ELEC-E7320
Internet Protocols Course
Protocol Design in the Real World

March 1, 2017

Jari Arkko
Chair, IETF
Senior Expert, Ericsson

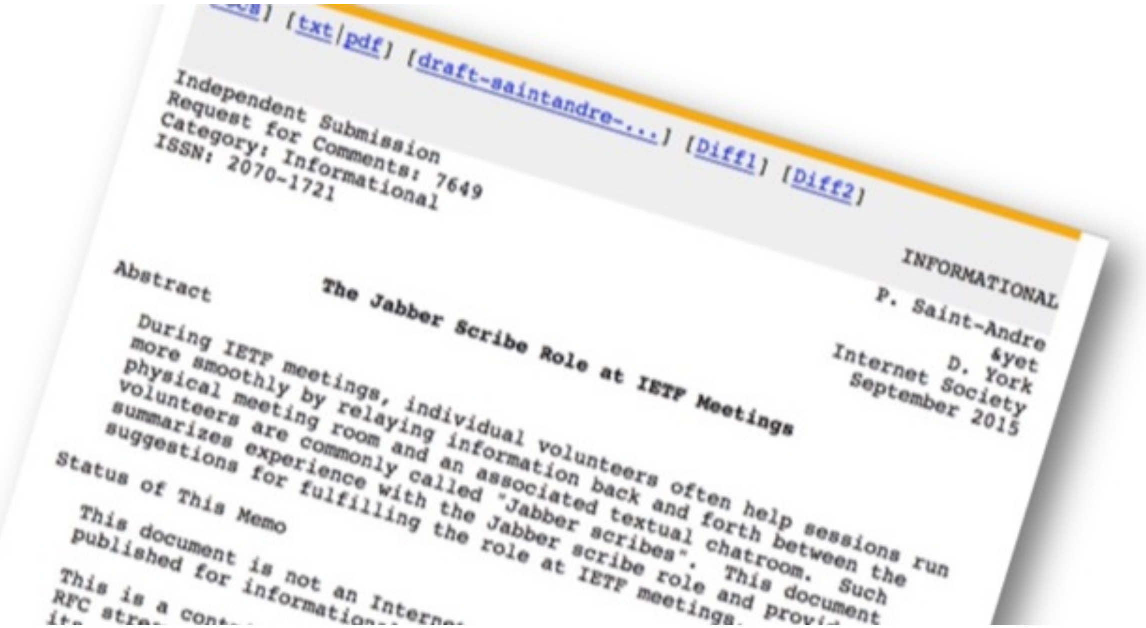
Contents

- IETF: what it is and what is it working on?
- Standards: Why? How?
- What makes for a successful protocol?
- Protocol design in theory vs. real world

IETF

The Internet Engineering Task Force is a loosely self-organized group of people who contribute to the engineering and evolution of Internet technologies.

It is the principal body engaged in the development of new Internet standard specifications. *(RFC 4677)*


[txt|pdf] [draft-saintandre-...] [Diff1] [Diff2]
Independent Submission
Request for Comments: 7649
Category: Informational
ISSN: 2070-1721
INFORMATIONAL
P. Saint-Andre
&yet
D. York
Internet Society
September 2015
The Jabber Scribe Role at IETF Meetings
Abstract
During IETF meetings, individual volunteers often help sessions run
more smoothly by relaying information back and forth between the
physical meeting room and an associated textual chatroom. Such
volunteers are commonly called "Jabber scribes". This document
summarizes experience with the Jabber scribe role and provides
suggestions for fulfilling the role at IETF meetings.
Status of This Memo
This document is not an Interne
published for information

I E T F

The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet.

(RFC 3935)

IETF

86 30 16

YEARS

Standards: Why? How?

Benefits

- Customers need standards to avoid being locked to a vendor
- Everybody benefits from creating a large market and ecosystem with compatible products
- The more reviewers, the less errors; security

Drawbacks

- Standard development may take more time
- A company won't have full control

Different Approaches to Standards

- Improving proven proprietary tech, making it open, and elevating status to a standard
- Developing new tech openly, together
- Code is the standard – open source
- Open vs. closed participation
- Available vs. not available specs
- Formal vs. informal standards

IETF


open
Task
Force
request
HTTP
IP
IPsec
IETF
consensus
security
code
meeting
virtual-meeting
rough-consensus
TCP
comments
privacy
mailing-list
IPv6
for
TLS
Internet
source
Engineering
running-code
RFC

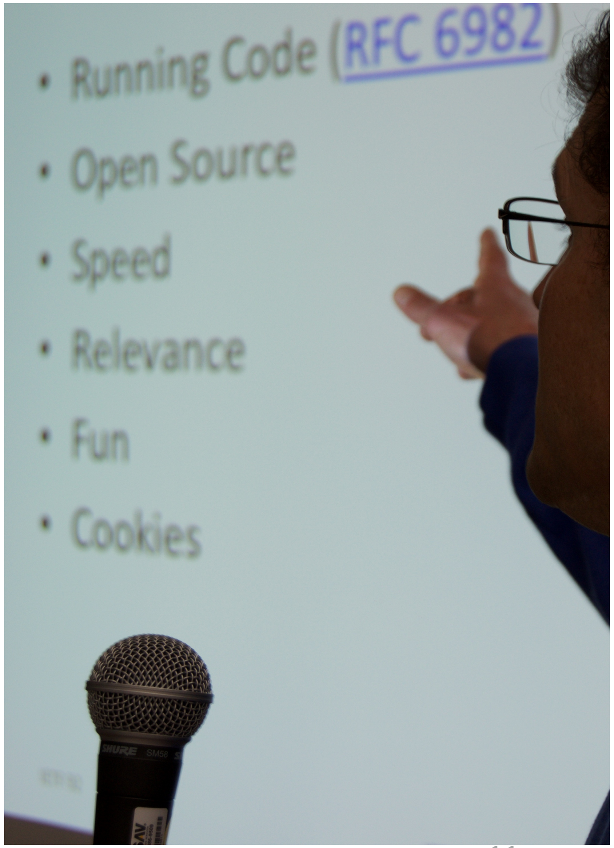
Roles of Different Standards and Organisations in Internet Tech

