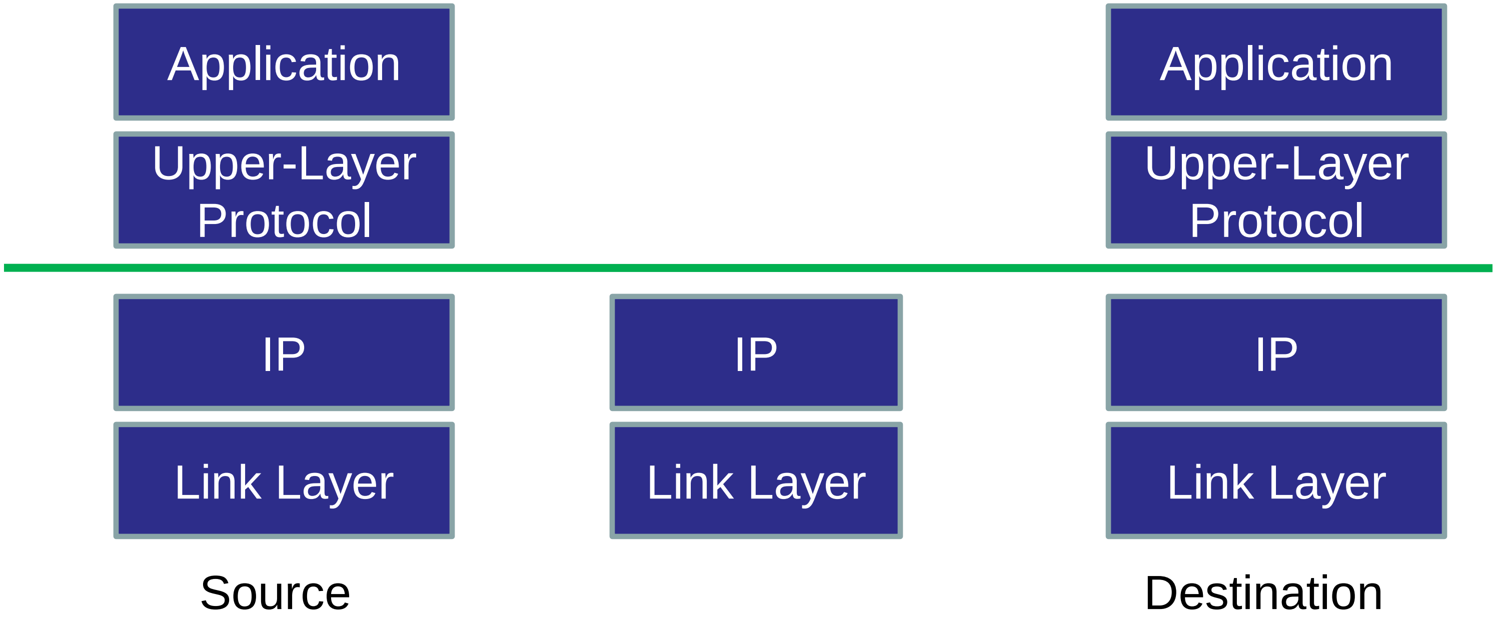
Evolution of the IP Model

draft-iab-ip-model-evolution-01.txt

Dave Thaler
dthaler@microsoft.com

What is the IP Model?

- The model exposed by IP to higher layer protocols and applications



In the beginning…

- IP was published in a series of IENs starting in 1978, then RFC 760 in 1980 and finally RFC 791 in 1981
- However, the model continued to evolve
- Some changes are intentional, some changes happen as a side effect of some other goal

Evolution…

- By 1989, there was already some confusion and so RFC 1122 clarified many things and extended the model
- In 2004, RFC 3819 (“Advice for Internet Subnetwork Designers”) gave advice to L2 designers on things that affect upper layers
- (and various RFCs give advice on other specific topics: RFC 2991, 4903, etc)

But through it all…

Since 1978 many applications and upper-layer protocols evolved around various **additional assumptions**, but they're:

- – not listed in one place
- – not necessarily well-known
- – not necessarily thought about when making changes
- – increasingly, not even true!

