Interoperability
Evolvability

Protocol Design

Interoperability

- Between implementations from different sources
  - specification quality
  - complexity
  - testability, debuggability
- Between less and more complete implementations
  - negotiation
  - optional functions
- Between early (buggy) and later implementations
  - robustness
- Between V1 and V2 implementations ➔ **evolvability**

Aiding extensibility

To enable V2, extensibility must already be built into V1

- Standard approaches: extension points
  - Managing protocol numbers (IANA!)
  - Negotiation (latency!)
  - Identifying optional information, reacting to it if understood
    - E.g., reserved fields (in V1: sent as 0, ignored on reception)
- Alternative:
  meta-information allows selection of appropriate version
  - Configuration (e.g., POP3 vs. IMAP)
  - Referencing data (e.g., URI schema)
  - Directory information (e.g., DNS SRV record)
  - Pre-negotiation

Never use up all extension points

Drivers for evolution

- Deployment experience
  - (handling old problems better, correctly at all)
- Environment changes, brings new requirements
  - At best, market driven evolution
- Protocol is applied to new problems
  - (but do they fit?)
  - Sometimes academic/vendor/architect driven evolution
- Box vendors want to sell new boxes
- Architects want to make new/better architecture
  - Often in the name of evolvability!

What is Evolvability?

- The ability to evolve easily
- Technology and human organization
  - What is the process that guides the evolution?
  - Is there an architecture, guidelines for future development? Does anyone guard against mission creep?
- Do you believe in “futureproof” technologies?
  - The junkyards are full of these
- Designing to be part of something else
  - Interfacing with the evolving environment
  - Accommodate **unforeseeable** requirements

[based on Tim Berners-Lee]

The “Test of Independent Invention”

- Design:
  - Important architectural decisions
  - Arbitrary decisions (“byte order”)
- Thought experiment: Somebody else invents the same
  - At some point, both designs will meet in the marketplace
- Now what?
  - A huge battle, involving the abandonment of projects, conversion, loss of data?
    - Sweden switches to driving on the right side of the road
  - Division of the world into two separate communities?
    - 110 V, 60 Hz, 525 lines, NTSC ↔ 230 V, 50 Hz, 625 lines, PAL
  - Smooth integration with only incremental effort?
- Can they be made to interoperate?
  - (Alternative: Wait until one has beaten the other)

How to obtain Evolvability?

- There are no hard and fast answers
  - Too many forces pull on a protocol design
- Rule 1: It is almost always wrong to optimize for the moment
  - Protocols need two, three years before they actually arrive on the market
  - Deployed life may then be 5, 10, 30 years!
- However, it is also wrong to optimize for an unknown future
  - Even if Moore's law can be taken into account:
    - Adaptive range needs to go into values that may seem preposterous now
  - Future requirements, future solutions can't
- The only constant is **change**!

- Let's look at specific protocols…

Case study: IP (1)

How did IP evolve? Not really much!

Addressing architecture: Two-dimensional (net/interface) in 32 bit

- Original: 8+24
- Class-based: 7+24, 14+16, 21+8
  - Augmented by subnetworking
- CIDR (class-less inter-domain routing): N+M
  - Killed RIPv1 (replaced by RIPv2 or OSPF)
  - Required host changes in ICMP, DHCP, forwarding
- End-of-life in full view ➔ IPv6 (complete redesign)

Case study: IP (2)

Other field sizes:

- 16-bit fragment ID (out of 32 bits): disaster in the making
  - **RFC 4963**: MTU 1500 bytes, MSL 30 s ➜ 26 Mbit/s max!
    - Hosts generally ignore this ➜ large number of mis-associated fragments can result
  - Fragmentation creates large number of other problems
    - DoS attacks on fragment buffers, making life harder for middleboxes
  - Implementations generally try to avoid fragmentation
    - Hard to do for certain UDP-based applications
  - Oh, and there is one free bit of extensibility left!
- 4-bit IP header length
  - Uses only 5-15 range: 40 bytes of options max
  - Seriously limits usefulness of IP options

Case study: IP (3)