Browsing The Web			
802.11	**IEEE**	LTE	**3GPP**
TLS	**IETF**	TCP/IP	**IETF**
NAT	**IETF**	HTTP	**IETF**
CSS	**W3C**	BGP	**IETF**
HTML	**W3C**	MPEG	**ISO/IEC**
XML	**W3C**	ADSL	**ITU-T**

Interoperability

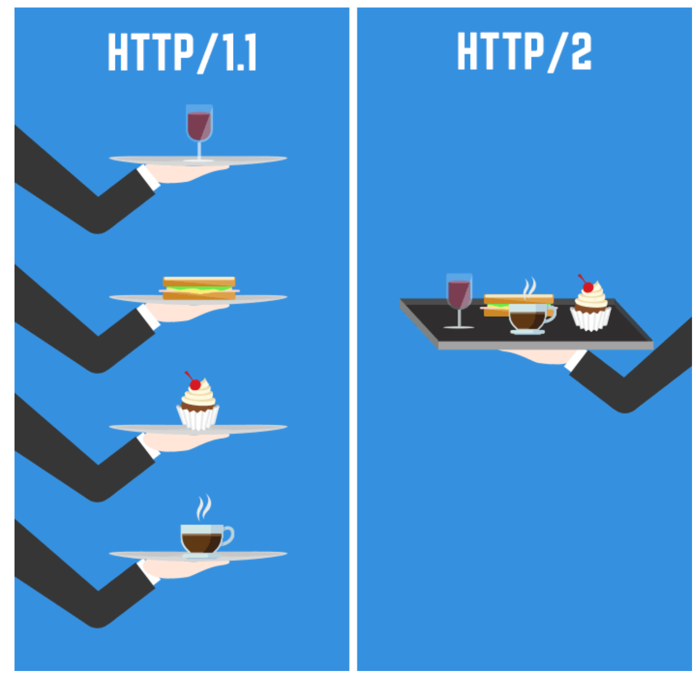
Some Areas of Active Work at the IETF

- Web protocol stack evolution (HTTP2, QUIC)
- Security & privacy (RFC7258, DPRIVE, TLS1.3)
- Enabling real-time communications in browsers
- Internet of Things
- Management, orchestration, virtualisation, and data-model driven networking (NVO, SFC, YANG)



Web Protocol Stack

- Overall, much change in last few years: HTTP2, certificate pinning, HSTS, webpush, increased use of encryption, WebRTC, TLS 1.3, …
- Now tackling even bigger changes: QUIC and the changing transport layer
- Why is this happening and what does it mean for the Internet?

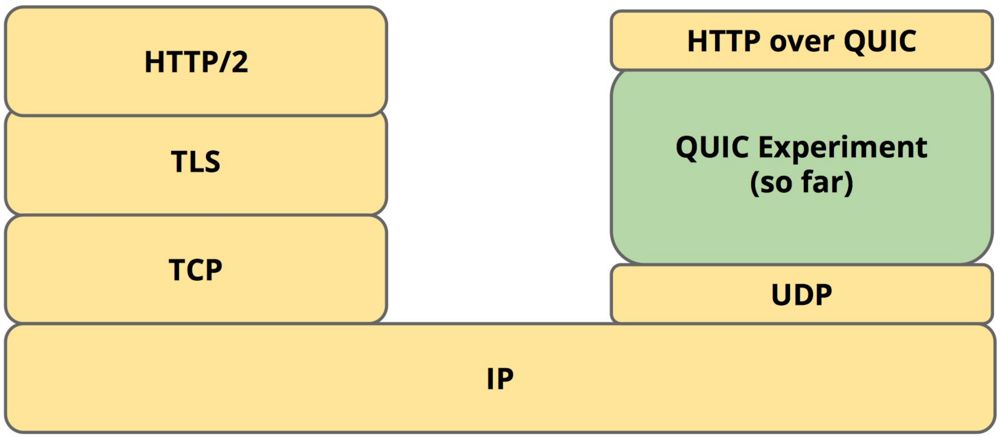


QUIC

Integration of transport and TLS; runs on top of UDP, in full control of applications

Goals:

- Minimise connection establishment time
- Multiplexing without head-of-line blocking
- Deployability and evolvability



QUIC Background

- We needed all this those things…
- As you know, almost everything runs on top of the web
- But also …

QUIC Background

- We needed all this those things…
- As you know, almost everything runs on top of the web
- But also, consolidation of Internet services, traffic, OSs and applications plays a role
- Internet architecture and role of endpoints plays a role as well, as does the ease at which software today gets updated

Observations

- Prediction: Big shifts so far, even bigger ahead
- Functionality moves to applications & browsers, fast change
- Encryption change was just an example — others will follow: specialised transports for movie download, etc.
- Applications are firmer in control: e2e security, browsers, now transport — what's next?

Success and Failure of Protocols

What Makes for a Successful Protocol?

- RFC 5218 (Thaler & Aboba)
- Following material adapter from Thaler's presentation at IETF 70

- Why think about this?

What Makes for a Successful Protocol?

- RFC 5218 (Thaler & Aboba)
- Following material adapter from Thaler's presentation at IETF 70

- Why think about this?

  So many technology developments never get widely deployed, or even used at all.

  Yet some are wildly successful

  Which one do you want to work on?

  Is it worth for us to attempt develop tech X?

