Royal Society

Beyond the Bandwidth Crunch

11-12 May 2015

Cross Layer Decentralization

Today's networks are suffering from capacity constraints, not necessarily in the core, but sometimes in backhaul. However, who suffers most are users, and their quality of experience with wireless access performance (sometimes impacted by backhaul limits) being highly variable, patchy, and more crucially for many users and applications, latency and availability.

Its the hypothesis of this talk that the solution to this problem is not "more bandwidth everywhere soonest". The solution is massive decentralisation of networking, processing and storage. The idea of direct device to device comms (LTE direct, or WiFi Direct) and cooperative relaying, the ideas of personal cloud, and peer-to-peer storage, are not new. The point is that they have to become mainstream, engineered for (more) predictable performace, and to avoid systemic (e.g. emergent) failure modes.

The challenge is to embed this thinking in transfering research results into future systems designs, and associated standards.

Team Jon

3 cultures

- Internet
- Software
- Engineering

Internet:: Challenges #1

- Scale
  - IoT 1000 times size of current internet in complexity
- Offline-to-Online working
  - Has to work mostly offline
- Sustainability (batteries not included)
  - “things” last 3-300 times longer than computers
- Resilience
  - I have front door, back door, windows for fire escape
  - But only one broadband link – not good enough
- Security & Safety
  - Probably where law/regulation comes in too
  - Liability etc

Challenges #2

- Not the Internet as we know it, Jim
  - E.g. COAP&IPv6&LowPAN not HTTP/TCP/IP/WiFi
  - People don’t understand it much
  - Attacks are all new
  - Culture in embedded systems engineering is “security free”
- Invasive
  - sits inside your home, car, office, shoe, bloodstream
  - Is a Tussle Space
  - c.f. smart meter – nano-cloud in home
  - Utility/provider facing VM, partitions from:
  - User facing VM, running user apps, accessing:
  - Some of the shared data, but not all….

Challenges #3

- Human Privacy
  - Honest-but-curious IoT companies will look into your life
  - Will get it wrong – multiple occupancy is v. complex
  - Using smart phone to disambiguate doesn’t work
  - e.g. in bath, shower, with babies, visitors etc
- Human Comprehension
  - Breaks principle of least astonishment, daily
  - People work out their own modus operandi
  - Won’t work for others
  - Deep customisation run riot

Legacy means…

- We will have non-internet things for decades
  - Interop is a nightmare
  - Appliances, controllers, integration, billing/control
- Consider smart or autonomous car
  - Choose to go over cliff, or run down children
  - Seen with ABS already
  - But now we have feature interaction…
  - Smart city/roads – insurance companies “driving”

Culture Clash

- Device Controller s/w&h/w embedded systems
  - Dev Biz seen as Engineering, not Computer Science
- No culture of “secure by default”
  - No idea of threat model
  - Or existence any threat at all!
  - Mostly lack verification (except aerospace)
  - Study (E Leverettt) showed many HVAC systems
    - wide open
    - Fragile beyond belief
    - Think Windows 3.1/DoS, only worse, with tcp/ip ☹

Conclusions #1

- Bandwidth crunch? No, but
  - No clear business cases
    - Business Crunch
  - Few considerations of consequences of fails
    - Badness Crunch
  - Very poor safety and security models all over
    - Bad crunch!

Software:: Systems Challenge #2

- Several things must change…

Communications Systems Complexity

- The Internet&Cellular Networks are quite big
- They contain many protocols (e.g. >5000 RFCs)
- Much code….for each one
- Interactions are complex…

Programming Languages

- Written (at best) in C and Java (and Javascript)
  - Very bad idea☺
- E.g. TLS (you know, when https is green☺
  - had >50 vulnerabilities reported in 2014
  - Most were preventable if….
- Language Babel
  - C doesn't have type safety
  - C11 and Java definitions aren't complete on concurrency (see next)
  - Prejudice amongst low level engineers is
    - functional languages arent efficient (wrong)
    - even if they are (more verifiable

Communications and Concurrency

- Communications is inherently concurrent
  - And non-deterministic
  - And error prone (including fail silent, or out-of-thin-air)
- Very hard to test all cases
  - Even “corner cases” ill defined
  - Interactions very complex indeed
- Disconnect between layers
  - TCP (IP) meets RAN
  - Terrible plethora of problems
- Interaction between Admin Domains
  - Intra v. Inter-AS
  - End Systems, Routers, and Middle Boxes…..

