Strategic Choices in Designing the Internet

Suzhou Institute for Advanced Study of USTC

Michael J. O'Donnell
The University of Chicago
Assistant: Xiang Sen, USTC

(12 June 2004; revised 9 July 2004)

- Instructor: Michael J. O'Donnell
  `odonnell@cs.uchicago.edu`

- Assistant: Xiang Sen
  `xiangsen@ustc.edu`

- Web page:
  `http://221.6.69.11/~od/Strategic_Internet/`

- Schedule:

  - 09:00-12:00 lectures—15 minute break around 10:30

  - 12:00-14:00 lunch, free discussion, work on exercises

  - 14:00-16:00 questions, lectures

- Required exercises: 1 set each day; free discussion; write your answer in English, email to Xiang Sen next day.

- Optional advanced exercises: Select from the list; work on your own; email to Xiang Sen by 19 July.

- 80-89: good job on all required exercises.

- 90-100: also very good job on at least one advanced exercise.

Nature of course

- Consider a logical sequence of design choices leading to the Internet.

- Evaluate the reason for each choice, and the impact of each choice.

- Preview future choices.

Why study design choices?

- Help design the next Internet
- Be a good netizen
- Anticipate future developments
- Use design principles in applications
- Learn to read primary sources (RFCs)

Modern method, ancient wisdom

"Designing application-level protocols is really no different than designing core network protocols."
—Peterson and Davie

"The core and the surface
Are essentially the same."
—Lao

Design layers come from scope of agreements

- Choices that affect more agents are more fundamental.

- Choices that constrain other choices are more fundamental.

If we're lucky, these two dimensions are reasonably compatible.

Layers of Internet design

<table>
<tr><td>Routing</td><td>Capacity management</td><td>Address management</td><td>Naming (relocation)</td></tr>
<tr><td colspan="4">Addressing</td></tr>
<tr><td colspan="4">Delivery (forwarding)</td></tr>
</table>

Internet protocols in design layers

- Delivery/forwarding: IP
  - Addressing: IP, UDP, TCP
    - Routing: ARP, ICMP, RIP, OSPF, CIDR, IGP, BGP
    - Capacity management: TCP, ICMP
    - Address management: ARP
  - Naming/relocation: DNS

Choices to study

1. Stateless message forwarded by stateless router

2. Two-level addresses

3. End-to-end acknowledgment, not hop-by-hop

4. End-to-end congestion control

5. Best-effort, no reservation

6. DNS: name service, no handle service

7. Future authentication: chain of trust?

Unusual evaluations of choices

Evaluate choices by principles, results:

- good
- bad
- neutral
- illusory: not really a choice

RFC791
Basic purpose of IP

1.1 Motivation

The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses.

RFC791
Limitation of IP

1.2. Scope

The internet protocol is specifically limited in scope to provide the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. There are no mechanisms to augment end-to-end data reliability, flow control, sequencing, ...

RFC791
Addressing is fundamental

1.4. Operation

The internet protocol implements two basic functions: addressing and fragmentation.

The internet modules use the addresses carried in the internet header to transmit internet datagrams toward their destinations. The selection of a path for transmission is called routing.

RFC791
Protocol: common rules

1.4. Operation

...

The model of operation is that an internet module resides in each host engaged in internet communication and in each gateway that interconnects networks. These modules share common rules for interpreting address fields.

RFC791
Protocol relationships

2.1. Relation to Other Protocols

```
+------+ +-----+ +-----+     +-----+
|Telnet| | FTP | | TFTP| ... | ... |
+------+ +-----+ +-----+     +-----+
      |   |         |           |
     +-----+     +-----+     +-----+
     | TCP |     | UDP | ... | ... |
     +-----+     +-----+     +-----+
        |           |           |
     +--------------------------+----+
     |    Internet Protocol & ICMP   |
     +--------------------------+----+
                    |
      +---------------------------+
      |   Local Network Protocol  |
      +---------------------------+

       Protocol Relationships

           Figure 1.
```

OSI network model vs. Internet

OSI	Internet	
Application	Telnet, FTP, ...	
Presentation		
Session	(open/close)	
Transport	TCP	UDP
Network	IP	
Data link	Local Network	
Physical	Protocol	

RFC791
Protocol relationships, my version

```
+------+ +-----+ +-----+     +-----+
|Telnet| | FTP | | TFTP| ... | ... |
+------+ +-----+ +-----+     +-----+
      |   |         |           |
     +-----+     +-----+     +-----+
     | TCP |     | UDP | ... | ... |
     +-----+     +-----+     +-----+
        |           |           |
     +---------------------------+----+
     |    Internet Protocol & ICMP    |
     +---------------------------+----+
       |         |        |         |
+--------+ +----------+ +---+     +---+
|Ethernet| |Token Ring| |ATM| ... |...|
+--------+ +----------+ +---+     +---+
```

From P&D Figure 1.14

RFC791

Packet traversing network and protocols

2.2. Model of Operation

```
Application                                  Application
Program                                          Program
     \                                             /
 Internet Module     Internet Module    Internet Module
       \                /       \              /
      LNI-1          LNI-1      LNI-2         LNI-2
         \           /             \          /
        Local Network 1           Local Network 2
```

Transmission Path

Figure 2

RFC791
Packet traversing network and protocols, my version

2.2. Model of Operation

```
Stuff                                               Stuff
     \                                               /
 Internet Module      Internet Module    Internet Module
       \                 /       \              /
      Stuffffffffffffffff       Stufffffffffffffff
```

Violations of layering

- PPP over Ethernet
- IP tunnelling (over IP), mobile IP
- NAT—port incorporated into address
- Ethernet over IP
  (distributed virtual machines,
  Prescience Lab)

RFC791
Names, addresses, routes

2.3. Function Description

...

Addressing

A distinction is made between names, addresses, and routes [4]. A name indicates what we seek. An address indicates where it is. A route indicates how to get there. The internet protocol deals primarily with addresses. It is the task of higher level (i.e., host-to-host or application) protocols to make the mapping from names to addresses.

RFC791
Addresses of what?

Care must be taken in mapping internet addresses to local net addresses; a single physical host must be able to act as if it were several distinct hosts to the extent of using several distinct internet addresses. Some hosts will also have several physical interfaces (multi-homing).

That is, provision must be made for a host to have several physical interfaces to the network with each having several logical internet addresses.

RFC791
Datagram Format

3.1. Internet Header Format

Version: 4 bits	4 bits
IH[eader]L[nth]: 4 bits	
Type of Service: 8 bits	4 bits
	Flow label: 24 bits
Total Length: 16 bits	16 bits
Identification: 16 bits	
Flags: 3 bits	
Fragment Offset: 13 bits	
Time to Live: 8 bits	8 bits
Protocol: 8 bits	8 bits
Header Checksum: 16 bits	
Source Address: 32 bits	128 bits
Destination Address: 32 bits	128 bits
Options: variable	

IP fowarding algorithm
RFC950

```
IF ip_net_number(dg.ip_dest) = ip_net_number(my_ip_addr)
     THEN
         send_dg_locally(dg, dg.ip_dest)
     ELSE
         send_dg_locally(dg,
                   gateway_to(ip_net_number(dg.ip_dest)))
```

IP forwarding algorithm in textbooks

- **if** *address* matches *here* **then** keep

- **else** forward to better neighbor
  - **if** address matches neighbor A **then** deliver to A
  - **else** forward by route table

IP forwarding essentials
my version

Choice: stateless forwarding

- **if** *address* matches *here* **then** keep
- **else** forward to better neighbor

Choice by omission

Key choices in forwarding algorithm:

- *address* doesn't change
- router state doesn't change
- no control messages

Why is forwarding fundamental?

Principle: **design infrastructure bottom-up.**

1. Multihop network is a system of forwarding routers (unavoidable).

2. Try simplest forwarding algorithm that provides value.

Why stateless forwarding?

Principle: **in distributed system, keep information local.**

Principle: **for robust system, avoid dependence on previous correctness.**

- Use of state creates dependence on previous calculation.

Why no control messages?

Principle: **minimize overhead of critical resource.**

- Purpose of network is to provide communication resource to applications.
- Avoid messages that are not critical to applications.