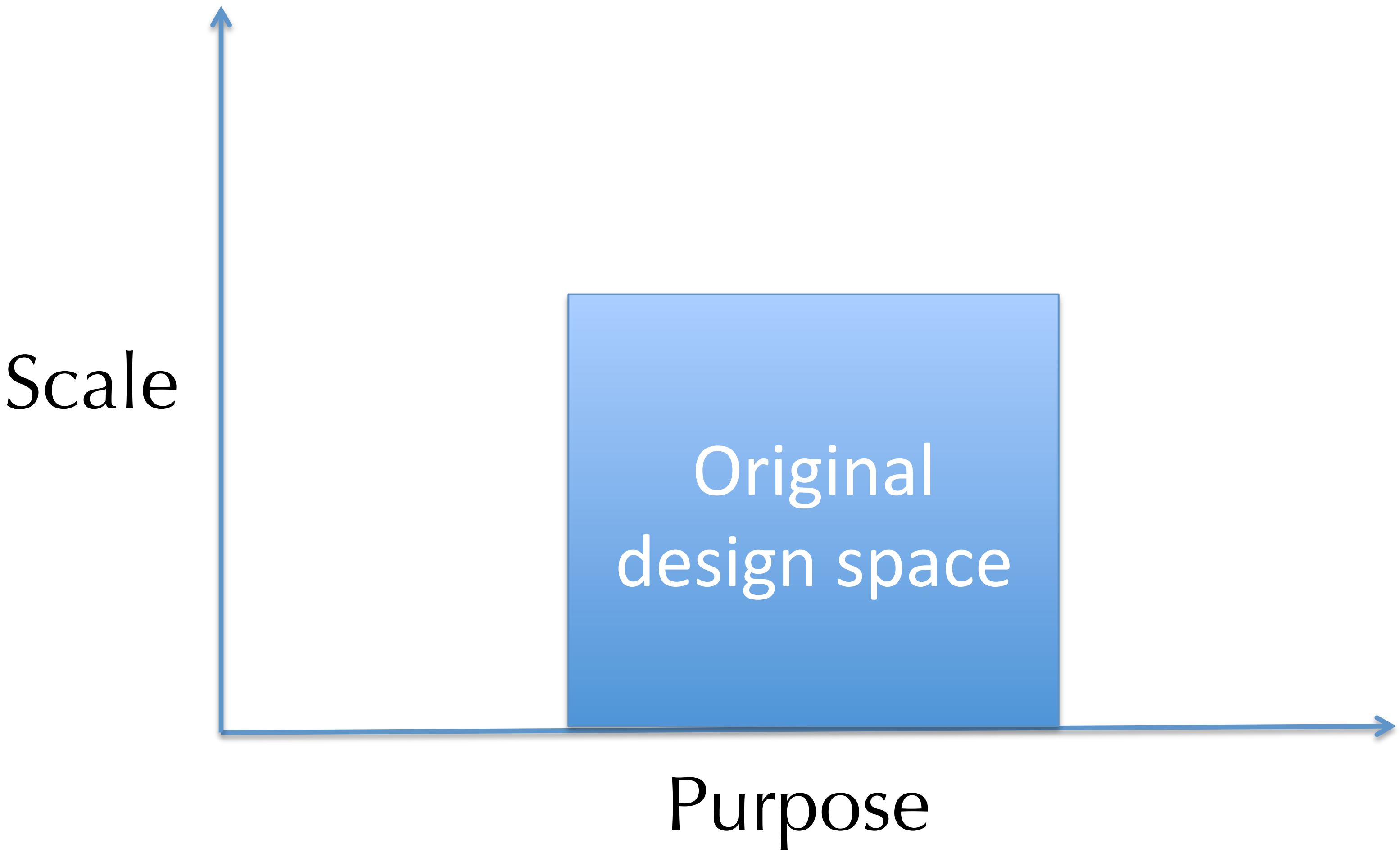
What Is Success?

- A protocol can be successful and still not be widely deployed, if it meets its original goals
- Protocols running **across the Internet** vs. **more local usage**
- We might consider the following as some examples of successes:
  - IPv4, TCP, HTTP, DNS, BGP, SMTP, etc
  - Local usage ARP, DHCP, RIP, OSPF, etc