Other field sizes (continued):

- 8-bit Precedence/TOS field
  - Now split into 6-bit TOS and 2-bit ECN
- 16-bit header checksum: useless, but impossible to reuse
- 8-bit protocol ID: serious limitation for protocol number assignment
- 8-bit TTL: apparently fine!
  - After de-facto redefinition from “time” to hop count

An IP innovation: IP multicast

- Previously unused address space: Class D
- New host-to-router (host-to-subnet) protocol: IGMP
- Requires pervasive host/router changes
  - Pretty much deployed, but not turned on on the router side
- Huge impact on routing infrastructure
  - Started out as overlay network (successful), DVMRP
  - Tried to “go native” (and died), PIM + BGMP
    - Never finished
    - A limited version survived as MSDP
- Essentially failed for global deployment
  - Works well in a corporate network or in special environments (academic)

An IP innovation: Integrated Services

- A new signalling protocol: RSVP
- QoS specs: Controlled Load (C-L), Guaranteed Service (G-S)
  - C-L is compatible with Ethernet style network
  - G-S requires more (ATM-style) control
- Requires pervasive host/router changes
  - Pretty much deployed, but not turned on
  - Applications don’t know how to make use of this
- Essentially failed
  - Almost nobody wants to pay for resource reservation
- Spawned successor (“ng” effort): NSIS

An IP innovation: ECN

- Original congestion management idea: ICMP source quench
  - Misguided (sending additional packets to signal congestion)
  - Never clearly defined (send them when, what do they do in hosts, see RFC896)
- TCP congestion control works with one signal: packet drop
- ECN: one more bit of router→host information (+ 1 host→router)
  - It was hard enough to free two bits

- Slow Deployment
  - Problems with middleboxes choking on these bits
    - Based on earlier experience with attackers playing tricks on rarely used bits
  - Situation only slowly improving (TBIT initiative)
  - 2006: ECN generally not turned on in client hosts (desktops)
  - RED is hard to tune (hard to configure routers to signal ECN)
  - But it is still too early to declare outright failure

IP: The verdict

- Apart from TTL, all field sizes were wrong
  - But then,
    the requirements of 2000’s Internet really were impossible to foresee in 1978
- Almost all innovations at the IP layer since 1990 failed
  - Often, hosts **and** routers would have had to upgrade — chicken and egg

- IPv6 is a better protocol
  - Unfortunately, incentive to deploy not clear in all markets

Case study: TCP (1)

How did TCP evolve? Extremely well!

- RFC 4614 (TCP roadmap)

- Some parts became obsolete
  - PSH flag is useless
  - Handling of IP precedence and security compartments
  - Urgent-pointer (out-of-band data) is near-obsolete
- **Algorithms** were replaced a lot!
  - General operation: e.g., silly window avoidance (RFC813)
  - RTO estimation (RFC1122, RFC2988)
  - Most prominently: congestion control
    - RFC 896 (January 1984!) diagnosed congestion collapse
    - VJ's 1988 paper showed the solution
    - RFC 2581 = Reno TCP documents it in detail:
      slow start, congestion avoidance, fast retransmit, and fast recovery.
    - Many more congestion control and retransmission tweaks were made or proposed

Case study: TCP (2)

- RFC 1323 fixed the more important field size problems
  - Optional window size scaling fixes 16-bit windows
  - Optional timestamps can be used to overcome 32-bit sequence number limit

- TCP was adapted to IPv6
- TCP supports jumbograms
  - Minimal changes in MSS option and Urgent pointer

- TCP now supports selective acknowledgements (SACK)

- TCP now supports ECN

TCP innovations that didn't work

- RFC1263: replace options by an elaborate versioning scheme
  - Would have added roundtrips at the start of each session
  - Would have reduced, not added to, interoperability
- T/TCP (transactional TCP)
  - Save 1/2 of a roundtrip
  - Too easy to attack
- RFC1693: Partial Order Service
  - Lack of interest
  - Was suppressed by ALF craze
  - Ideas later resurfaced in SCTP

Why did TCP evolution work so well?