Goals of the IAB work

1. Collect assumptions (or, increasingly, “myths”) in one place
2. Document to what extent they are true
3. Provide some guidance to the community

- Most of #1 & #2 were presented in either INTAREA or EXPLISP in Dublin
- In this presentation, we concentrate mostly on #3

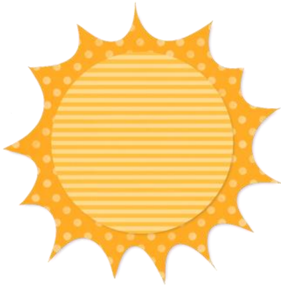
Basic IP (RFC 791) Service Model

- Senders just send to an address, without signaling a priori
- Receivers just listen on an already provisioned address, without signaling a priori
- Packets can be of variable size
- No guarantee of reliability, ordering, or lack of duplication

End-system models (RFC 1122)

- Strong host:
  - Outgoing datagrams MUST be sent on an interface corresponding to the source address
  - Incoming datagrams MUST arrive on an interface corresponding to the destination address or be dropped

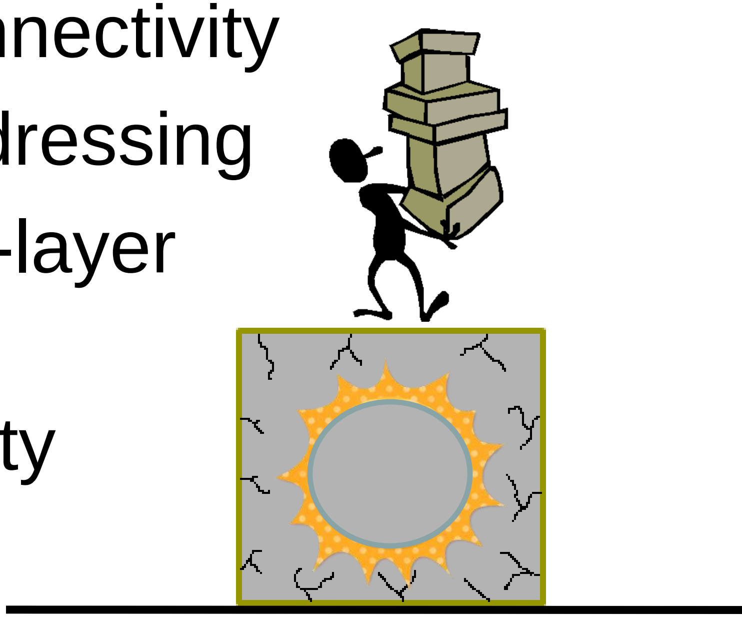
- Weak host:
  - Outgoing datagrams can be sent out any interface
  - Incoming datagrams can arrive on any interface