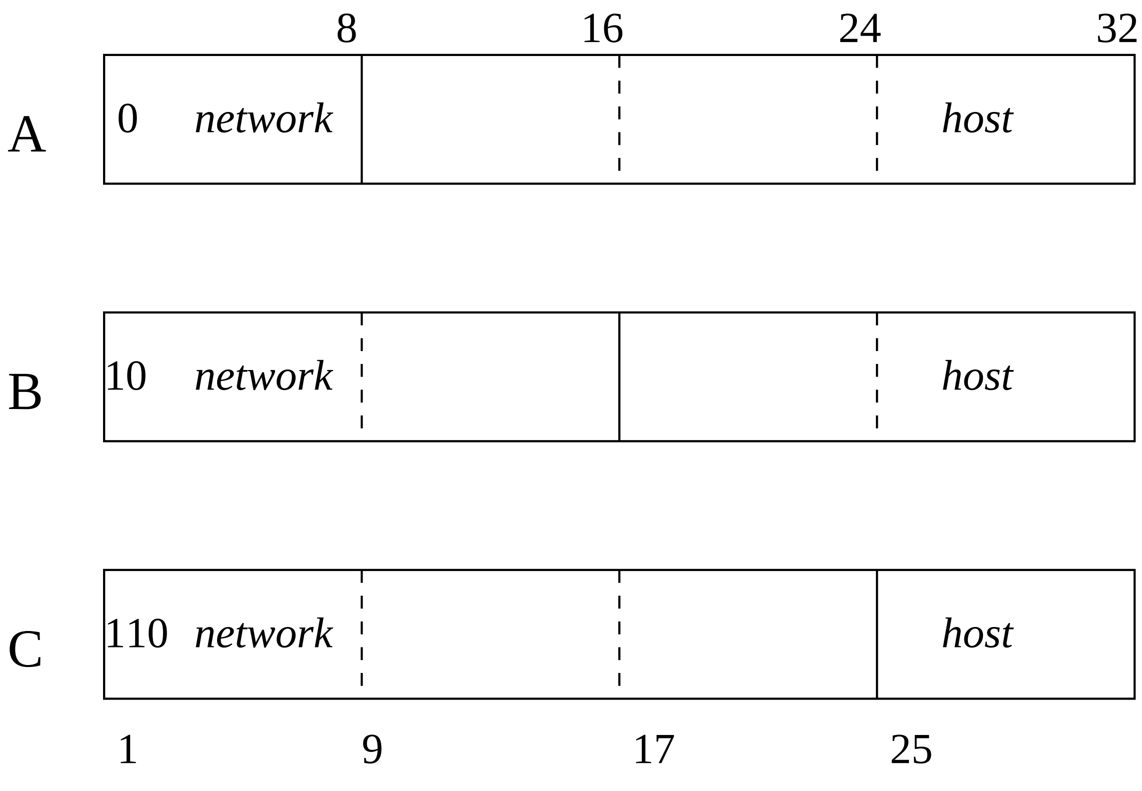
Absolute address vs. directions

History: *UUCP* used directions naming all intermediate hosts: `host1!host2!...!hostn`

- Port to new sender?
  - address: just copy
  - directions: must translate
- Translate directions through net?
  - Tricky—reversibility, cycles, long routes.
  - Can't send directions out-of-band.
- Look up directions from address?
  - Need directions to the directory.

Official definition of IP address

Choice: Two-level address *(network/host)*



Choice of two-level address is illusion

History:

1. 1980–81: RFC 760, 791 2-level addresses
2. 1984–85: RFC 917, 940, 950 subnets
3. 1992–93: RFC 1338, 1519 CIDR

Subnetting/CIDR may be practiced locally, transparently to outsiders

- Subnet is virtual host in larger net.

RFC950
locality of subnetting

The third approach is to explicitly support subnets. This does have a disadvantage, in that it is a modification of the Internet Protocol, and thus requires changes to IP implementations already in use (if these implementations are to be used on a subnetted network). However, these changes are relatively minor, and once made, yield a simple and efficient solution to the problem. Also, the approach avoids any changes that would be incompatible with existing hosts on non-subnetted networks.

RFC950
incremental subnetting

When appropriate design choices are made, it is possible for hosts which believe they are on a non-subnetted network to be used on a subnetted one.

Real choice: aggregated addresses.

- Aggregation allows compact route tables.

- Mostly a consequence of forwarding algorithm.

CIDR routing tables

Net prefix	Network prefix as bits	Nexthop
C4.5E.2.0/23	11000100.01011110.0000001	A
C4.5E.4.0/22	11000100.01011110.000001	B
C4.5E.C0.0/19	11000100.01011110.110	C
C4.5E.40.0/18	11000100.01011110.01	D
C4.4C.0.0/14	11000100.010011	E
C0.0.0.0/2	11	F
80.0.0.0/1	1	G

Table 4.17 (Exercise 46, Chapter 4) P&D

Routing protocols

Forwarding: use of routes.

Routing: discovery of routes.

- ARP: query and announce single hops
- IGPs: query and announce internal routes
- BGP: query and announce distributed routes

ARP queries by host

ARP is algorithmically trivial, but deals with problems of distributed data.

- Local cache of <IP, LANaddress> pairs

- Store only IPs
  - used by this host, or
  - targetting this host

- Broadcast query for unknown IP

- Monitor all broadcast responses

- Delete entry after 15 minutes

END-TO-END ARGUMENTS IN SYSTEM DESIGN

J.H. Saltzer, D.P. Reed and D.D. Clark
M.I.T. Laboratory for Computer Science

This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level.

End-to-End Arguments
Failure at low level

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)

Principle: **Many qualities at lower level affect only performance at higher level.**

End-to-End Arguments
acknowledgment in file transfer

The extra effort expended in the communication system to provide a guarantee of reliable data transmission is only reducing the frequency of retries by the file transfer application.

End-to-End Arguments
disastrous example

A too-real example

[Many software source files lost at MIT. Programmers depended on reliable network links. Gateway computer corrupted the files.]

No hop-by-hop acknowledgment

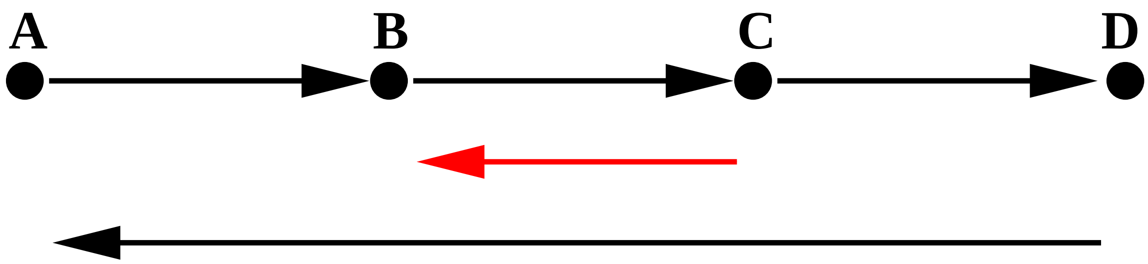
Choice: Don't require intermediate acknowledgment

Principle: **Many qualities at lower level affect only performance at higher level.**

