Crayon Drawing

A Vital Engineering Skill
Obligatory
Audience
Participation

● 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.
Why are we talking about this?

- 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?

- 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?
We build abstractions
My Home Network™
• Every physical device, and every cable
- Elide some physical devices, make subnets explicit
  - Wifi (phones, laptops)
  - Wired (desktops, NAS, etc)
  - Jail (untrusted IoT junk)
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?
I’m an engineer

I solve practical problems
Demo time
- **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

- 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

- 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...

- [https://www.usenix.org/system/files/nsd-i21-ferguson.pdf](https://www.usenix.org/system/files/nsd-i21-ferguson.pdf) - Figure 2
Questions?