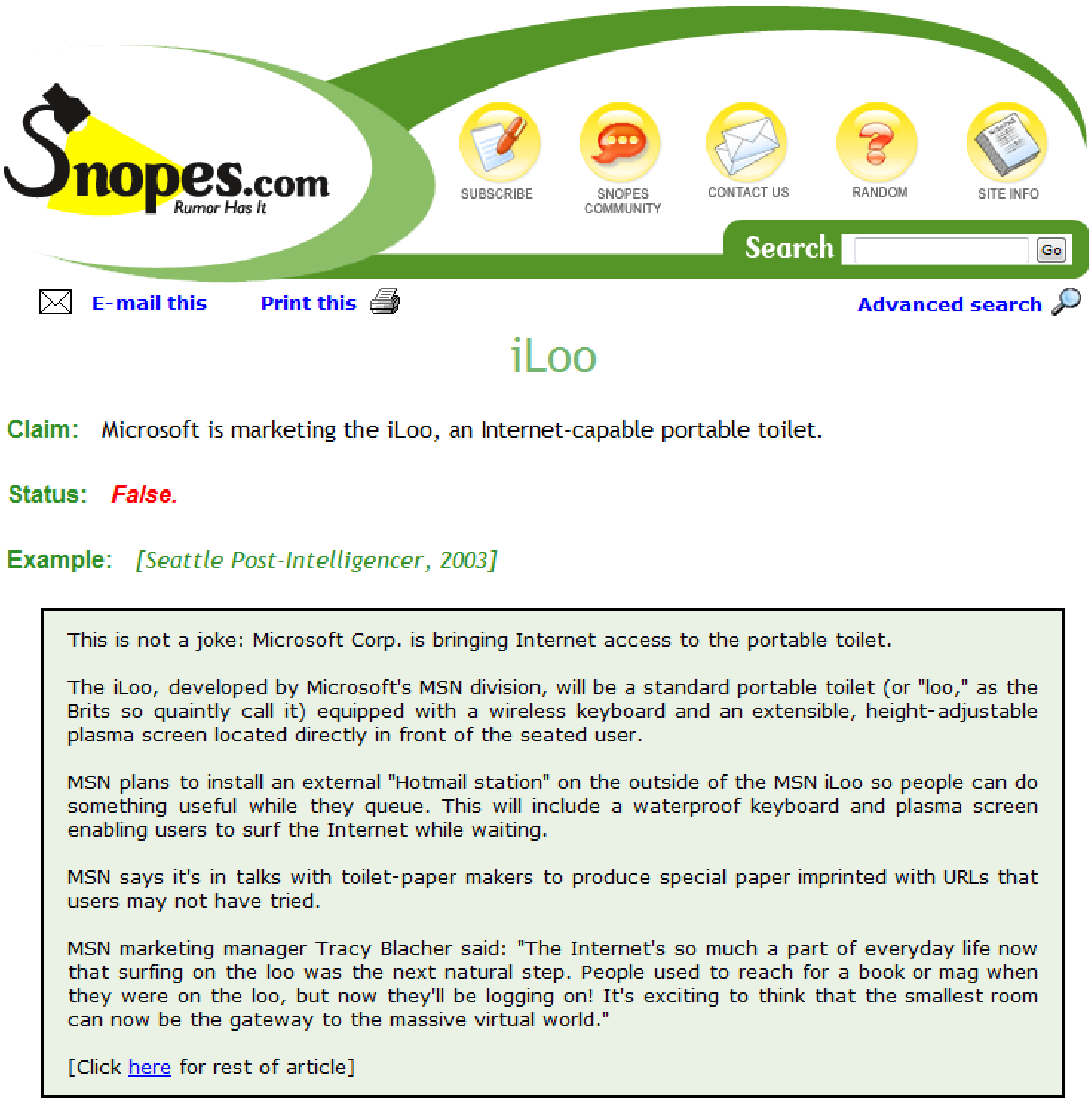


- Note that enabling forwarding results in weak host
- Some OS’s use strong host, some use weak host

But wait… there's more!

- Common application/upper-layer protocol assumptions (or, increasingly, myths)
  - – Assumptions about IP connectivity
  - – Assumptions about IP addressing
  - – Assumptions about upper-layer protocol extensibility
  - – Assumptions about security