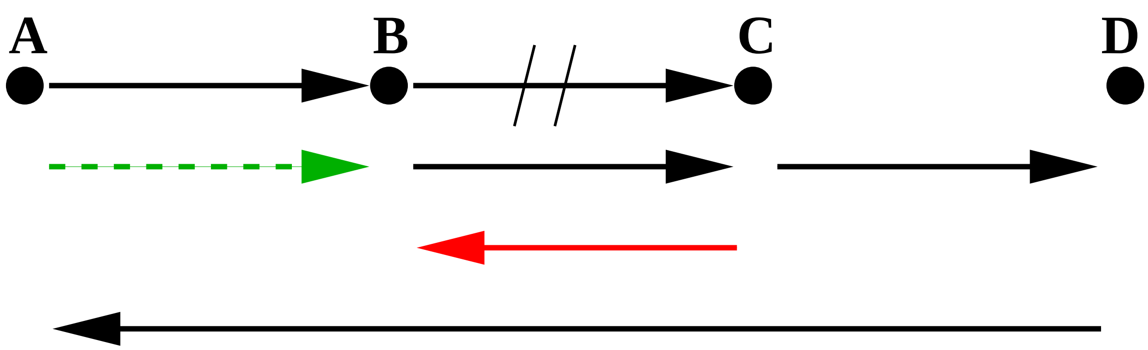
Intermediate acknowledgment

- doesn't provide reliability.
- seldom helps performance

Intermediate acknowledgment

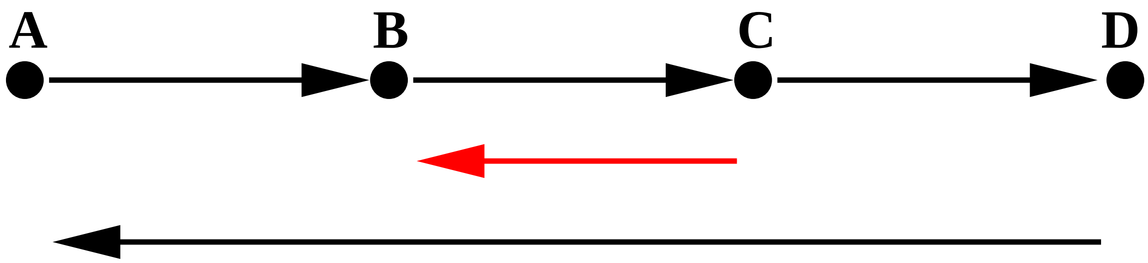


Successful delivery

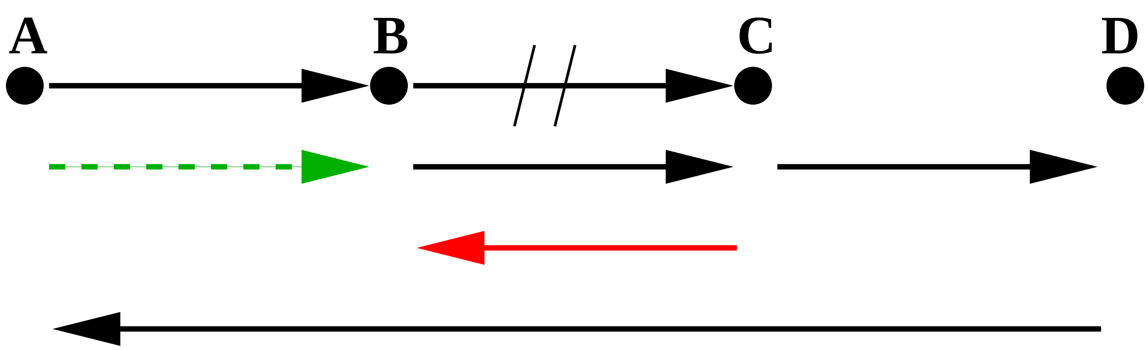


Failure in **B** segment

Intermediate acknowledgment



Successful delivery



Failure in **B** segment

Intermediate ack is cheaper if:

$$RTT(BC) \cdot \textit{buffer} + (1 - \Pr(\textit{fail})) \cdot \text{cost}(CB) < \Pr(\textit{fail}) \cdot \text{cost}(AB)$$

Reasoning about intermediate ack

- Path cost is roughly proportional to length
- Typical Pr(*fail*) is about 0.01
- Typical AB path is 1–30 hops

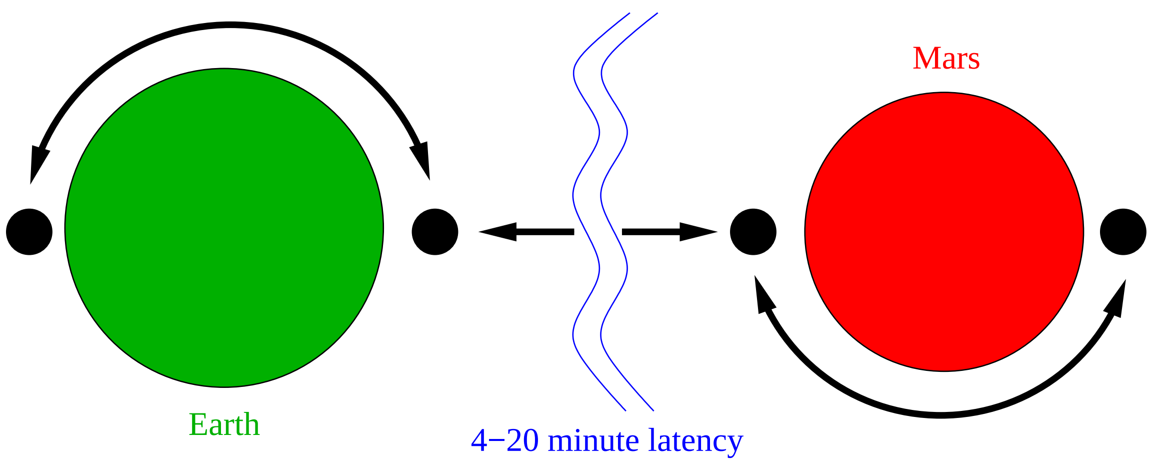
So expected cost of 1-hop ack is usually more than expected cost of retransmission from source.

Where are the ends?

Failure might not be in the network itself, but in the sending or receiving host, at any level of application or OS.

Example: InterPlanetary IP (IPN)

Where would intermediate acks help the most?



End-to-End Arguments
lower level provides performance, not correctness

The amount of effort to put into reliability measures within the data communication system is seen to be an engineering trade-off based on performance, rather than a requirement for correctness.

End-to-End Arguments
more examples

- Acknowledgment/checksum—structurally the same
- Secure transmission of data
- Duplicate message suppression
- Guaranteeing FIFO message delivery
- Transaction management

End-to-End Arguments
principles are not rules:
where are the endpoints?

The end-to-end argument is not an absolute rule, but rather a guideline that helps in application and protocol design analysis; one must use some care to identify the end points to which the argument should be applied.

Example: real-time audio vs. audio file transfer

Active Networking and End-To-End Arguments
Comment by David P. Reed, Jerome H. Saltzer, and David D. Clark

active networking ... comprises attaching programs to data packets, with the intent that those programs be executed at points within the network. ... The question being raised is whether or not this idea directly violates an end-to-end argument.

End-to-End Arguments
modularization of authority

End-to-end arguments address design more than implementation and implementation more than execution—that is, they suggest who should provide the code, not which box it should run on. ... Programmability in a lower layer ... defer[s] design choices upwards in the layering ... and later in time, even though the resulting functions may actually take place deep inside the network.

End-to-End Arguments
design infrastructure bottom-up

A lower layer of a system should support the widest possible variety of services and functions, so as to permit applications that cannot be anticipated.

Principle: **the most important uses of a tool are the ones that haven't been invented yet.**

Example: FTP $\hookrightarrow$ Gopher $\hookrightarrow$ WWW

End-to-End Arguments
design infrastructure bottom-up

- Higher-level layers ... are free to ... organize lower-level network resources to achieve application-specific design goals efficiently. (application autonomy)

- Lower-level layers ... should provide only resources of broad utility across applications, while providing ...
effective sharing of resources and resolution of resource conflicts. (network transparency)