- Simple service, simple + orthogonal mechanisms, little policy
  - could be made to work with later requirements
- Field sizes were somewhat preposterous at the outset (32-bit sequence numbers!) so they have aged well

- Algorithm enhancements could be introduced unilaterally
- Some enhancements require both hosts to play (e.g., SACK)
- Only a few need cooperation from both hosts **and** the routers

- Problems remain with SYN flooding and RST attacks
  - Mitigations exist, outright solutions are hard to find

Case study: Mail

- Mail = RFC821 (SMTP) + RFC822 (header format)
  - These evolved out of earlier specifications that sent mail in FTP
- Both are text-based protocols
  - Require TCP, DNS (retrofit)
- SMTP: Interactive
  - Can try out new commands without losing state
  - Extension mechanism retrofit to announce capabilities (1995, RFC1869)
- RFC822: “Batch”
  - Rule: Ignore what you don’t understand
  - Pioneered “free extension” situation
- RFC2821/2: Consolidate 19 years of operational experience
- MIME (1992): retrofit content types and encodings
- Secure Mail (S/MIME and OpenPGP): not so successful

Case study: HTML

- HTML was officially an SGML application
  - Only validated pages should have been used
- Reality: “free extension” to the max
  - Principle: unknown markup is ignored
- Development between 1994 and 1998 was influenced by the “browser wars”
  - Microsoft and Netscape tried to one-up each other on browser features
  - HTML extensions played a major role here (“embrace and extend”)
- Cycle-based development bursts, fuelled by tension between:
  - the competitive urge of companies to outdo each other and
  - the common need for standards for moving forward

The HTML cycle (1)

[based on Tim Berners-Lee]

Experimentation phase:

- HTML standard is open and usable by anyone
  - any engineer, in any company or waiting for a bus can think of new ways to extend HTML, and try them out

Growth phase:

- some of these many ideas are tried out in prototypes or products
  - free extension rule: any unrecognized extensions will be ignored by everything which does not understand them
  - result: dramatic growth in features
- Some of these become product differentiators
  - Now, originators are loath to discuss the technology with the competition (hard to do because of "view source", though).
- Some features die in the market and disappear from the products
- Successful features don't stay product differentiators:
  - soon emulated in some equivalent (though different) feature in competing products

The HTML cycle (2)

Consolidation ("firefighting"?) phase:

- there are now three or four ways of doing the same thing
  - engineers in each company are forced to spend their time writing three of four different versions of the same thing,
  - coping with the software architectural problems which arise from the mix of different models.
- This wastes program size, and confuses users.
- Example: TABLE element
  - multiple extensions were all using the same element name
  - browser had to guess which semantics to render
  - server could never be sure what to send
- Result: Fragmentation, brittleness.

- Fix: develop common specification from the best features
  - And let the cycle begin again…

The end of the HTML cycle

- 1998: W3C was starting to lead the development
- Spec was big enough to **require** some modularity
- CSS, DOM/JavaScript were split off
- New developments (MathML, SVG) could use XML namespaces
  - identify extensions -- no ambiguity
  - Modularity
  - language mixing
- “partial understanding”!

- “When expressing something, use the least powerful language you can.”
  - (cf. "be conservative in what you do"...)

Case study: HTTP

- HTTP 0.9: hack
- HTTP 1.0: uses MIME, RFC822 style text-based
  - Formalized only 1996 (RFC1945) — based on considerable experience
  - Deployed 1.0 then significantly extended by pre-1.1 functions
- HTTP 1.1: addresses connection reuse, caching, “virtual hosts”
  - Formalized 1999 (RFC 2616)
  - Fully compatible to HTTP 1.0 and various deployed pre-1.1 versions
  - Stable! Ubiquitous! Used beyond the traditional Web.

- HTTPng: attempt to redo HTTP in a more well-layered way
  - Much uncertainty, little demonstrable gain
  - Abandoned

Case Study: SIP

355

+30

+50

+...