iLoo

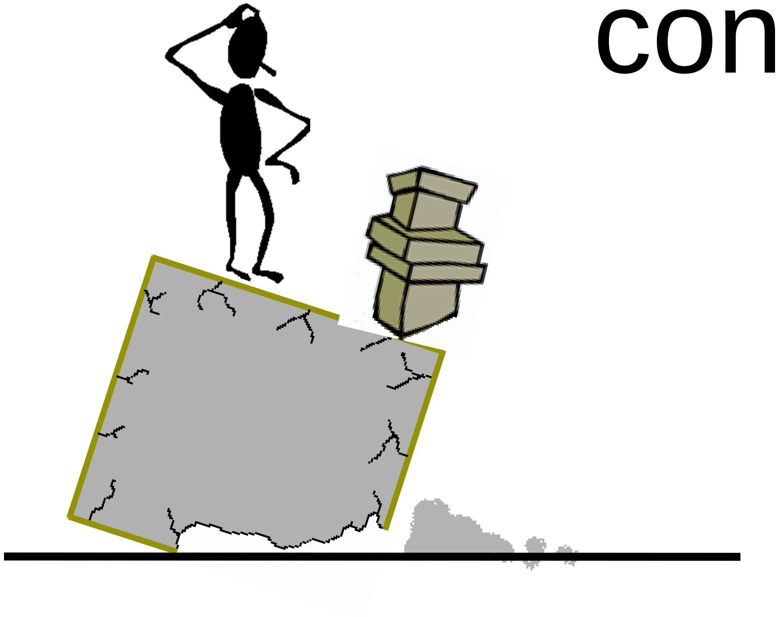
Claim: Microsoft is marketing the iLoo, an Internet-capable portable toilet.

Status: *False.*

Example: *[Seattle Post-Intelligencer, 2003]*

> This is not a joke: Microsoft Corp. is bringing Internet access to the portable toilet.
>
> The iLoo, developed by Microsoft's MSN division, will be a standard portable toilet (or "loo," as the Brits so quaintly call it) equipped with a wireless keyboard and an extensible, height-adjustable plasma screen located directly in front of the seated user.
>
> MSN plans to install an external "Hotmail station" on the outside of the MSN iLoo so people can do something useful while they queue. This will include a waterproof keyboard and plasma screen enabling users to surf the Internet while waiting.
>
> MSN says it's in talks with toilet-paper makers to produce special paper imprinted with URLs that users may not have tried.
>
> MSN marketing manager Tracy Blacher said: "The Internet's so much a part of everyday life now that surfing on the loo was the next natural step. People used to reach for a book or mag when they were on the loo, but now they'll be logging on! It's exciting to think that the smallest room can now be the gateway to the massive virtual world."
>
> [Click here for rest of article]

Assumptions about IP connectivity



Claim: Reachability is Symmetric

- Examples of behavior:
  - Apps do request-response, callbacks, etc

- Status:
  - Much less true with NAT, firewall, 802.11 ad-hoc, satellite, admission control proxies, etc.
  - UDLR was one effort to help restore
  - Request-response *usually* works, but not callbacks

Claim: Reachability is Transitive

- Examples of behavior:
  - Apps do referrals/redirects
- Status:
  - Much less true with NAT, firewall, 802.11 ad-hoc, satellite, etc.

Claim: E2E delay of first packet to a destination is typical

- Examples of behavior:
  - Applications “ping” candidate servers and use the first one to respond

- Status:
  - First packet may have additional latency (e.g. ARP, flow-based routers)
  - MIPv6, PIM-SM, MSDP, some RRG proposals, etc allow deterministic path switching during initial data burst
  - “Choice” of server can hence be highly suboptimal, resulting in longer paths, lower throughput, and higher load on the Internet

Other assumptions (see draft)

- Multicast is supported within a link
- IPv4 broadcast is supported
- Broadcast/multicast is less expensive than replicated unicast
- Reordering is rare
- Loss is rare and probabilistic, not deterministic
- An end-to-end path exists at a single point in time

Discussion

- There are two types of causes of assumption violations:
  - – Effects of **link-layer** technologies
  - – Effects of **network-layer** technologies

Effects of link-layer technologies

- They’re not intentionally trying to break IP
- Defining IP over them “accidentally” creates the problems
- RFC 3819 gives advice to L2 designers to minimize such effects
- Guidance: *IP-over-Foo definition should compensate for the rest as much as possible*
  - Asymmetry: e.g. UDLR [RFC3077]
  - NBMA: e.g. IPoNBMA [RFC2491]
  - Transitivity: ???