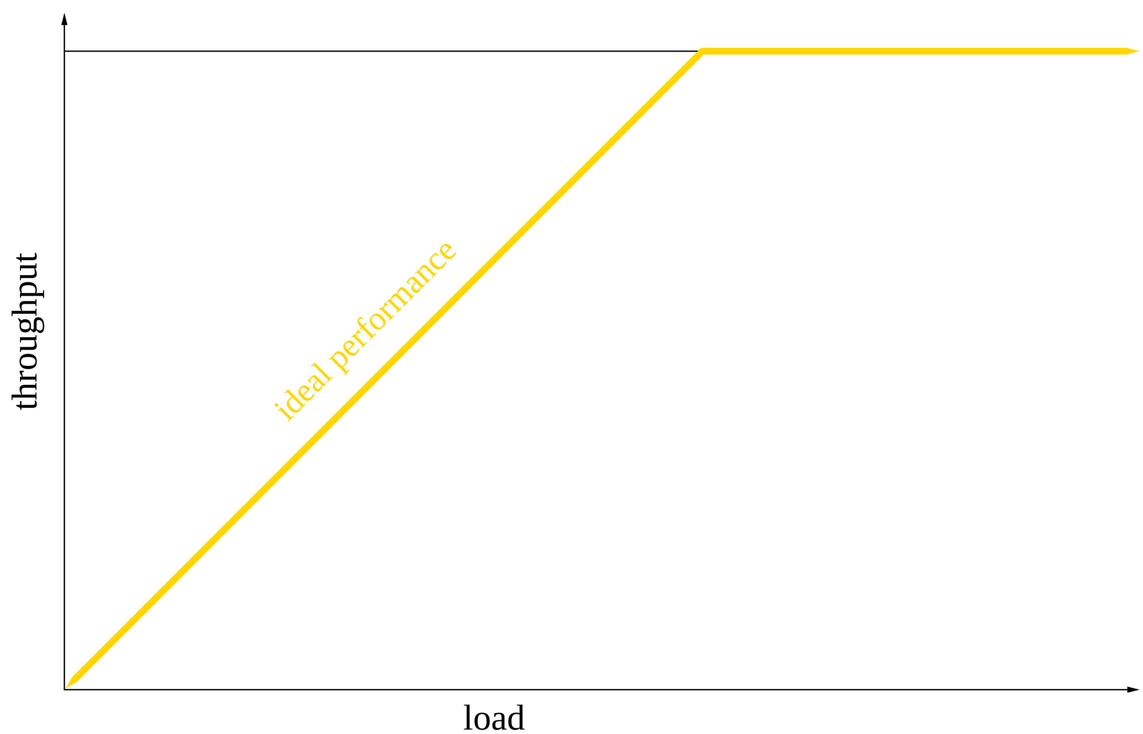
Congestion control

Choice: End-to-end congestion control in TCP

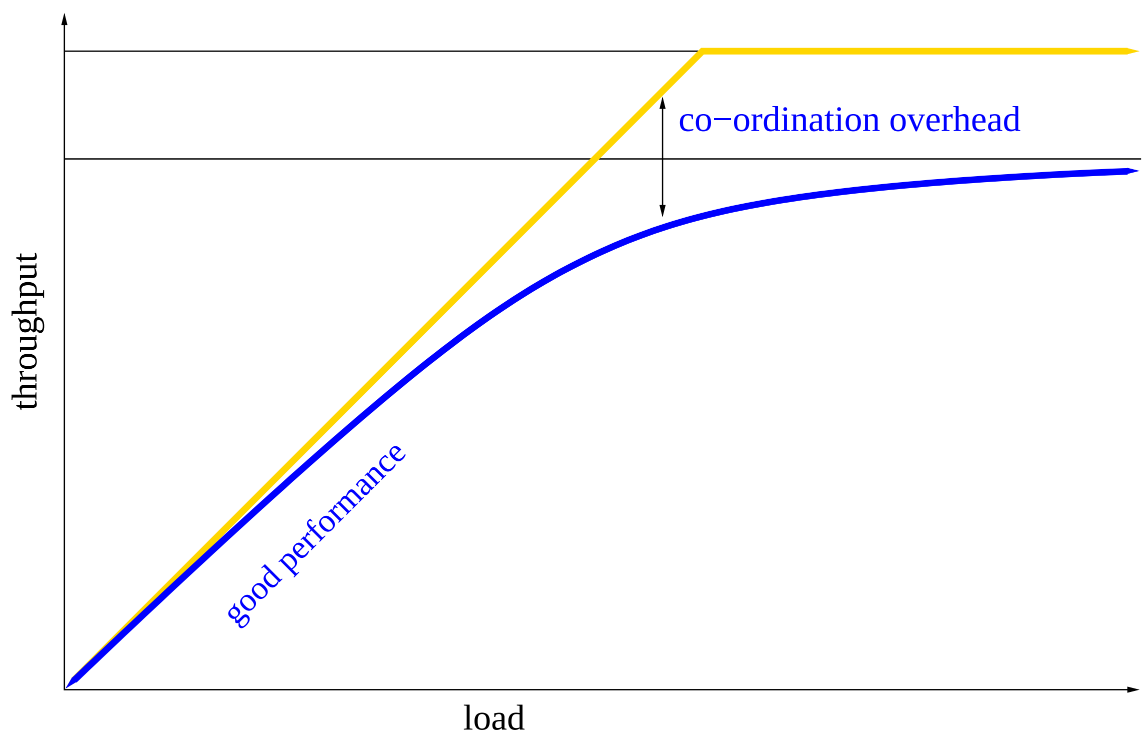
History:

1. 1980–81: RFC 761, 793, RCP
2. c. 1987: Internet congestion collapse
3. 1988: Van Jacobson SIGCOMM, congestion control in TCP
4. 1997–99: RFC 2001, 2581, TCP congestion control