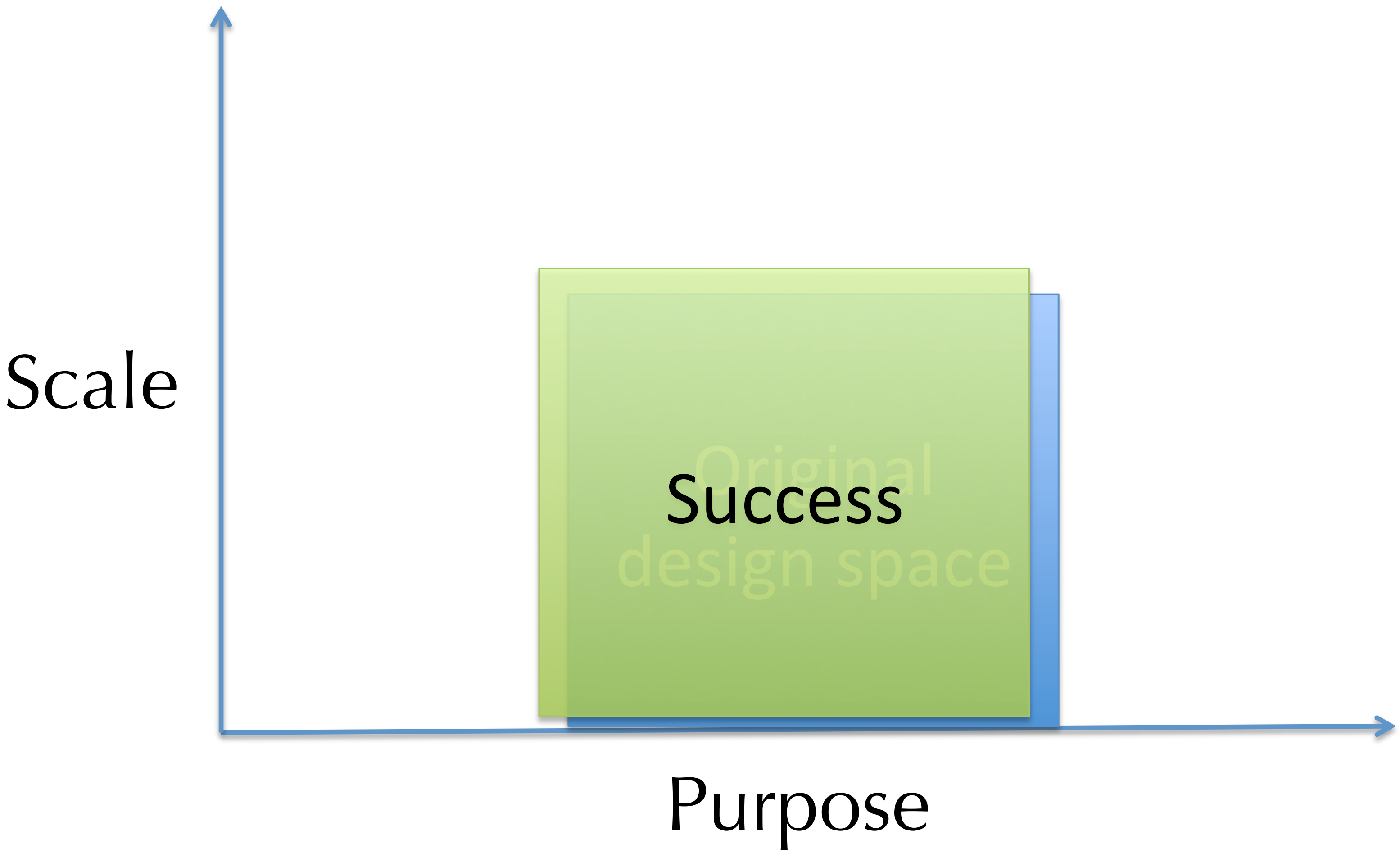
Defining Success

- **Successful**: a protocol that is used in the way it was originally envisioned, and to the scale it was originally envisioned
- **Wildly successful**: a successful protocol that is deployed on a scale much greater than originally envisioned and/or in ways beyond what it was originally designed for.