Timeline: 1996

Initial Internet Drafts:
Session Invitation Protocol (SIP) – M. Handley, E. Schooler
Simple Conference Invitation Protocol (SCIP) – H. Schulzrinne

SIP: Setup + Caps Negotiation

SCIP: Setup + Caps Modify + Terminate

Presentations at 35th IETF, Los Angeles

Merged Draft: SIP -01

Main Features set: TCP/UDP, Forking, Redirection, addrs INVITE,CAPABILITY From: To: Path:

22 Feb 1996 | **4-8 Mar 1996** | **2 Dec 1996**

Timeline: 1997

Draft SIP -02

Formal syntax
CAPABILITY ➔ OPTIONS
Path: ➔ Via:
Ideas for Alternates:

IETF Action: Split SIP into base spec and extensions

Draft SIP -03

SIP URL: sip://jo@…
CONNECTED, BYE, REGISTER
Call-ID: Sequence:
Allow: Expires:

Draft SIP -04

CONNECTED ➔ ACK
UNREGISTER
Sequence: ➔CSeq:
Call-Disposition:
Require:

27 Mar 97 | 31 Jul 97 | 11 Nov 97 | Dec 97

Timeline: 1998

SIP -05

CANCEL
UNREGISTER ➔ ∅
URL sip://jo ➔ sip:jo
Record-Route:
IANA assignments
Security Cons. Sect.