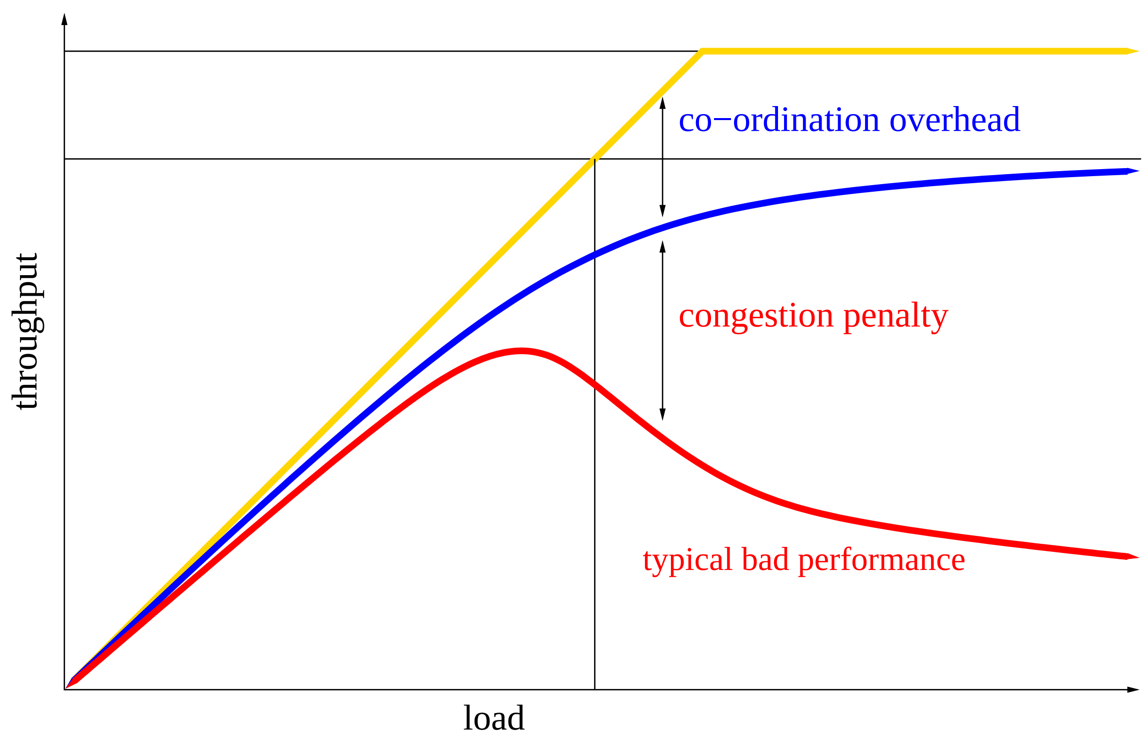
Throughput vs. load



Throughput vs. load



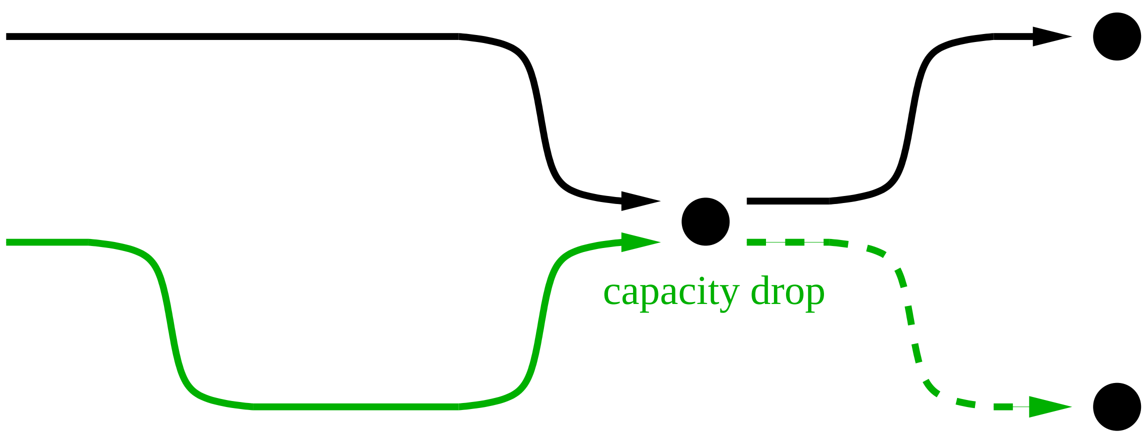
Throughput vs. load



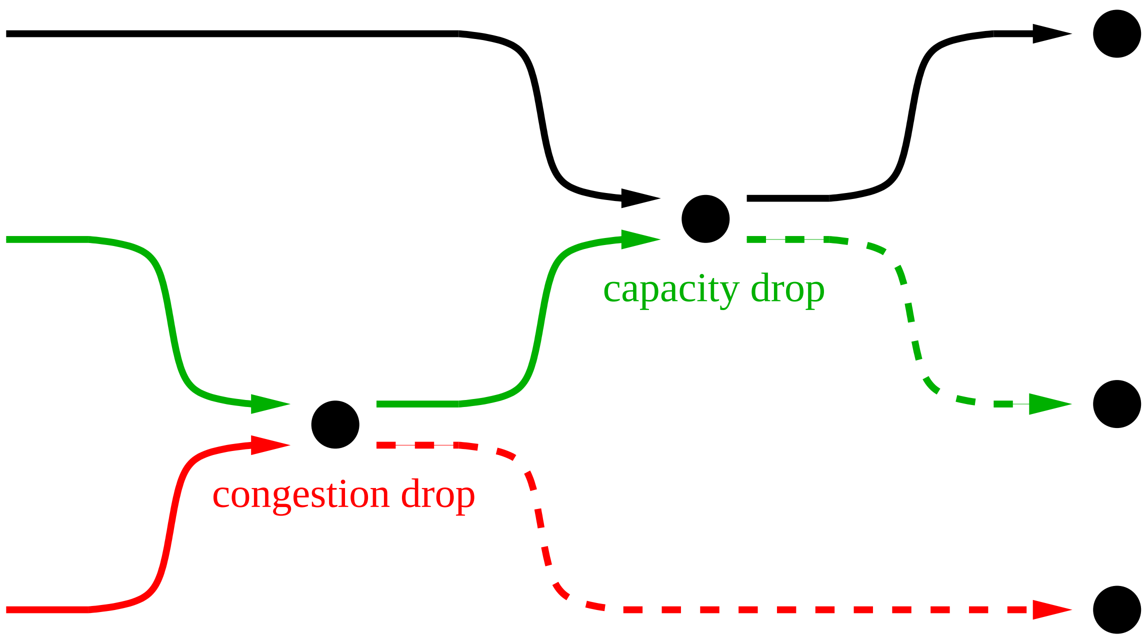
Reasons for bad performance

- One hop: collisions waste time on bus
- Multihop:
  - capacity mismatch
  - packet kills another packet, then dies

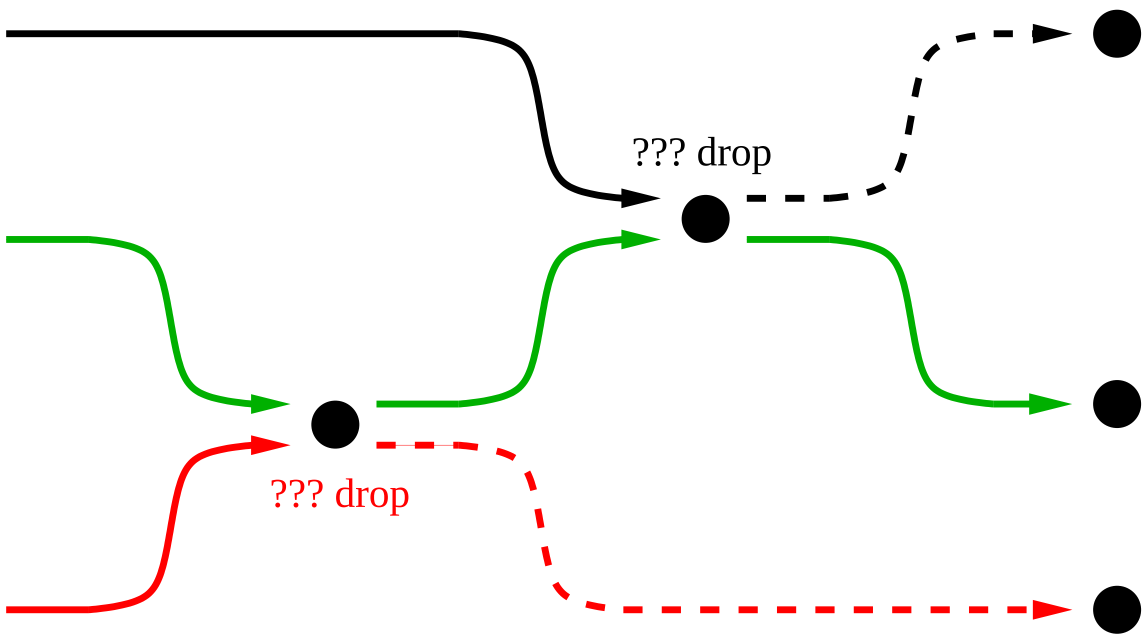
Capacity drop vs. congestion drop



Capacity drop vs. congestion drop



Which kind of drop?



TCP sole place for congestion control?

Pro:

- dropping at source is optimal

Con:

- bad modularization
- **authority-incentive mismatch**

Case for router congestion control

Principle: **align authority with incentive**

- consequence of flooding is global
- congestion *rewards* attack by flooding

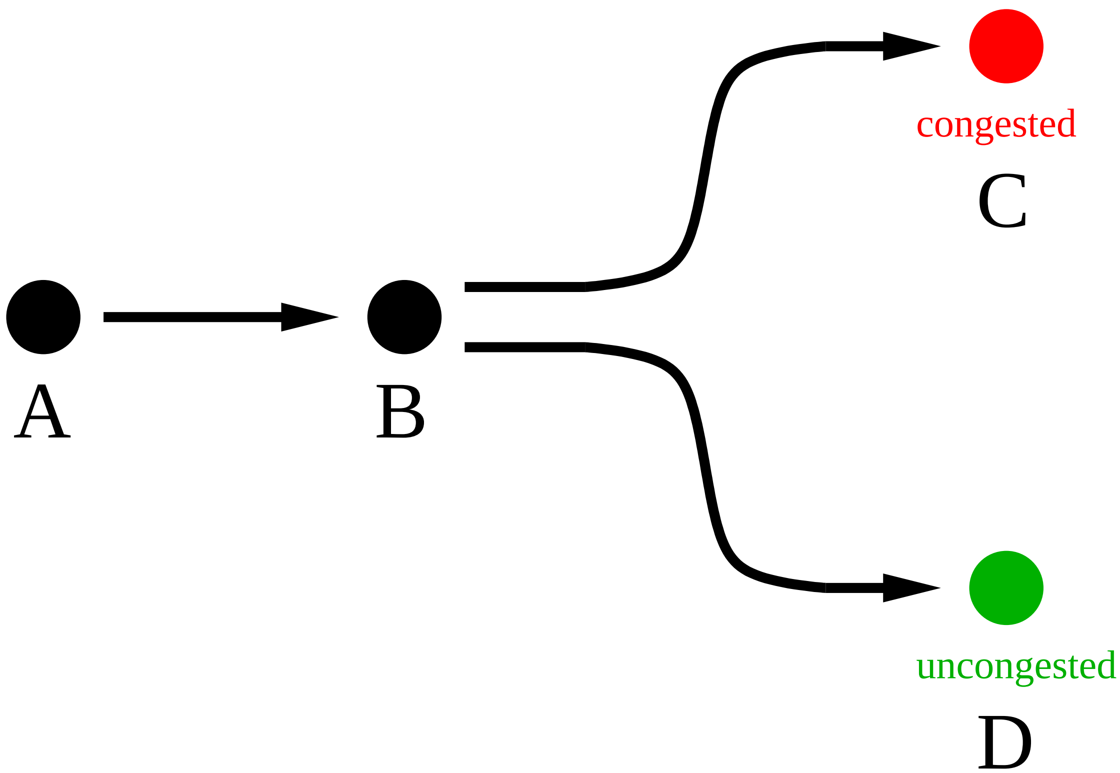
So network routers should
control congestion.