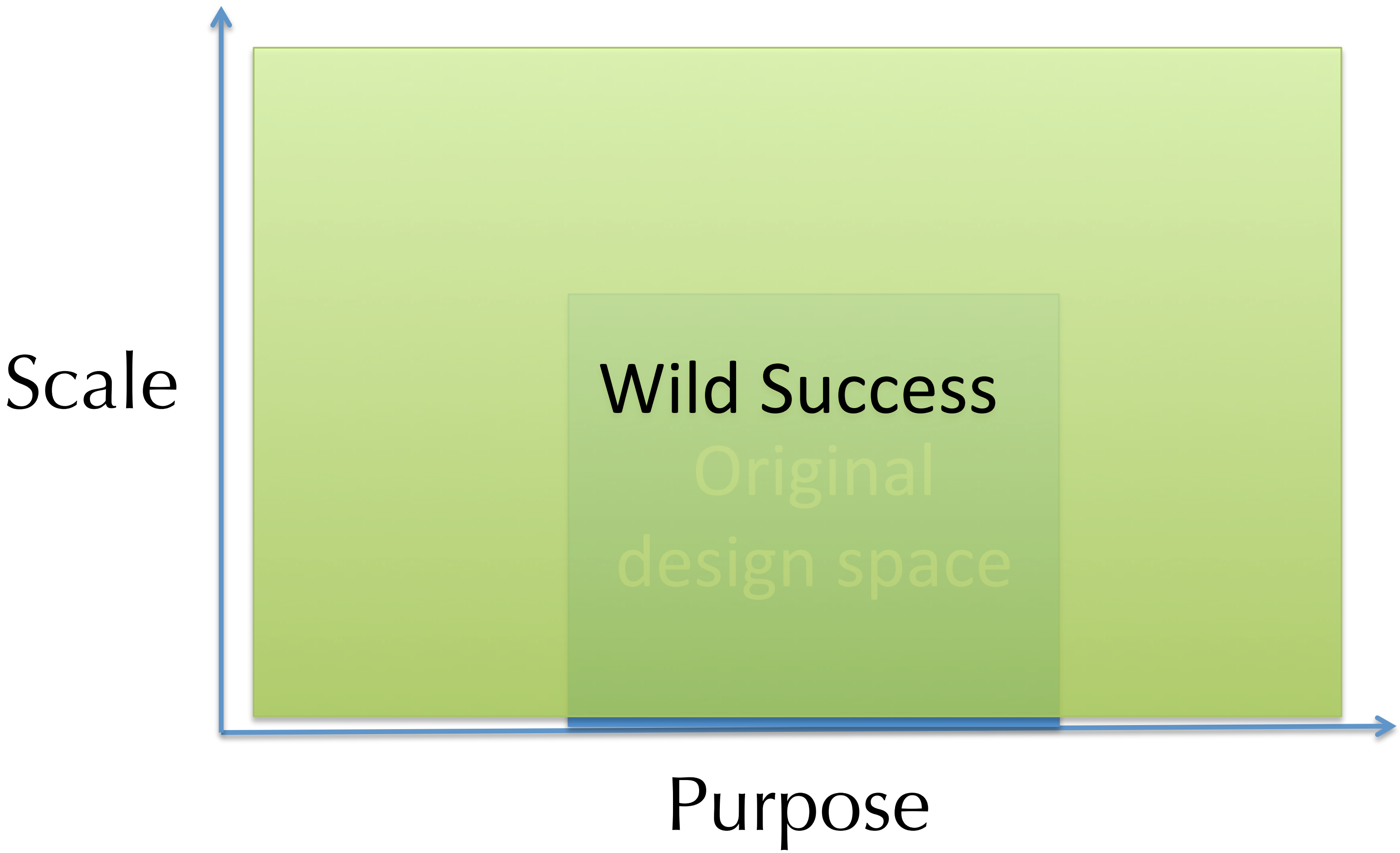
Defining Success



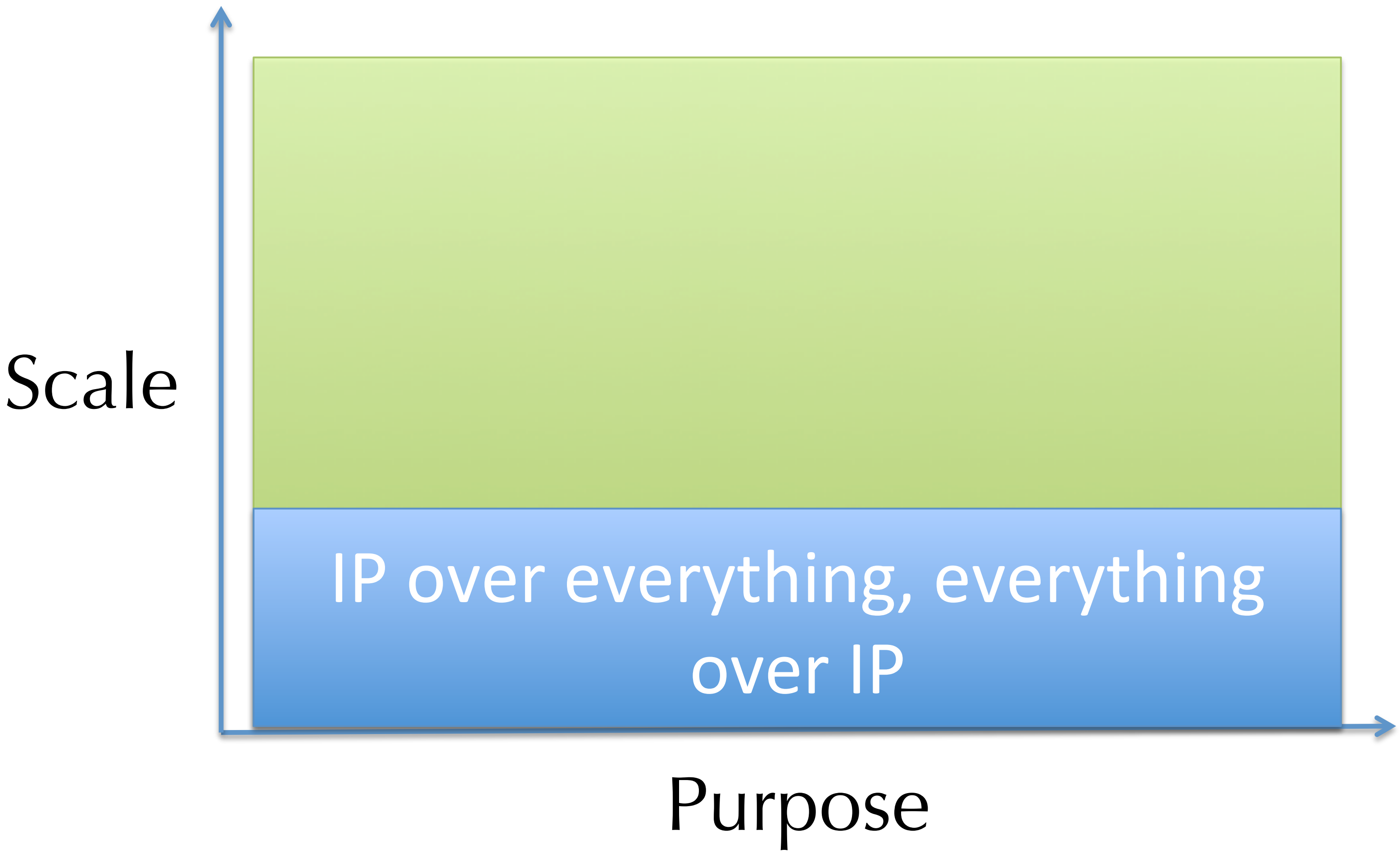
Defining Success



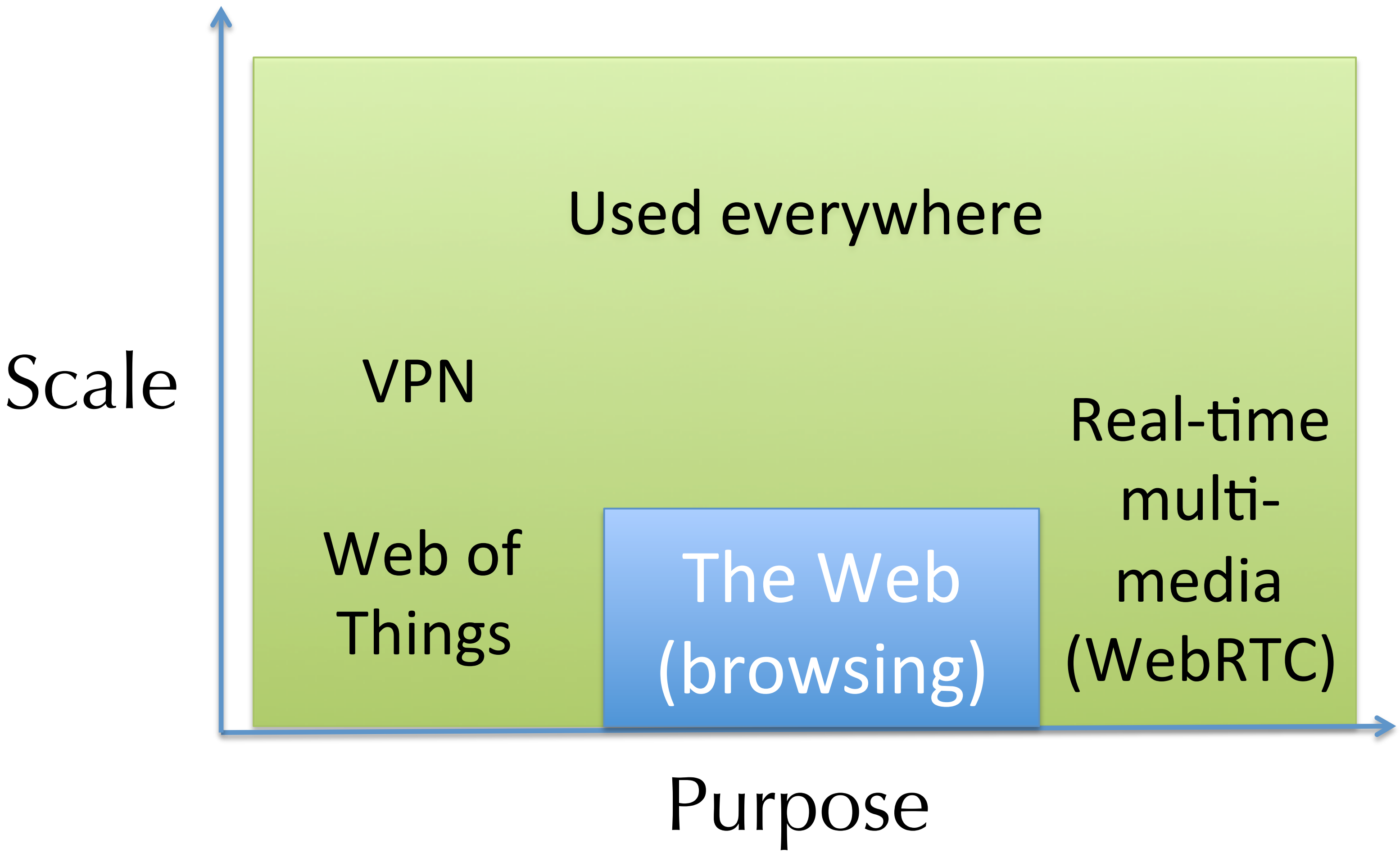
Defining Success



Example: IP (the original, IPv4 version)



Example: HTTP



Wild Success

- Can be both good and bad
- Undesirable side effects when used outside intended purpose
- Performance problems
- Ugly hacks to work around design limitations
- High value target for attackers
- "Death by success"

Failure

Common problem:

- No usage
- No implementations
- No support
- A chicken-and-egg problem

Other issues

- Assumptions about environment turn false
- Too difficult / complex / costly

Avoiding the Chicken-and-Egg Problem

Avoiding the Chicken-and-Egg Problem

- Address a critical and imminent problem
- Provide a “killer app” with low deployment costs
- Provide value under existing unmodified apps
- Narrow the intended purpose to an area where it is easiest to succeed
- Reduce cost by removing complexity not required for that purpose