Clarifications & fixes
Cleaning up the spec
Call-ID: MUST
tag parameter

IETF Action: Last Call for Proposed

SIP -06

SIP -07

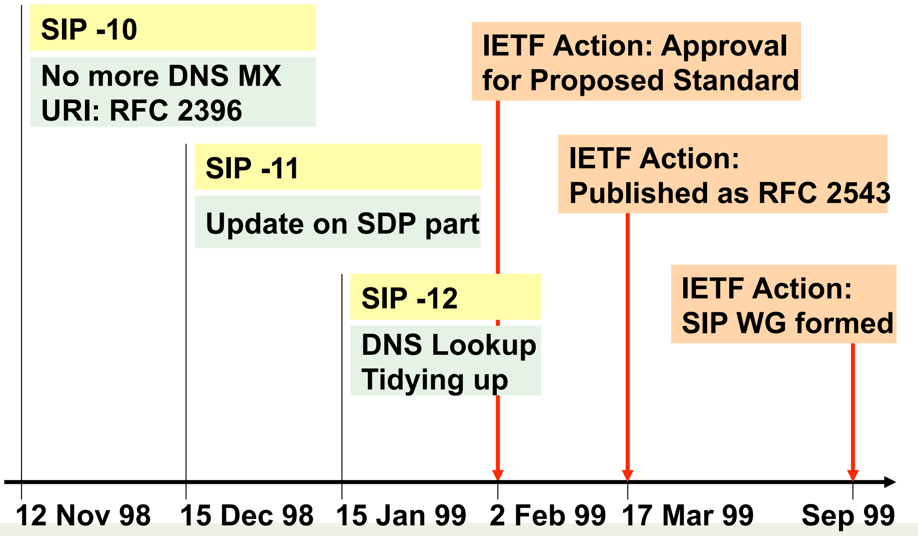
SIP -08

SIP -09

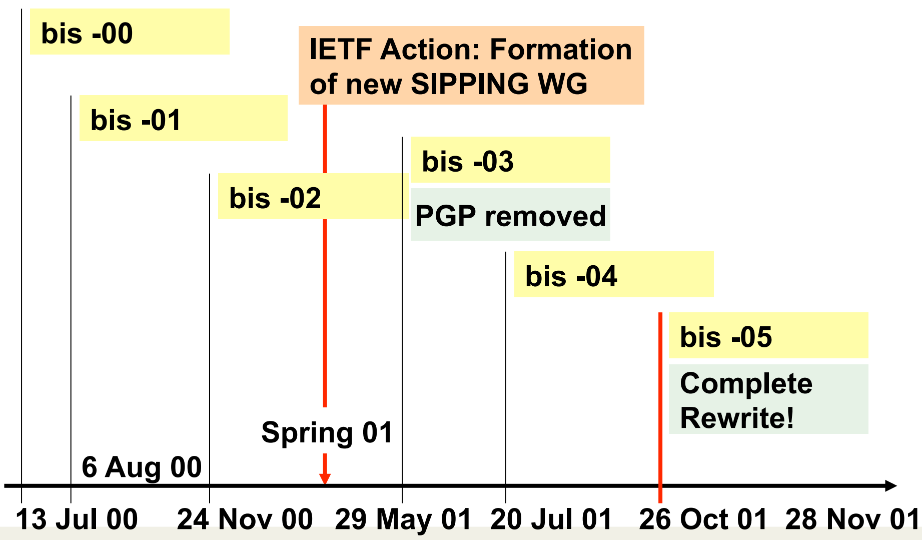
Call Hold SDP

14 May | 17 Jun | 16 Jul | 8 Aug | 18 Sep | 28 Sep

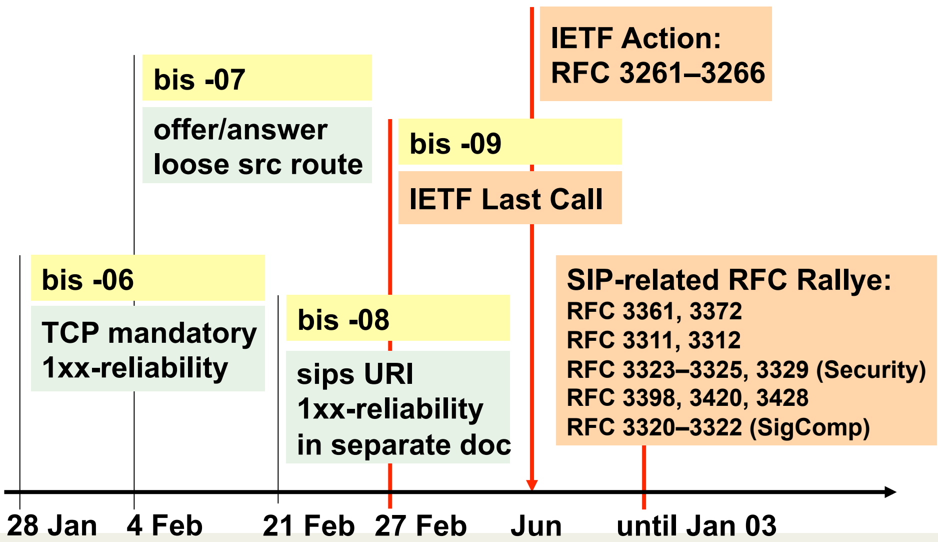
Timeline: 1998/99



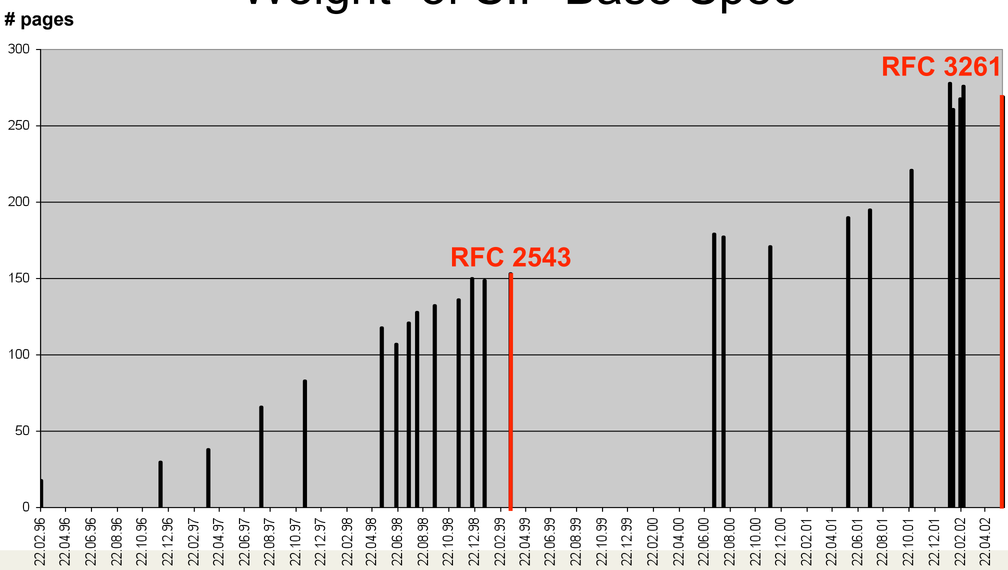
Timeline: RFC2543bis (2000/2001)



