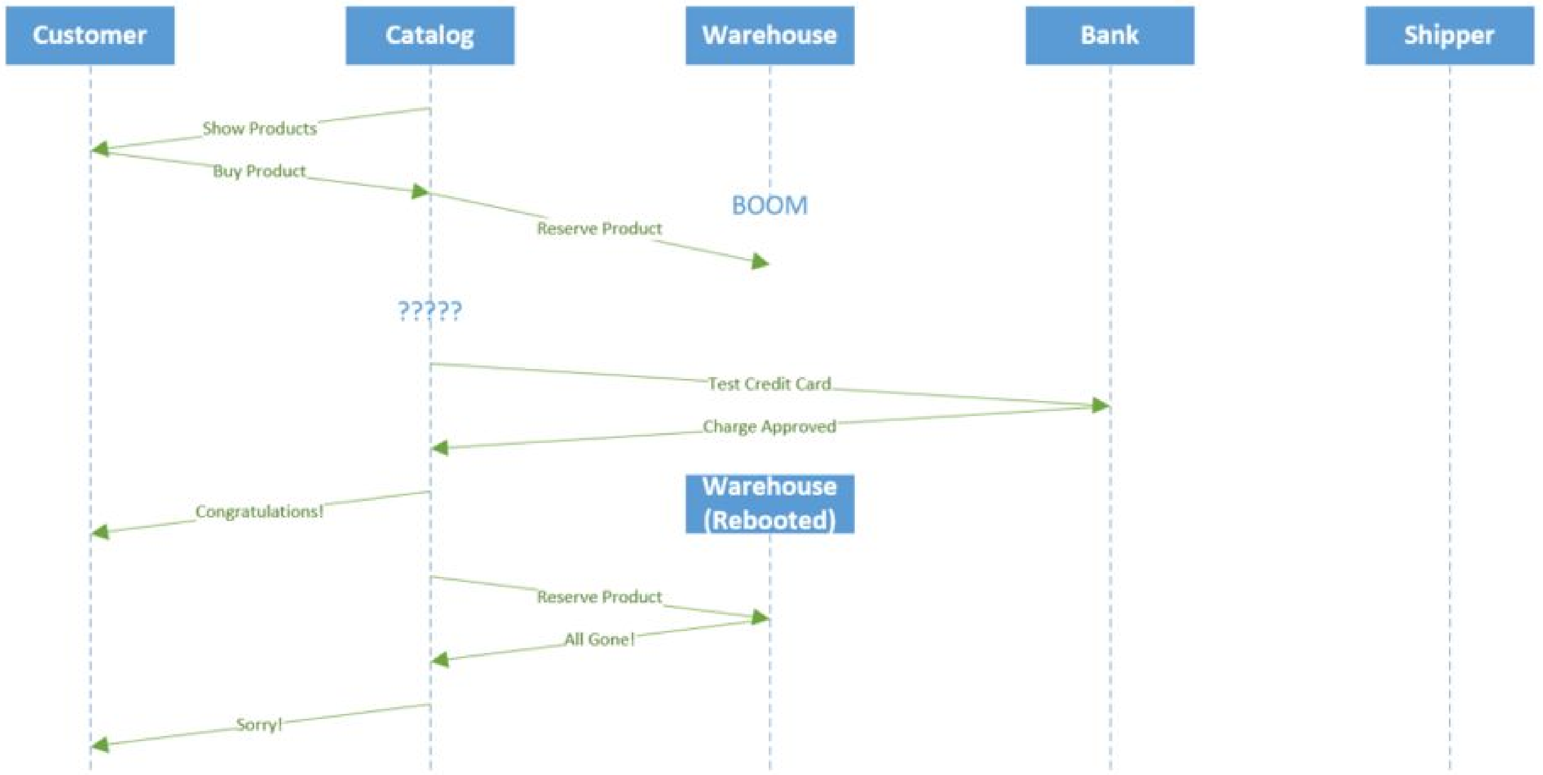
Charge Approved

Congratulations!

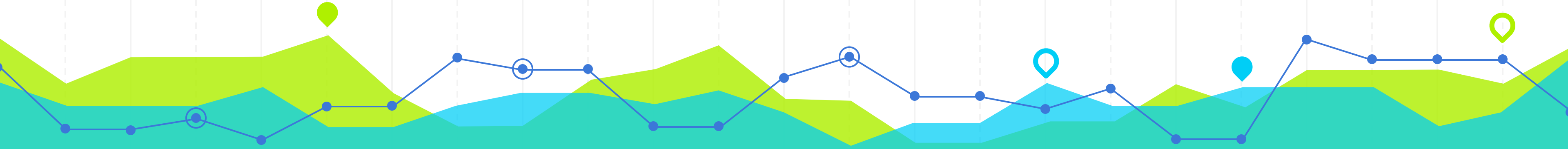
Ship Product

Ship Product

Eventual-Correctness Synchrony Sequence Diagram



Eventual-Correctness Synchrony Sequence Diagram



Reality Bites

If this is so easy, why doesn’t everybody do it?