Push back congested flows

Adaptive **H**op-**B**y-**H**op **A**ggregate congestion control,

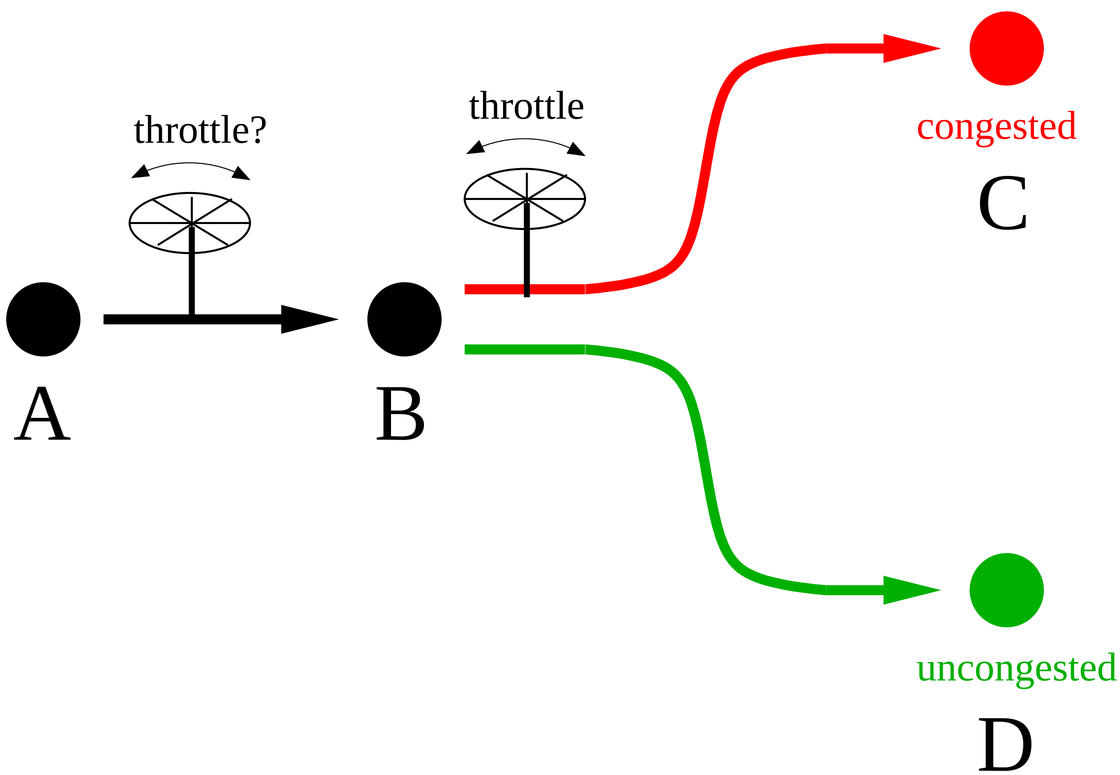
Michael Greenwald, *U. Pennsylvania*

AHBHA congestion control



Problem: Congestion drops at B, A, ...

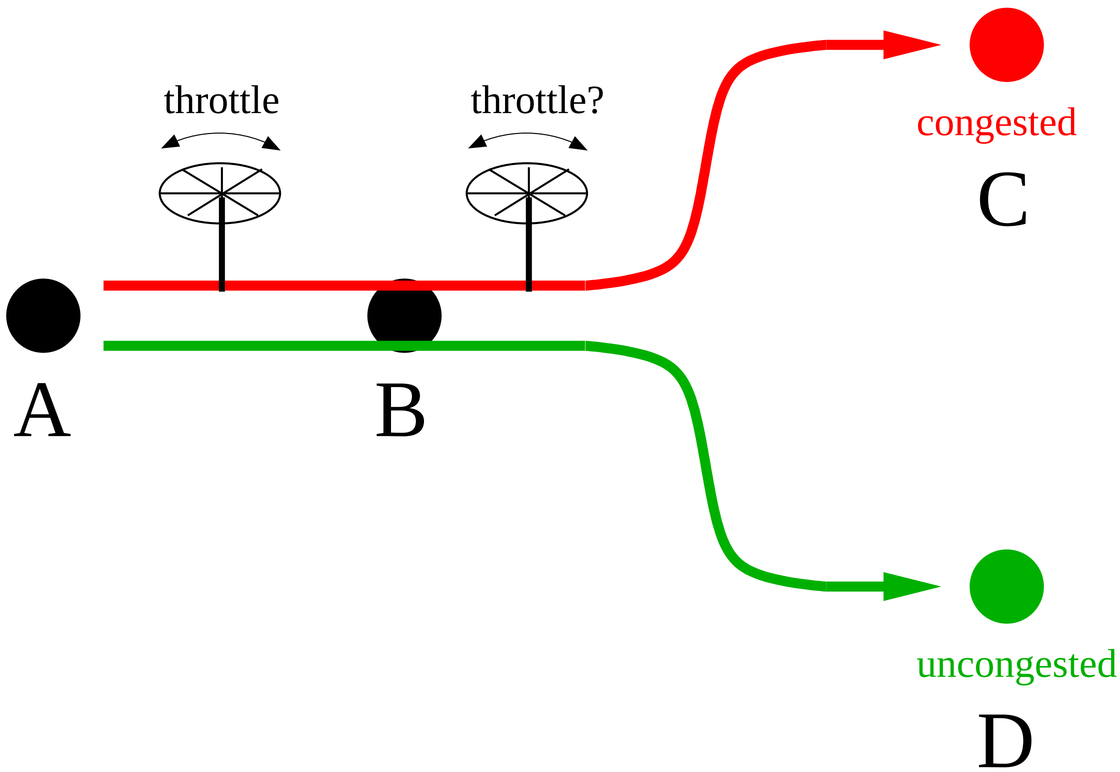
AHBHA congestion control



Problem: B cannot be labelled (un)congested

- Throttling BC is insufficient
- Throttling AB underutilizes BD

AHBHA congestion control



Solution: Split route table entries at A

- AB throttle avoids congestion drop at B
- BC throttle optional

Routes from A		
CIDR prefix	Next hop	throttle?
129.4.112/20	B	no
192.8.48/20	B	no
193.146.128/18	B	no

split

Routes from B		
CIDR prefix	Next hop	throttle?
129.4/16	C	yes
192.8.54/24	C	yes
192.8.56/24	D	no
193.146.128/18	D	no

Routes from A			
CIDR prefix	Next hop	throttle?	
129.4.112/20	B	yes	
192.8.54/24	B	yes	split
192.8.56/24	B	no	split
193.146.128/18	B	no	

Routes from B		
CIDR prefix	Next hop	throttle?
129.4/16	C	yes
192.8.54/24	C	yes
192.8.56/24	D	no
193.146.128/18	D	no

AHBHA adaptation

- Iterate table splits backwards past congestion drops

- Collapse table entries when congestion eases

Quality of service, guarantees, reservations

- Current IP service: “best effort.”