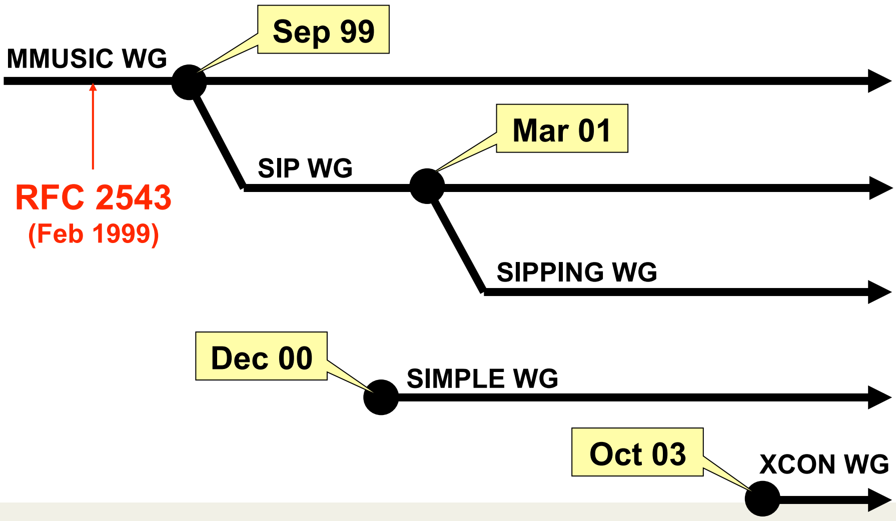
Timeline: RFC2543bis, RFC3261 (2002)