- Governmental (dis)incentives: promise of longterm economic or military benefits

Failure Example: IP Mobility Protocols

- Make it possible to retain sessions as you move around, while changing your IP address
- Why did it fail?

Failure Example: IP Mobility Protocols

- Make it possible to retain sessions as you move around, while changing your IP address
- Why did it fail?
  - Target market squeezed from two sides
  - Link layers started hiding (most) movements
  - Applications learned to not care about address changes

Success Factors

- Positive net value (meet a real need)
- Incremental deployability
- Open code availability
- Freedom from usage restrictions
- Open spec availability
- Open development and maintenance processes
- Good technical design

"Wild" success factors

Success Factors

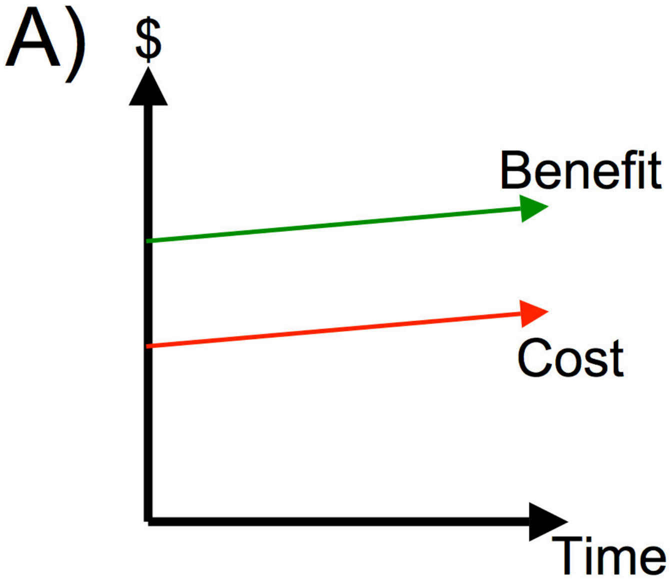
- Positive net value (meet a real need)
- Incremental deployability
- Open code availability
- Freedom from usage restrictions
- Open spec availability
- Open development and maintenance processes
- Good technical design