- “**Q**uality **o**f **S**ervice”: distinguish different types of service.

- “Guarantee” is commercial, not technical.

- Technical offering is reservation, not guarantee.

- Reservations reshape failure distribution.

QoS is really reservation

Choice: Add reservations
to Internet service?

The impact of reservations is to reshape the statistical distribution of success/failure.

Lee Breslau, Scott Shenker.
"Best-effort versus reservations:
a simple comparative analysis."
ACM SIGCOMM '98

Example of failure w/wo reservation

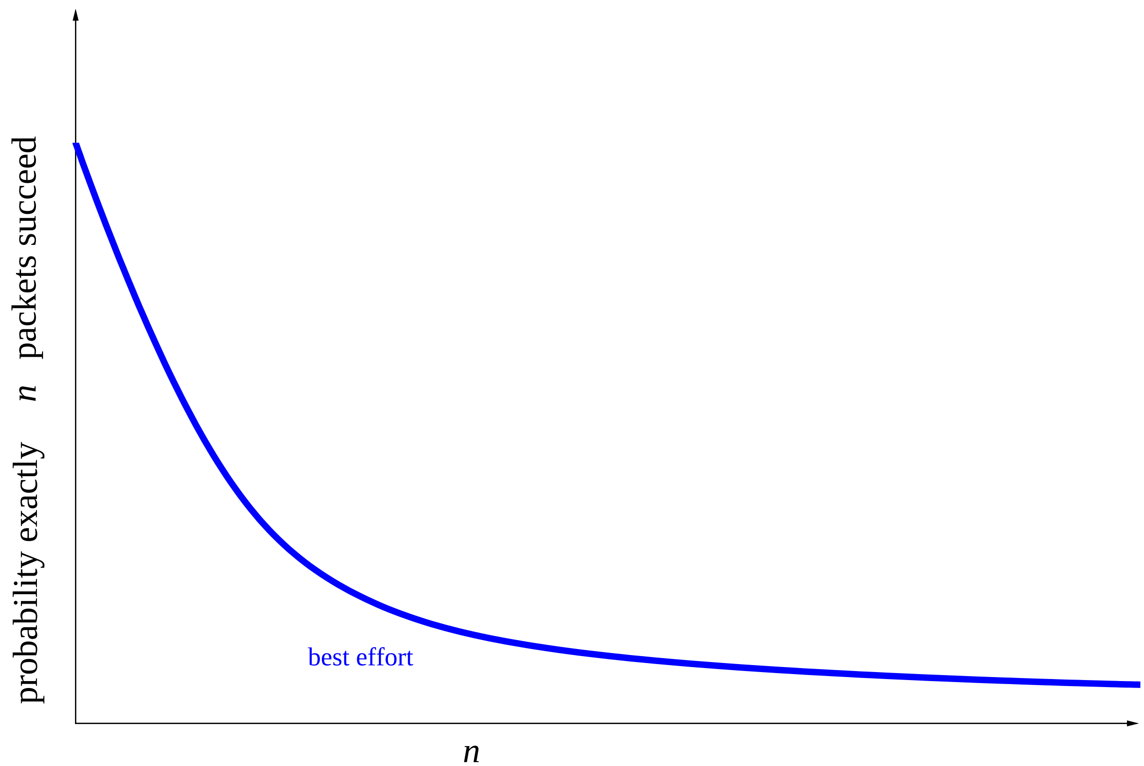
Load network with applications that send a stream of packets, quit at first drop.

Probability exactly n packets succeed:

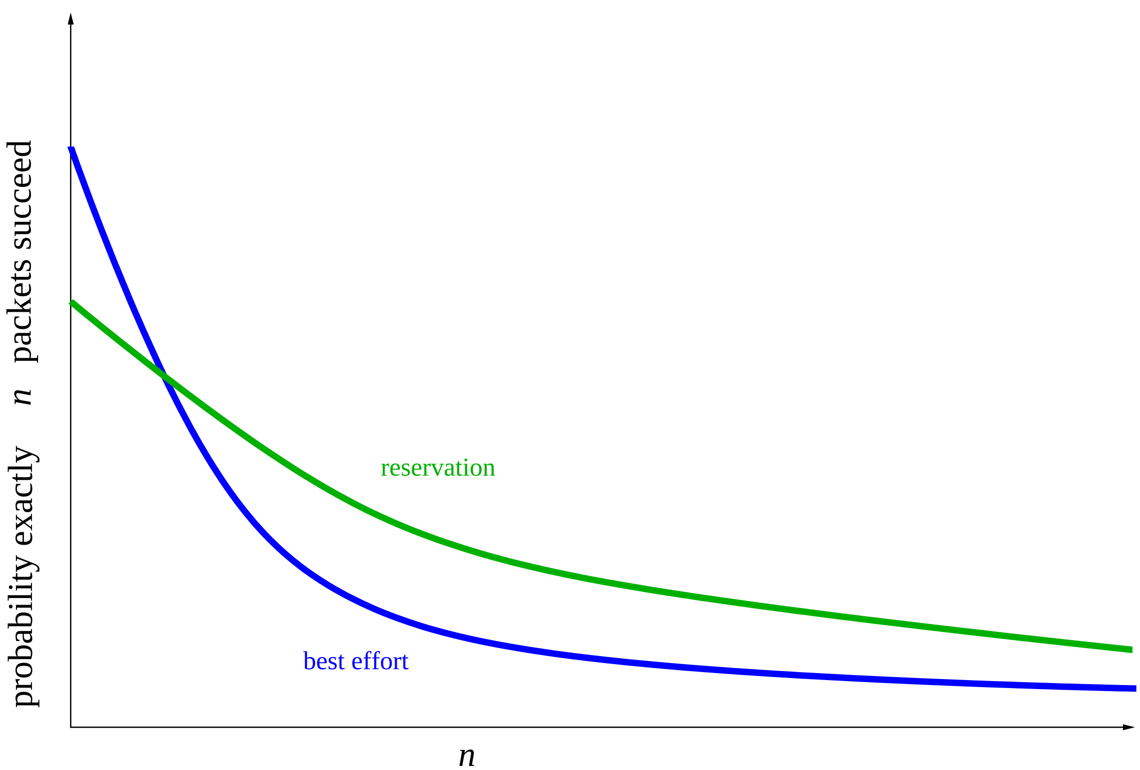
$$
\begin{aligned}
\mathrm{Pr}(n) &= \mathrm{Pr}(\textit{fail}) \cdot (1 - \mathrm{Pr}(\textit{fail}))^n \\
&= \mathrm{Pr}(\textit{fail}) \cdot e^{\ln(1-\mathrm{Pr}(\textit{fail}))n}
\end{aligned}
$$

Reservation reduces Pr(*fail*), flattens exponential density.