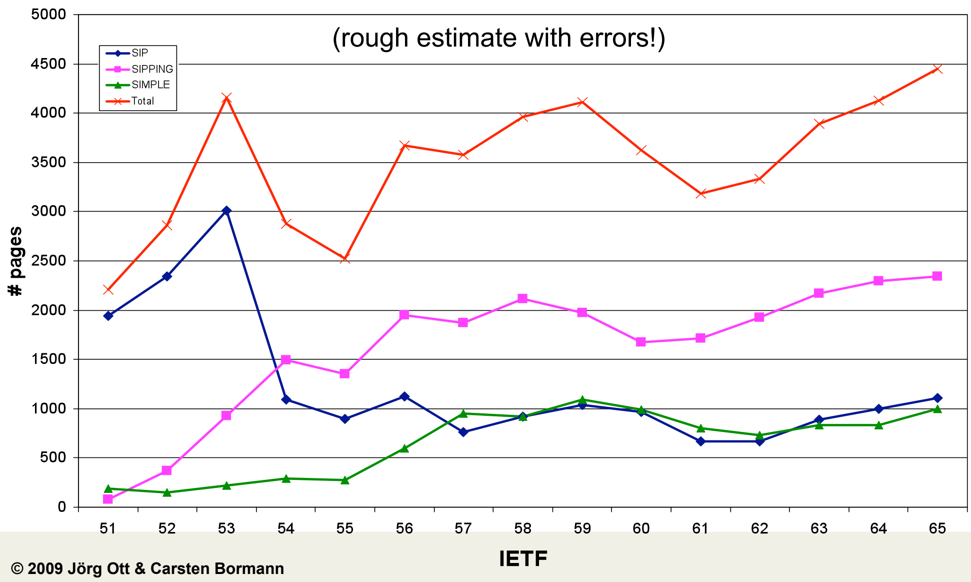
“Weight” of SIP Base Spec



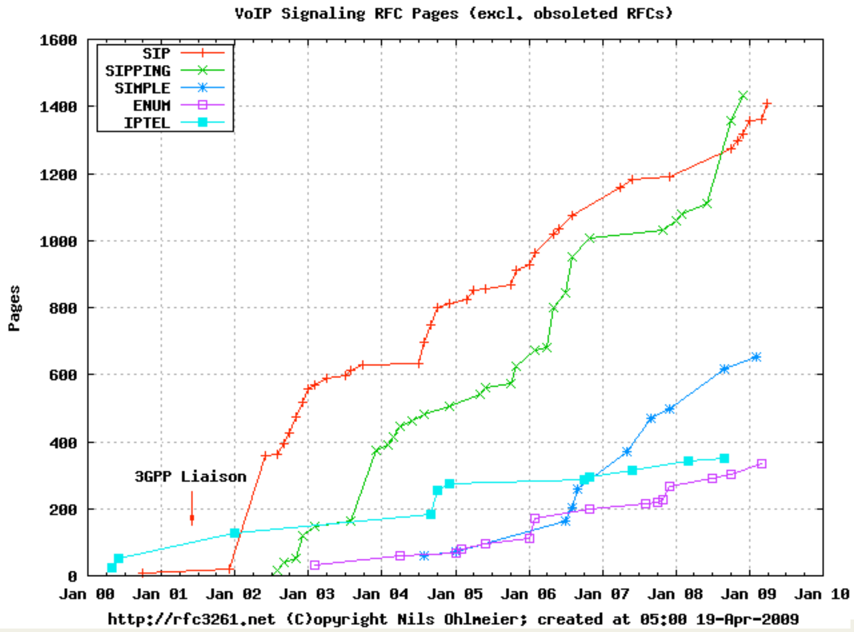
IETF SIP-related Working Groups (1)



"Productivity" (1): Internet Draft Pages



“Productivity” (2): RFC Pages



SIP: The verdict

- Set out with the promise of Simplicity (“Simple Conference Invitation Protocol”)
  - Was meant for conferencing
  - Retargeted for embracing telephony
- Tried to leverage (and extend) an unrelated protocol (HTTP) and a vaguely related protocol (RFC822)
- Protocol Issue: Confusing transport layer and application layer
  - The curse of UDP, fragmentation, forking/multicast, …
- Marred by SDP
  - Another retargeted protocol extended to death (“offer-answer”)
- Interesting case study for evolvebility:
  building-block based extensibility vs. well-defined services
- NAT traversal capabilities add to its appeal

“ng” efforts

- IP: IPv4 ➔ IPv6
  - Motivated by field size issues
  - Convenient time to change not only syntax, but also semantics
  - No interoperability (ships in the night) because of fear of NATs
- HTTP: HTTP 1.1 ➔ HTTPng
  - Grandiose ideas of a “new session layer”
  - Just wasn’t worth it
- SDP: SDP ➔ SDPng
  - XML substrate came too early
- RADIUS: RADIUS ➔ DIAMETER
  - Field size issues again
  - “Fixing” broken protocol semantics

Why “ng” efforts usually don't work

- Market is supplied by market players
- Incumbents are heavily invested (and have debugged) “pg”
- “ng” might exhibit unknown technical (as well as patent!) issues
- Incumbents consider complexity of working with old, overstretched protocol to be a convenient barrier to market entry
- “ng” development is likely to fall victim to:
  - second system syndrome
  - random non-market oriented forces (academics, patent players, architects, …)
- All the while more market-driven features continue to be put into “pg” — even when it hurts

Wholesale replacements do work, if…

- Disruptive technology
  - Market values new economy over features that are oversupplied by “pg”
- Carried forward not by incumbents, but by strong new players
- Concurrence with investment/technology replacement cycle

- GGP ➔ EGP ➔ BGP
  - The underlying structure of the Internet changed
  - There just **had** to be a change at the protocol level
- (PSTN, H.323) ➔ SIP
  - H.323 eclipse was helped tremendously by PER disaster
    - H.323 had no “Henry”, either
  - Bubble helped, too