Distributed Systems and the Web

Karl Kirch

@joekarl

SIMPLE

What is a distributed system

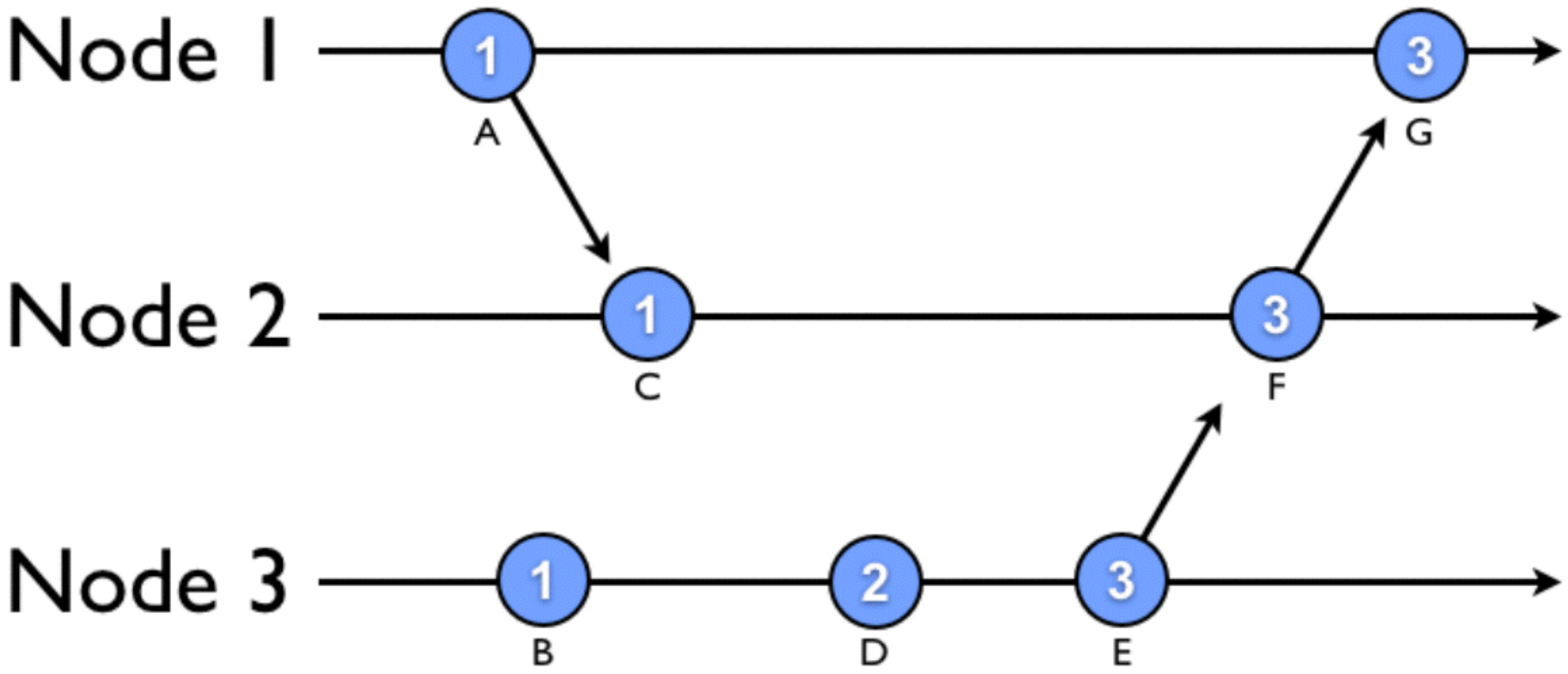
> *“A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages.”*

- **Leslie Lamport - Time, Clocks, and the Ordering of Events in a Distributed System (1978)**

Examples of nodes

- Web server
- A browser
- Instance of a mobile application
- IOT device
- A message broker
- Datastore
- Another distributed system

Nodes in the System



Example of communication between nodes

- Browser makes request to server (via page load/ajax)
- Server makes request to database (SQL call)
- Server makes request to another server (HTTP request)
- Datastore replication (clustering)
- Messaging (rabbitmq, kafka, etc...)

“A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable”

- Leslie Lamport

Why even distribute a system?

- Scale due to performance
- Redundancy
- Scale due to data size
- Availability
- Physical distance
- Microservices*

➡Take advantage of the internet

8 Fallacies of Distributed Computing

1-7 Peter Deutsch (1994)

8 James Gosling (1997)

Fallacy #1

The network is reliable

- Traffic is reordered
- Traffic is redelivered
- Traffic is dropped or delayed
- TCP largely handles these issues, but only per connection

➡Use error handling callbacks for everything network related (Promise.catch, callback err)

➡Have a plan for unavailable resources (progressive enhancement)

Fallacy #2

Latency is zero

- Bounded by physics
- Specifically the speed of light
- 1 foot per nanosecond - (See Grace Hopper https://www.youtube.com/watch?v=JEpsKnWZrJ8)
- Beware treating remote calls like local calls
- Only solution is to move data closer to the consumer

➡Assume any remote call can and will be slow (AJAX, HTTP, SQL, Write to Disk, etc…)

Fallacy #3

Bandwidth is infinite

- Still an issue, but less of one than in the past
- Still tends to show up because of mobile networks
- Differs wildly based on geographic location
- If you have large amounts of data this is still a very real issue
- Amazon Snowmobile - Will truck your data from a datacenter to “the cloud”

➡Don’t load more data than you need (avoid unneeded JS/CSS, minify/pack your sources)

Fallacy #4

The network is secure

- Build with security in mind, don’t be complacent
- The internet is dangerous
- Spoofing, DDOS, MITM
- Watch Luke Crouch's Thunderplains talk for more related things - https://www.youtube.com/watch?v=0XDpJUhDTos

➡Use HTTPs, don’t send/store secure data in plain text

➡Don’t trust user input

Fallacy #5

Topology doesn't change

- Don’t depend on specific routes/locations
- DNS is a (decent) abstraction around IP locations
- Don’t assume quick paths will always stay quick
- Nodes on the network move

➡Have a plan for being offline

➡Be aware of DNS, how it works, it’s issues

Fallacy #6

There is one administrator

- You don’t control the network
- VPNs, Firewalls, etc…
- Especially prevalent for the web
- Can you fallback to different protocols if you cannot communicate

➡Provide fallbacks for things like websockets, web RTC

Fallacy #7

Transport cost is zero

- Cost as in Money
  - Highly available, low latency networks/systems can be built, but cost a lot money
- Cost as in Resources
  - Transporting data over the network consumes system resources

➡Microservices == cost

➡Avoid additional network calls (AJAX, CSS, JS)

Fallacy #8

The network is homogeneous

- Different parts of the network have different latency/ bandwidth
- Last leg tends to be slow
- Some network links are less reliable than others

➡Calls to different servers will have different behaviors

➡Have a plan for partial availability of web resources (what happens when your web font doesn’t load?)

Failure is always an option

Plan for failure

- If possible, retry
- Utilize Idempotency
- Respect back pressure
- Circuit breakers
- Assume duplicate delivery
- Enforce or avoid explicit ordering
- Use queueing to insulate yourself from failure
- Instrument your system

Day to day problems

Unreliable networks

- How do you handle slow networks?
- How do you communicate network problems to the user?
- How do you handle partial failure? Rollback?

Bad user behavior

- How do you handle things like double submit problems?
- What if a user reloads a POST’d page?
- How do you handle user impatience? Can you avoid this?

Concurrency

- How to handle multiple concurrent updates to the same resource?
- Can you enforce ordering? Should you?
- How to avoid dirty writes?

“The app is slow”

- What is slow?
- What does slow mean?
- Where is it slow?
- Can you insulate against the slowness?

Final thoughts

Distributed Systems are Challenging

Distributed Systems are Unavoidable

Distributed Systems are Necessary

Resources

- Contemporaries
  - Caitie McCaffrey
  - Camille Fournier
  - Christopher Meiklejohn
  - Ines Sombre
  - Kyle Kingsbury
- Papers
  - Distributed Systems reading list - https://dancres.github.io/Pages/

Questions?