knowledge plane

- based on high level declaration of intent
  - assemble, re-assemble, detect failures & repair
- focus of this rant
  - failure detection & repair aspects

knowledge plane: preliminaries

- KP definition:
  - a plane orthogonal to separations between other functions, managing system failures
  - understands intent
  - an architectural direction
- types of failure
  - non-functioning component
  - incorrect information
  - poor performance (congestion or partial failure)
failure detection issues

- detection hard for
  - ‘incorrect information’ failures, e.g. incorrect DNS mapping
    - supplied IP address exhibits some failure (host not found, doesn’t understand protocol, object not found)
    - or next process steps work, but do the wrong thing, e.g. a stale DNS mapping leads to a stale information object
    - or perhaps addresses aren’t accessed, but used to build an (incorrect) topology map, perhaps for route optimisation
  - ‘poor performance’ failures, e.g. time-out interaction
    - slow response leads to time-out of request at head of request cascade

never-ending scope: scenario

- underlying comms svc (DNS, forwarding, proxies) all fine
- database behind Web server accidentally rolled back 1 day
- hyperlink stored in the database points to stale address
- causes the output of a sensor to go to the wrong place
- so faults in production line (monitored by sensor) reported to contractor who lost contract yesterday...
- KP should find root cause (the database roll back)
- factory ex-contractor doesn't even realise she is part of a communication system
- let alone that her data is needed to trigger fixing it
intent

• “what the network is supposed to do”?
  ≡ “what the superset of its applications are supposed to do”
• application author (partially) knows intent
  • doesn’t consider all the things that might go wrong
• source code partial representation of intent
  • only the mechanism considered nec. to achieve the intent with
    traps for foreseen potential failures
• if 3rd parties (KP) try to infer intent
  • inference errors will compound
  • discriminates against minority (incl. emerging) apps
• no intent role for KP
  • reduces KP role to correlation detection

responsibility

• declaration of operator responsibility & value
  chain relationships
  • need framework
  • nice problem to bite off separately
  • e.g. whois++ programme
    – who is responsible for IP address x? system y?
    – who has the contract for dealing with consumer/business faults
      due to failure z?
• some companies won’t publish their relationships
  • KP breaks ‘modularise design along tussle boundaries’
incentive issues

• incentives to reveal failure
  • my revenue depends on not admitting to failures unless forced
  • alternatives:
    – exception peering
      – failures affect retail revenue only; bulk allee in wholesale charges
    – hiding within aggregates
      – militates against fault tracing
  • both models require free-rider detection & penalty enforcement
    – see www.mmapps.org
• incentives to invest in KP
  • which model is realistic?:
    – p2p end-users only? operators too? 3rd parties as well?
  • will have to be bundled: people don’t buy fault mgmt software
    – why haven’t we even got good component fault detection?
  • who will invest the time to write federated code?

grandiosity

• need better bottom up component failure detection
  • better error reporting from components
  • less weakly defined semantics (defining time-outs etc)
  • consequent better application writing
  • need to ask why we haven’t even got that: no incentive
• ways to supply mgmt expertise to app developers
  • KP involves operating a separate service (run-time KP)
  • preferably supply mgmt libraries (build time KP)
    – hints to app developer on which exceptions to handle
knowledge plane: summary

• need framework to declare operator responsibility
• need inter-provider/layer correlation detection
• don’t need inference of intent – ever
• outstanding incentives issues:
  • why isn’t low level failure notification done well now?
  • incentives to reveal failure
  • incentives to build KP?
• KP ends & means: correct & questionable resp.
• KP has helped make comms mgmt research sexy

better alternative

• focus on whole system robustness
  • diversity in all dimensions
  • underlying simplicity
• occasional system use → less reliable
  • half the time, KP won’t work when you need it
  • management not the main driver of revenue
  • so give up KP direction now