Effects of network-layer technologies

- Reachability is good! (hey, we’re the IETF…)
- But sometimes we don’t *want* to be reachable by everyone
  - – RFC 4948 (IAB Unwanted Traffic workshop)
  - – IPsec, for example, can restrict reachability as an integral part of the current IP model
- Blocking communication to/from “unauthorized” parties is legitimate and already a part of the IP model

Guidance 1/2

When reachability is affected for reasons beyond simply restricting to only “authorized” parties, the IETF should attempt to avoid or solve

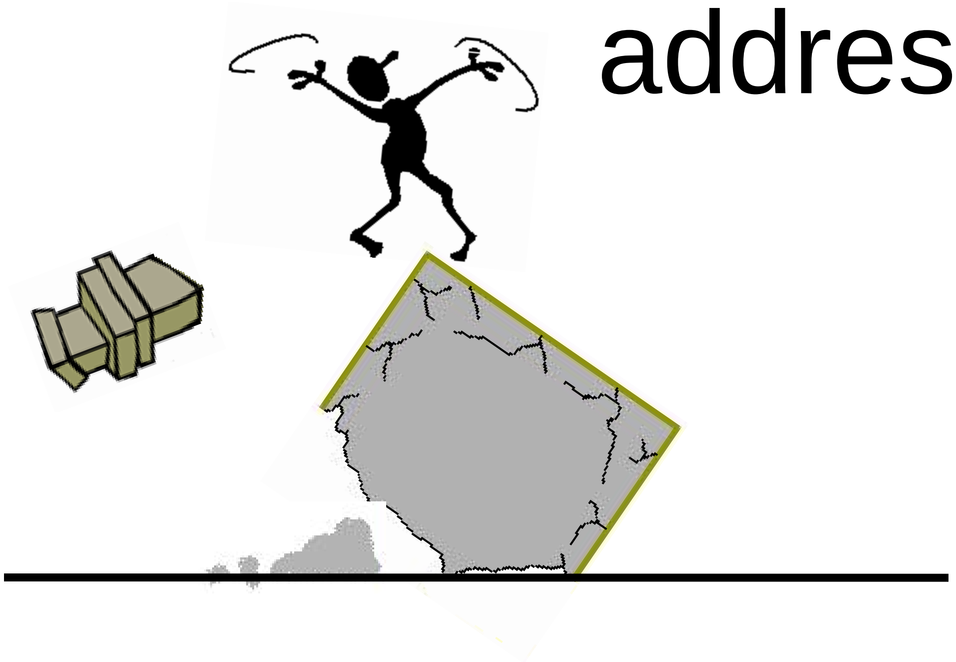
- Diagnostics are important
- Design new/improved solutions

Guidance 2/2

When defining a protocol, be liberal in what effects you accept, and conservative in what effects you cause

- Upper-layers should
  - Avoid such assumptions where practical
  - Consider them when doing applicability statements
- Lower-layers should
  - Avoid making them less true in general
  - Consider effects on upper layers

Assumptions about IP addressing



Claim: Addresses are stable over long periods of time

- Examples of behavior:
  - Apps resolve names to addresses and cache them without any notion of lifetime
  - Name resolution APIs don’t even provide the lifetime
- Status:
  - – Much less true with DHCP, roaming, etc.
  - – PMIP trying to restore within a local network
  - – MIP, HIP, etc trying to restore to some extent by adding an additional address that is stable

Claim: A host has only one address and one interface

- Examples of behavior:
  - – Apps resolve name to address and just use the first one returned
  - – Some apps use address to identify users/machines
  - – Some DHCP options are defined as machine-wide
- Status:
  - Much less true with multihoming, dual-stack nodes, VPNs, etc.
  - MIP, HIP, etc trying to restore to some extent

Claim: An "address" used by an application is the same as the "address" used for routing

- A.k.a. “ID == Locator”

- Examples of behavior:
  - Apps make assumptions about locality (e.g., same subnet) by comparing addresses
  - Server-selection apps/protocols make assumptions about locality by comparing source address against configured ranges
  - Apps use raw sockets to read/write packet headers