Specification and Model

- Most systems specified in “English”
- Not exactly world’s most precise definitions
- Amenable/Accessible to lots of people
- No matter how few skills… (ASN.1)
- Or how many (SIP)

Culture Clash

- We have better languages&tool chains:
  - Modelling approaches &
  - Verification tools
- But we have legacy code
  - And legacy coders
- So we're tackling on both fronts
  - Re-engineer systems in safer languages w/ models
  - Try to fix languages (&compilers) (C11, JS)

Conclusions #2

- Bandwidth Crunch? Yes
  - Software Engineering Crunch
    - Production/Debugging
    - Reliability
    - Security
    - You name it…

Engineering Challenge #3
Unreliability as a QoE goal

Internet Classic

- Reliability a goal at every layer
  - modulo End-to-end argument
  - Integrity checks in link CRC
  - Integrity of routing & fwd
    - Ip hdr checksum
    - Reverse path check
    - Router-router (bgp) sec
  - Reliable, integrity of data, in-order delivery in TCP transport

Internet Classic Apps reliable

- Telnet, FTP, SMTP
  - Were all reliable
- SSH, HTTP-S, PGP add
  - Security/more integrity
- Distributed apps like E-Mail even
  - Had 40 days retry timers
  - from back in uucp days

Reality is less assured…

- We live in a filter bubble (pace Eli Pariser)
- What you get in
  - Search
  - Social Nets
  - Microblogs
  - And even E-Mail
- Depends
  - who you are, where you are, when you look
  - And your mileage may vary too

So all that hard work is to no avail

- Apps and proxies and middle boxes
  - Select, modify, remove etc
- So we should engineer accordingly
- Were aware of this for multimedia
  - RTP/UDP designed for loss tolerant codecs
  - Even do partial packet checksums…
  - Add symbol level net coding, etc etc
- Also for trading nets (gaming)
  - Pragmatic General Multicast

Go further: s/IP/DTN+ICN/

- users experience poor performance:
- *Because* of misplaced reliability effort
- Lets go back to drawing board
- Reliability (even integrity) is the exception, not the rule
- So, Information Centric Net (ICN)?
  - Aka Named Data Networking
  - Substitute pub/sub for IP…

Not enough just ICN…

- Need delay tolerance too…
- And delay intolerance –
  - When to discard (idea from PGM):
  - Time limited window of delivery
  - When to forget (cache eviction etc)
- Note also integrity
  - may not care which version, just any will do
  - May not care if transcoded en route

Locality of Relevance

- As with googling, facebook, twitter
- What you get depends
  - when you look,
  - Where you are
  - Who you are
- Locality is a good optimisation technique
  - But if we keep all that reliability legacy
  - We downright downrate the user experience!

In summary

- to re-assert unreliability as a feature
  - Not a bug
- Modern apps re-target e2e argument
  - In app protocol layer, not transport
- Your mileage should vary less
  - And be in kilometers, anyhow

Conclusions #3

- 5 9s (99.999% available) maybe false goal
- 10 9s better
  - but not on a component by component basis
  - Systems approach
  - Diversity (components, paths, software)
  - Decentralization
- Bandwidth Crunch: Yes, but no…

Conclusions #3,2,1

- Need radical, cross-layer, re-think
- Some cultural challenges
- Integrating across layers&disciplines…

- yottabits/sec is not the challenge –
  - forget bandwidth
- The right bits per micosecond is
  - Latency and availability are king