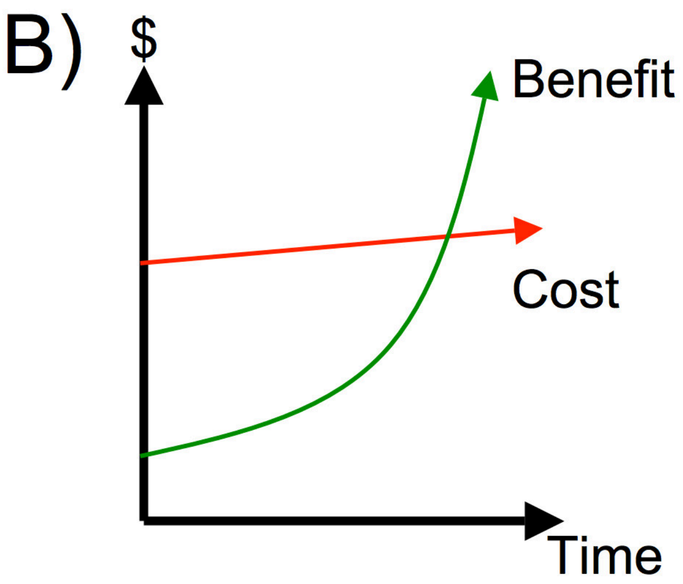
“Wild” success factors

- Extensibility
- No hard scalability bound
- Threats sufficiently mitigated

Positive Net Value

- The benefits (e.g., monetary) of deploying the protocol clearly outweigh the costs of deploying it





Benefits and Costs

Benefits

- Remove pain
- Enable a new scenario (riskier)
- Incremental improvements (lower payoff)

Costs

- Hardware changes
- Operations changes
  - Can use overlays
- Retraining
- Business model changes

- There must be incentives for change at *every* network whose participation is required

Incremental Deployability

- Early adopters gain some benefit even though the rest of the Internet does not yet support
- Autonomy: deployment by a single party is easier than cooperation across multiple organizations (no flag day)
- One-end benefit: benefit from only one end changes is useful (e.g., MIPv6 vs. HIP)
- Backward compatibility: backward compatibility with legacy implementations

Code and Spec Availability, Restrictions

- Open code availability often more important than technical factors
- Legal or commercial limitations may hinder deployment
- Open spec availability helps allow multiple implementations

RFCs are

- Accessible anywhere
- Distributed without restrictions or payment
- Permanent
- Stable

Extensibility and Scalability

- **Extensibility**: easy to add more information to be carried, easy to evolve the overall protocol design
  - E.g., TLVs, version numbers
  - Software is changeable (e.g. kernel vs. app)
- **Scalability**: no inherent limit near the edge of the originally envisioned scale
  - Size of "address" fields
  - Performance "knee" that causes meltdown
  - How did IPv4 do here?

What Factors Are Important?

Less important initially

- Technical design
- Maintenance

Important initially

- Very positive net value
- Availability of code, specs

Important for wild success

- Extensibility
- No hard scalability limits

What Should You Do?

Ask questions for your new project!

- Do the success factors exist?
- Can the technology help potential high-profile customers?
- Are there potential niches in desperate need?
- How extensible should the protocol be?
- If success is uncertain, should you wait or do something else?

Example: QUIC and Success Criteria?

Example: QUIC and Success Criteria?