- Status:
  - Not true with tunneling, most ID-locator split schemes, etc.
    - Some ID-locator split schemes only break it in the core of the Internet

Other assumptions (see draft)

- A non-mcast/bcast address identifies 1 host over a long period of time
- “subnet” <= “link”
- Selecting a local address selects the interface
- Every address is part of an on-link subnet

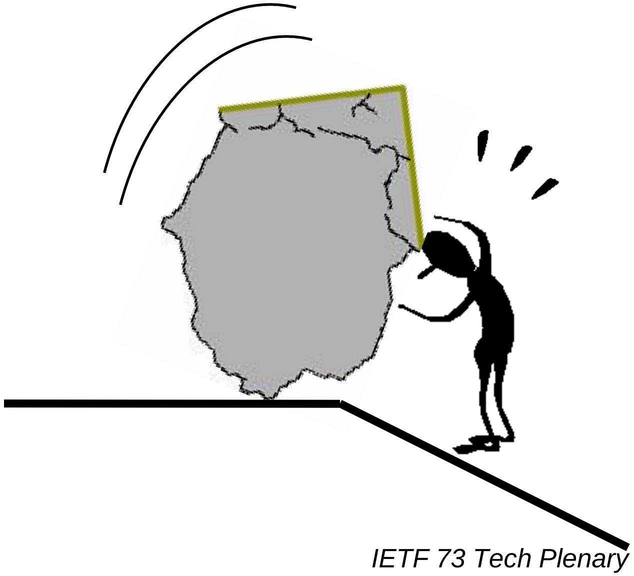
Discussion

- From “Architectural Principles of the Internet” [RFC1958], section 4.1:
  - “In general, user applications should use names rather than addresses.”
- Today:
  - Many APIs unnecessarily expose addresses to applications
  - Some protocols/apps can only carry addresses rather than names (instead or in addition)

Guidance

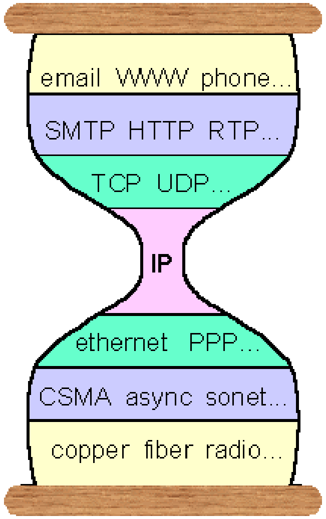
- *Anything already dependent on a naming system should avoid using addresses*
  - API providers can do a better job here (e.g. connect-by-name)
  - Many apps/protocols probably could too
  - This also eases IPv6 transition

Assumptions about upper-layer extensibility

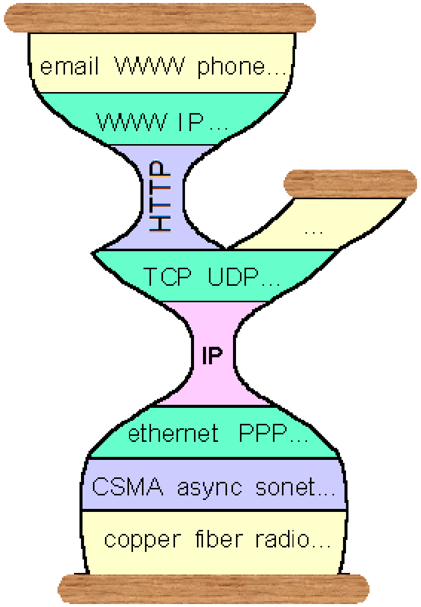


Claim: New transport-layer protocols can work across the Internet

- Examples of behavior:
  - SCTP, DCCP
  - Raw sockets



- Status:
  - NATs/firewalls only allow UDP/
  - Some only allow HTTP



Claim: If one stream to a destination can get through then so can another

- Examples of behavior:
  – Open many connections to get throughput
  – Open separate control vs data channels

- Status:
  – Firewalls may block specific ports
  – Middleboxes may run out of per-connection state

Discussion

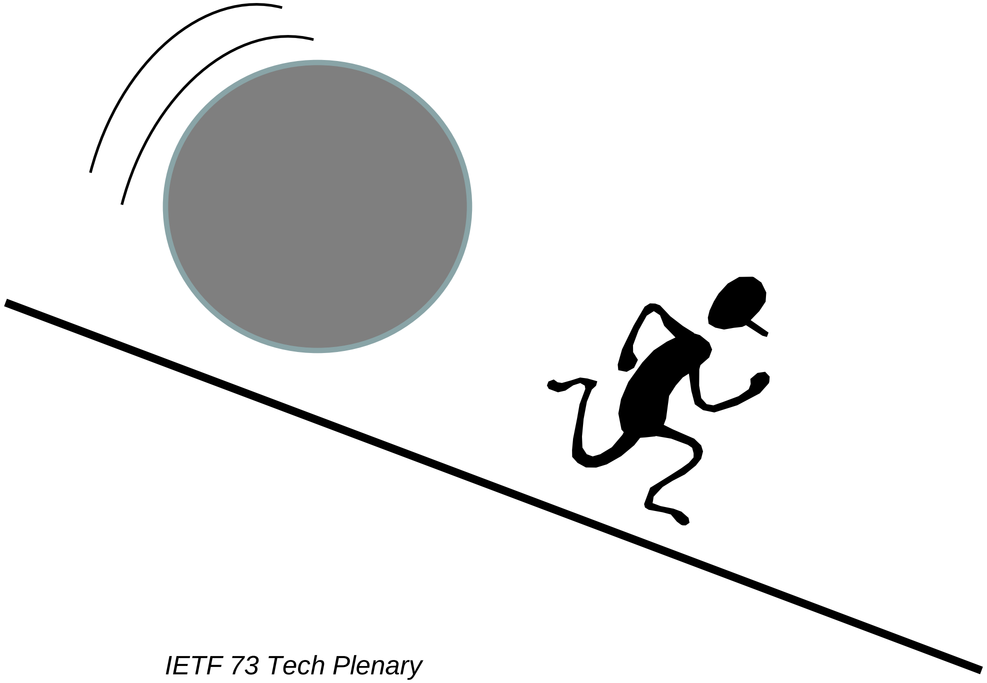
- Original Internet architecture requirements included Service Generality
  - [Dave Clark, “New Arch: Future Generation Internet Architecture”]
  - "This goal was to support the widest possible range of applications, by supporting a variety of types of service at the transport level. […]”
- Issues today are either for security or as a side effect of something else (e.g. address shortage)

Guidance (same as earlier)

1. *For reasons other than restricting reachability to only “authorized” parties, the IETF should attempt to avoid or solve*

2. *Be liberal in what effects you accept, and conservative in what effects you cause*

Assumptions about security



Assumptions (see draft)

- Packets are unmodified in transit
- Packets are private
- Source addresses are not forged

Discussion

- See RFC 3552 (“Guidelines for Writing RFC Text on Security Considerations”)
- Changes to other assumptions might have security impact
  - E.g. app binds to IP of “trusted” interface assuming it will only get traffic from that interface
- Great care should be taken when making an assumption less true
- Upper layers should carefully consider the impact if basing security on any such assumption
- Of course, many assumption violations were done for security, at expense of breaking some apps

Conclusions

- Any changes to assumptions break some apps
  - Ossification of the Internet means changes cause pain
  - Changes must be done with extreme care
- Adding *opt-in* functionality is generally safe
  - But fewer apps use

- Network layer or below: consider effect on upper layers when making changes
- Transport layer or above: avoid assumptions where possible, consider them when doing requirements and applicability statements

Discussion

Be liberal in what effects you accept, and conservative in what effects you cause

Questions?