3

What Can Possibly Go Wrong

- Lag
- Jitter
- Packet loss
- Limited bandwidth

Lag

Lag interferes with the player's sense of control.

It comes from:

- Processing delay (number of hops)
- Queuing delay (congestion)
- Transmission delay (bandwidth)
- Propagation delay (distance)

Handle using the previously-mentioned techniques.

Jitter

Jitter is variation in lag. It disrupts our lag-tolerance schemes and makes the simulation choppy.

It comes from:

- Variation in routing (congestion and outages)
- Queuing (congestion)
- Packet loss

Handle by deepening lag compensation; only restore “normal” levels gradually.

Packet Loss

Packet loss reduces bandwidth and introduces jitter.

It comes from:

- Hardware faults
- Analog noise and radio interference
- Congestion

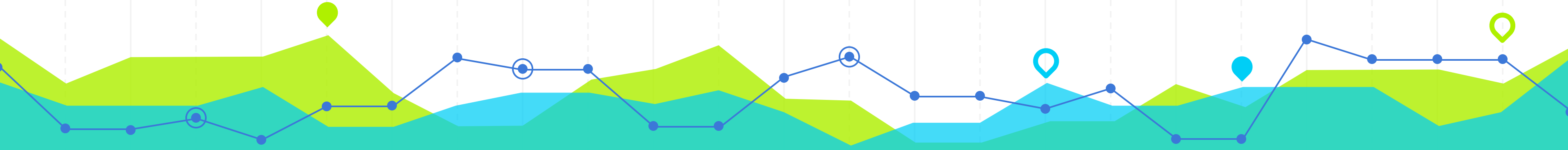
TCP handles packet loss automatically (if poorly); for UDP we have to develop special schemes.

Limited Bandwidth

Limited bandwidth effectively introduces lag.
It comes from:

- Poor infrastructure investment
- Lack of competition for “last mile” service
- Bad router design decisions

Handle it through compression or delaying less-important updates. A good rule of thumb is to fit within 8KB/sec.



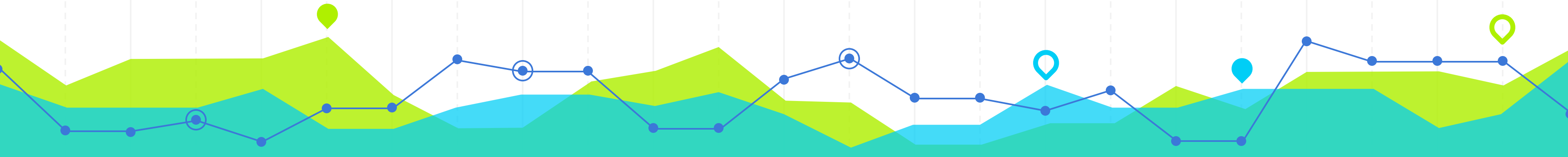
Easy Mode

Quick and dirty networking for GAM projects

4

Easy Mode

1. Use “dumb client” networking
2. Use TCP
3. Don’t run on wifi
4. If needed for your game, project simulation forward one frame while waiting for the next frame to arrive. Patch over errors using simple linear interpolation.



5 Summary

And in conclusion...

Summary

- A distributed system is one in which state is modified by more than one process…
- ...which means processes will inevitably disagree on what that state actually is.

- Either halt until disagreement is resolved (consistency over availability)...
- ...or keep on working and cope with the disagreement (availability over consistency)

Networking Models

Lockstep
All clients send input to each other, then advance simulation one frame together.

Dumb Client
All clients send input to server, which advances simulation one frame itself and sends results back to clients.

Optimistic
All clients send input to server *and* advance the simulation locally, patching up discrepancies as they arise.

CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash

PRESENTATION DESIGN

This presentation uses the following typographies and colors:

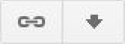
- Titles: **Oswald**
- Body copy: **Source Sans Pro**

You can download the fonts on this page:

https://www.google.com/fonts#UsePlace:use/Collection:Source+Sans+Pro:400,700|Oswald:400,700

Click on the “arrow button” that appears on the top right

Sky blue **#00cef6** / Bright green **#aff000** / Blue **#3c78d8** / Dark blue **#28324a**



You don’t need to keep this slide in your presentation. It’s only here to serve you as a design guide if you need to create new slides or download the fonts to edit the presentation in PowerPoint®

SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change fill color and opacity.
- Change line color, width and style.

Isn’t that nice? :)

Examples: