Crayon Drawing

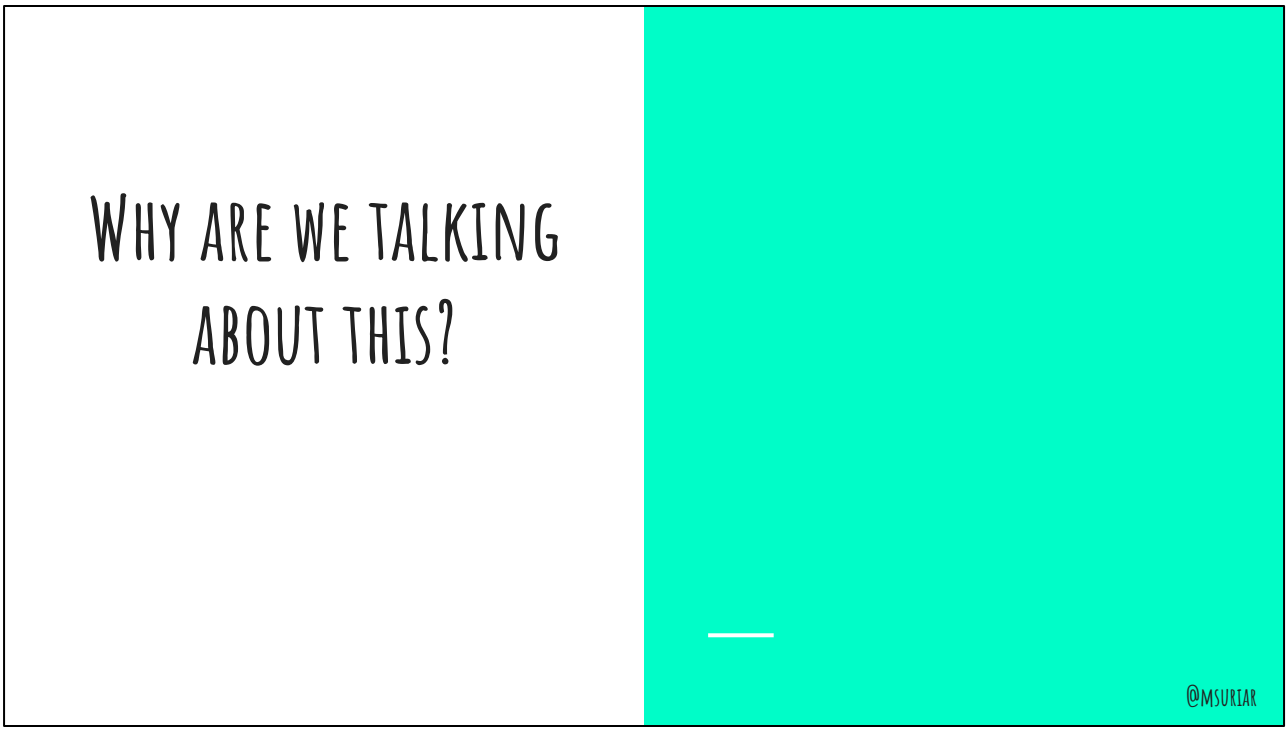
A Vital Engineering Skill

@msuriar

- How many people could draw a diagram of a system they're responsible for from memory?
- Of those: if someone else in your team drew the diagram, would they look the same?

Who am I?

- Currently an SRE at Snowflake
  - I am not speaking on behalf of Snowflake.
- Worked in infrastructure in a couple of other companies
- @msuriar on Twitter, Github.



- We run complex systems.
- Reasoning about complexity is hard.
- Documenting complexity is hard.

Because we run complex systems

Reasoning

- How do systems respond in steady state?
- How do systems respond in various degraded modes?

@msuriar

- Given some symptoms of problems, where should we look to further narrow things down?

Documenting

- How do you “best” document a system?
- How many high level components are there?
- How complicated are each of your subcomponents?
- What are the linkages between components… ?

How do we make all this tractable?

@msuriar

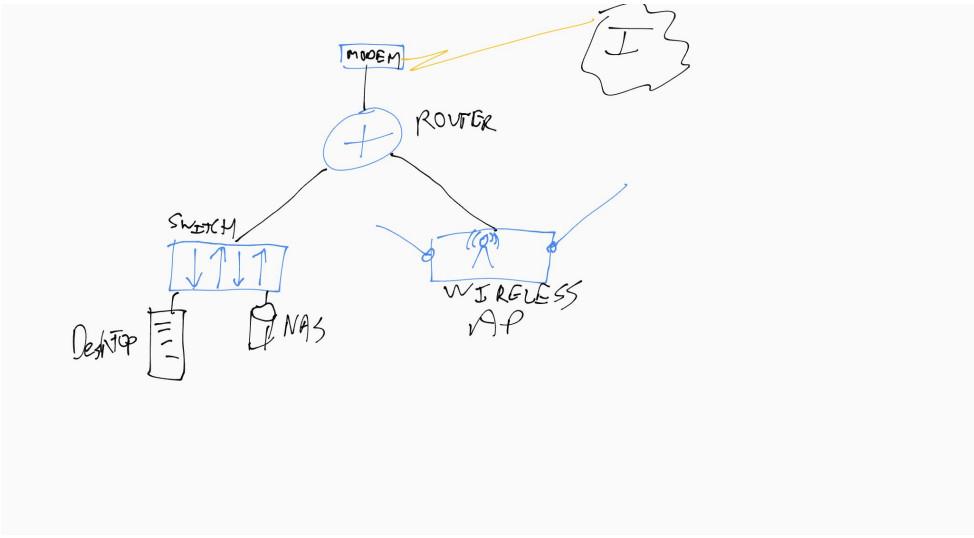
We build abstractions

@msuriar

My Home Network™

@msuriar

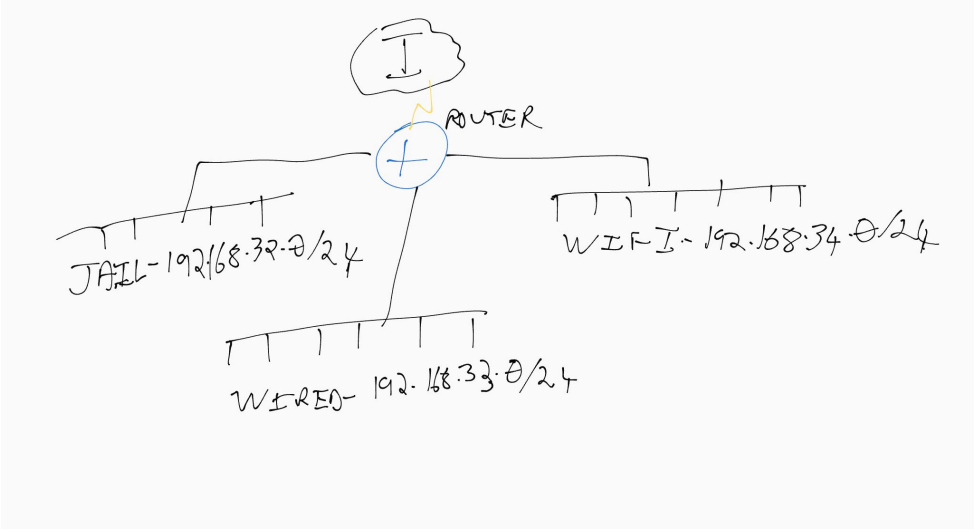
Physical



@msuriar

- Every physical device, and every cable

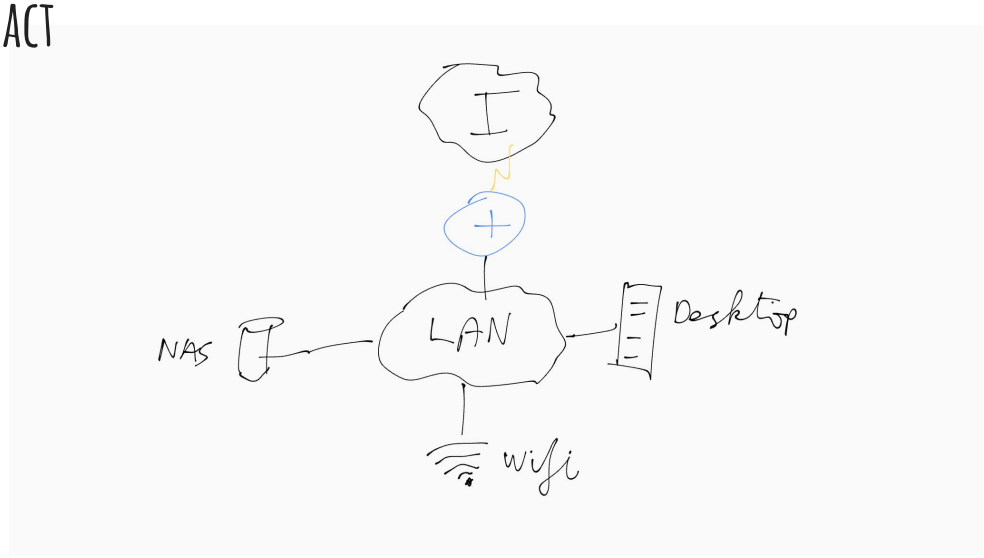
Layer 3



@msuriar

- Elide some physical devices, make subnets explicit
  - Wifi (phones, laptops)
  - Wired (desktops, NAS, etc)
  - Jail (untrusted IoT junk)

Abstract



@msuriar

- All network infrastructure devices (apart from main router) hidden.

Which of these..

- … is the most *accurate*?
- … is the most *useful*?

Alternatively, which would you use to debug...

- Stuttering video from your NAS?
- Slow page load times to usenix.org ?

@msuriar

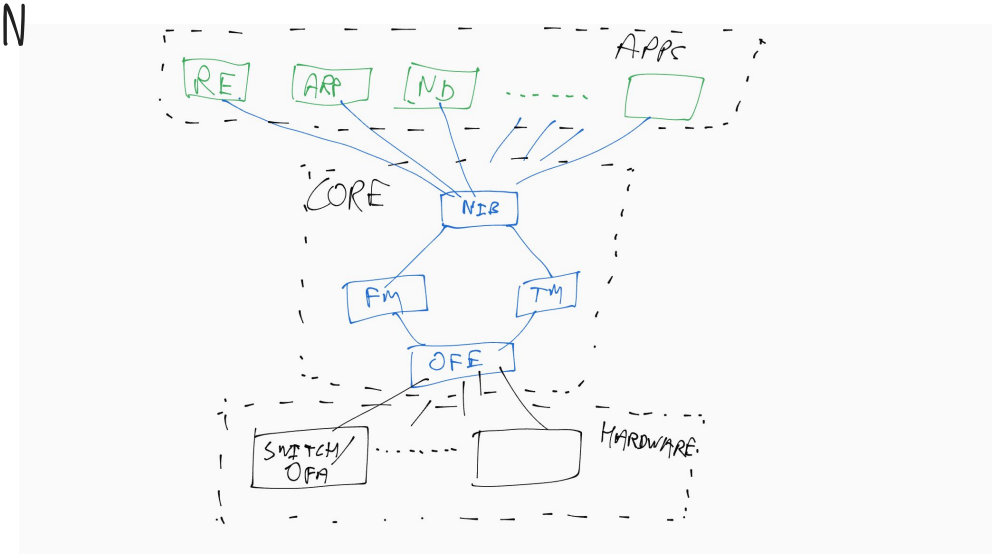
I'M AN ENGINEER

I solve practical problems

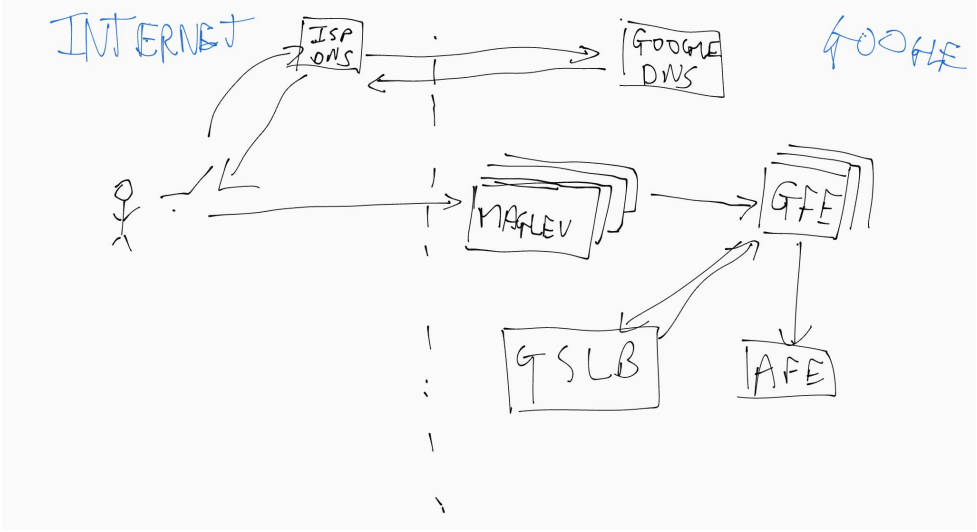
@MSURIAR

Demo time

Orion



Frontend



@msuriar

- ISP DNS: your recursive resolver
- Google DNS: Google's authoritative nameservers
- Maglev: Google network loadbalancers
- GFE: reverse proxy
- GSLB: internal RPC loadbalancing service
- AFE: "application frontend" - service-specific thing that can respond to query.

- How many people could you draw (or close to draw) one or the other of those from memory?
- How many people could then explain how X worked to someone else?

So what?

- Standard diagrams are useful tools
  - … for communication
  - … for training/onboarding
  - … for identifying opportunities to improve things
  - …

Standard diagrams build shared understanding

@msuriar

- In this case, everyone having the same diagram is much more important than it being strictly accurate
- Has knock on implications for system design
  - if a system doesn't have a useful abstraction that can be drawn from memory…
  - consider maybe it's too complicated to understand?

System diagrams are good for...

- Onboarding new team members
- Reinforcing shared understanding (or identifying inconsistencies in understanding) between team members
- Reasoning about the system
  - when it's working
  - … when it's not
- Identifying potential changes
- Understanding the impact of proposed changes to the overall system design

Right - now what?

- Take a system you're responsible for, try and draw it from memory
- Do it again every few days
  - what changes?
  - what stays the same?
  - does your intuition about the system improve?
- Do your diagrams help you explain things to colleagues?
  - Do their diagrams help you?
- Group exercise
  - Have your entire team each draw your system (alone)
  - Contrast and compare

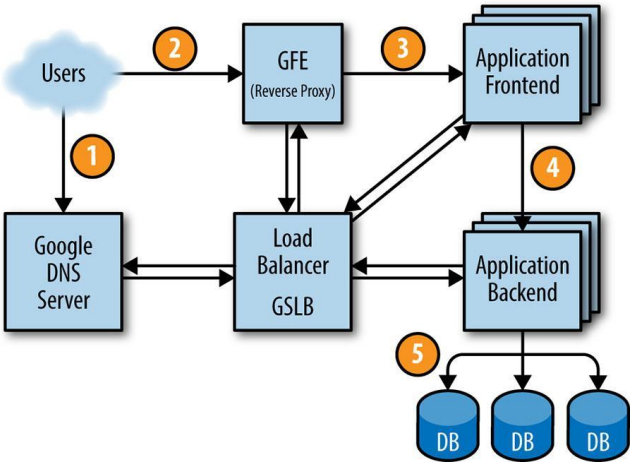
@msuriar

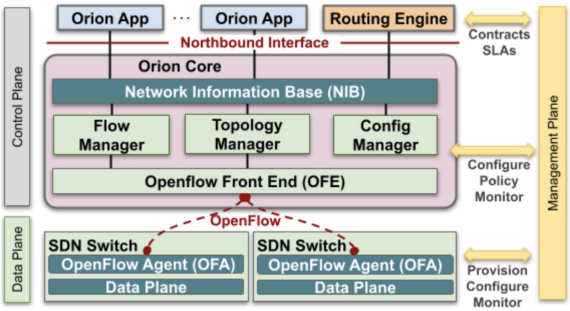
- Repetition and correction
  - Onboarding - do it several times with new hires, until they can do it themselves
- Get everyone on your team to draw a diagram of what you're oncall for

Use the tools which work for you

- Pen and paper
- Dry-erase board and markers
- Digital tools
  - SVG/DOT
  - mermaid-js (supported inline in Github markdown)

In case you thought I made those up...





@MSURIAR

- https://sre.google/sre-book/production-environment/#fig_production-environment_life-of-a-request
- https://www.usenix.org/system/files/nsdi21-ferguson.pdf - Figure 2

Questions?

@msuriar