Important initially

- Very positive net value ✔
- Availability of code, specs ✔

Less important initially

- Technical design ✔
- Maintenance ✔

Important for wild success

- Extensibility ✔
- No hard scalability limits ✔

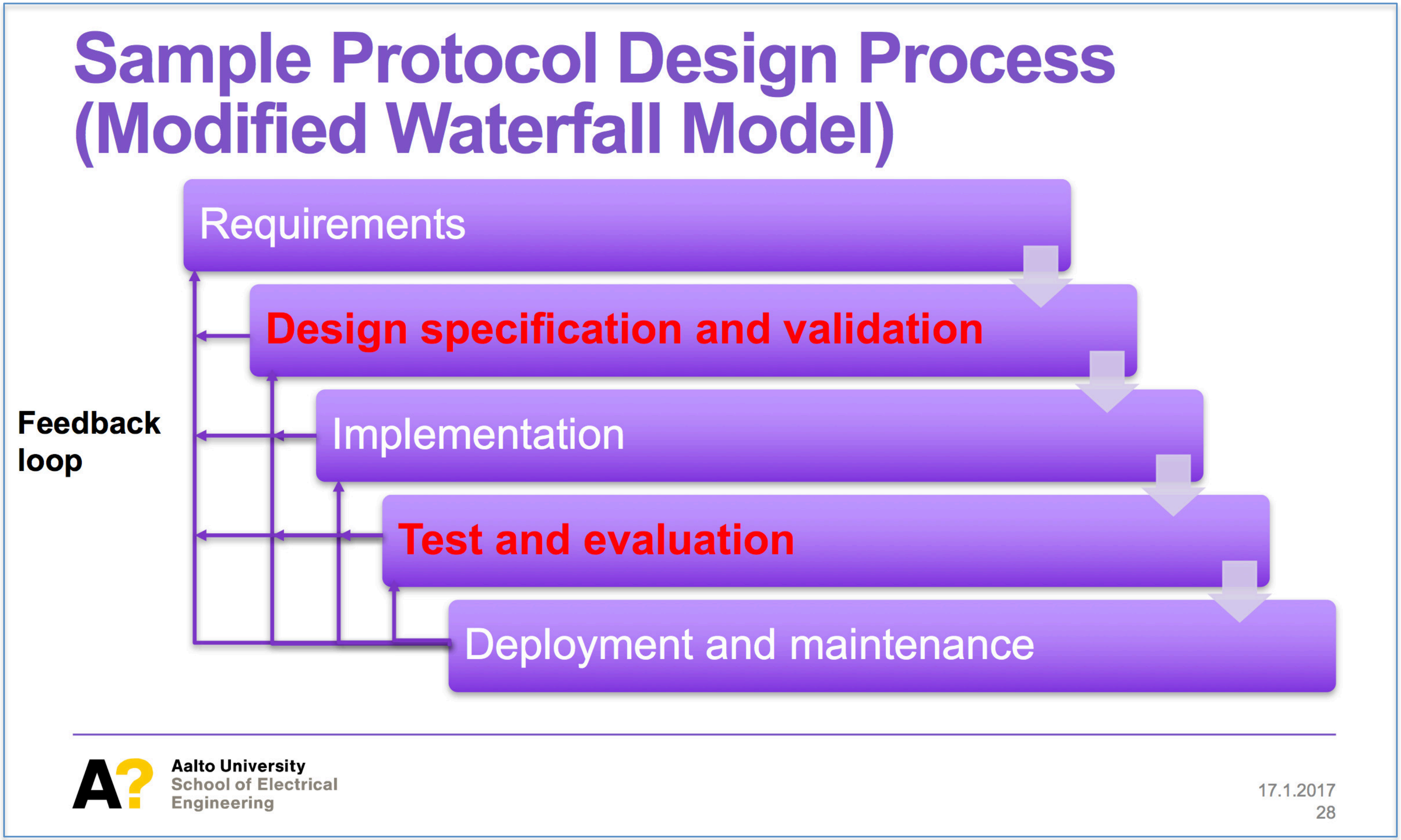
Theory vs. Practice

Protocol Design in Theory vs. Practice

- Our theory is solid
- We know how to design good protocols
- But when mixed with reality, you'll get a mixture of factors affecting outcomes

- Time-to-market, business decisions, IPR, …
- Human factors, fashionable solutions, marketing
- …

Theory


Sample Protocol Design Process (Modified Waterfall Model)
Requirements
Design specification and validation
Implementation
Test and evaluation
Deployment and maintenance
Feedback loop
Aalto University
School of Electrical Engineering
17.1.2017
28

Practice

- You rarely get handed opportunities for clean slate design
- "Reduce latency for searches" (or VR, 5G, …)
- "… while not being able affect any of the components on the path"
- "… while supporting all legacy clients"
- Most tasks we end up doing are small but continuous improvements

Practice

- Requirements and careful design are not the only criteria that is applied
- It may make economic sense to push a product to market early, with limited validation, testing, security, or extensibility
- More seriously, requirements can be stated in a particular way to drive a particular outcome
    - To "win", have time-to-market advantage, …

Modularity – A hierarchy of functions

- A protocol that performs a complex function can be built from smaller pieces that interact in a well-defined and simple way.
- Each smaller piece is a light-weight protocol that can be separately developed, verified, implemented, and maintained. No assumptions about other modules
- Main structuring techniques: protocol layering, structuring of data

Practice

- That was actually very good advice
- And it is followed… mostly
- But there is a lot of pressure against it, mostly for short term benefits, outweighed by costs and difficulty of evolving the design later
- Bundling layers together allows optimisation (QUIC)
- Bundling with cloud or security or identity services may force users to use a particular service

Practice

- Other examples of bundling for optimizations?

Theory

Internet Architecture Board (IAB)
Request for Comments: 6973
Category: Informational
ISSN: 2070-1721

A. Cooper
CDT
H. Tschofenig
Nokia Siemens Networks
B. Aboba
Skype
J. Peterson
NeuStar, Inc.
J. Morris

M. Hansen
ULD
R. Smith
Janet

July 2013

Privacy Considerations for Internet Protocols

Abstract

This document offers guidance for developing privacy considerations for inclusion in protocol specifications. It aims to make designers, implementers, and users of Internet protocols aware of privacy-related design choices. It suggests that whether any individual RFC warrants a specific privacy considerations section will depend on the document's content.

Practice

- A lot of attention on security & privacy
- Many improvements have happened

- All necessary
- But, not entirely for the sake of protecting users
- Communications security is helpful, but other avenues for attack remain (at servers etc)
- Multiple business reasons for e2e security

Theory

Goal:
Permissionless
Innovation

IP as the common substrate to build things on

Mini note: HTTP is more and more the de-facto substrate

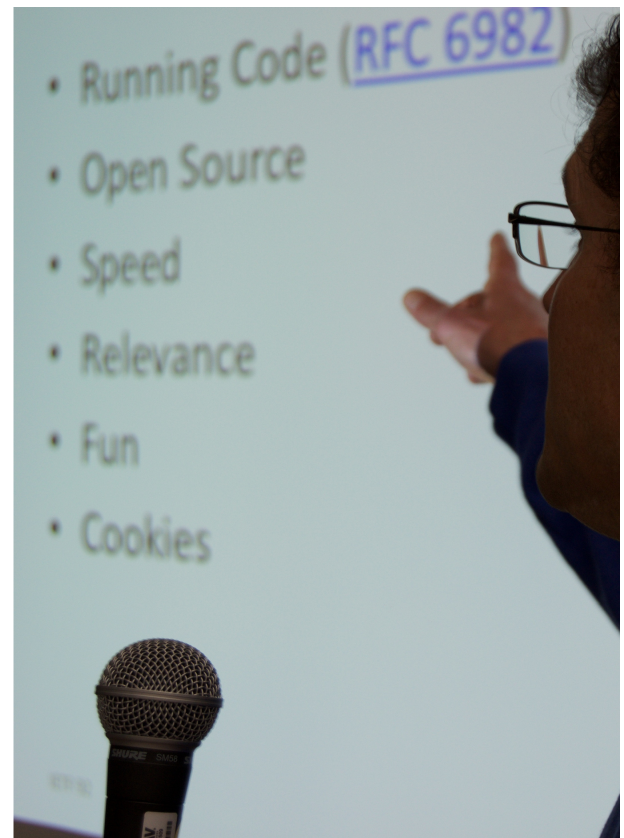
Practice

- Permissionless innovation is what gave us the current Internet services
- IP, the Web, WebRTC, Web of Things, SDN all promise to provide this – and Wild Success!
- But sometimes tech can be so good that the results are surprising
- Companies can create hugely successful but closed systems on top
- Basic technology may no longer get updates
- Keep the path open to new entrants!

Concluding Words

Concluding Words

- Protocols are evolving at a fast pace today
- You can make an impact in the tech
- The tech can have an impact on real world issues
- There's a lot of power in the community and ecosystem
- Use it, build a following



Questions
&
Answers
&
Discussion