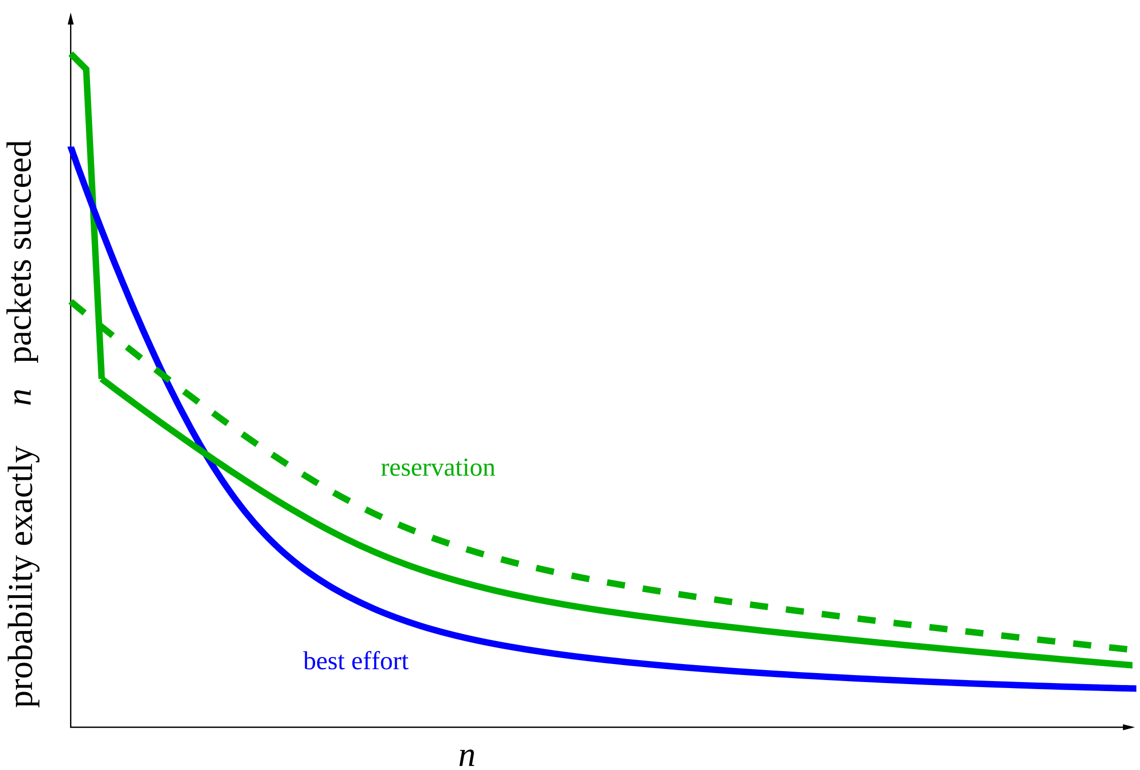
Probabilty of delivering prefix best effort



Probability of delivering prefix given reservation



Probability of delivering prefix requesting reservation



Requesting reservation changes failure granularity

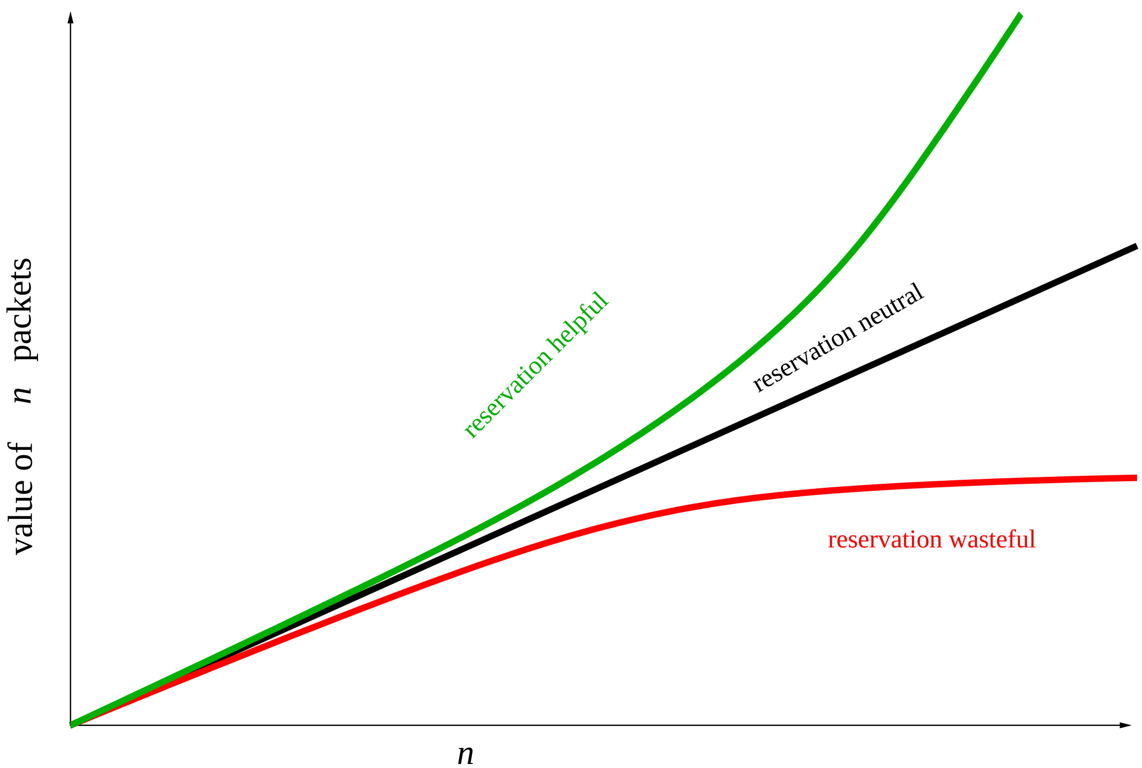
- Flatter exponential has larger mean—exceeds network capacity
- Spike at 0 packets reduces mean to $\leq$ best effort
- Packet failure shifted to reservation failure

Value of reservation to user

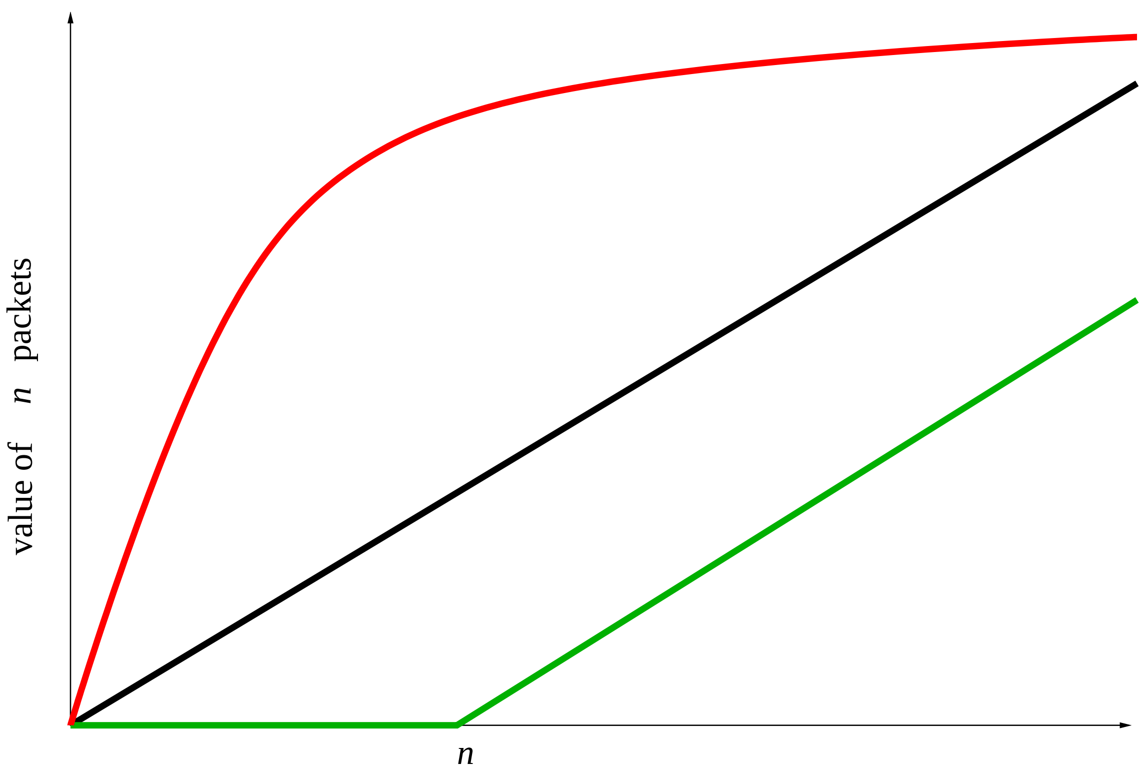
$\pi(n)$ is value of n packets to user

- Overall value $= \sum \pi(n) \cdot \mathrm{Pr}(n)$
- $\pi''(n) > 0 \Rightarrow \pi(a+b) > \pi(a) + \pi(b)$
  - reservation helpful
- $\pi''(n) = 0 \Rightarrow \pi(a+b) = \pi(a) + \pi(b)$
  - reservation neutral
- $\pi''(n) < 0 \Rightarrow \pi(a+b) < \pi(a) + \pi(b)$
  - reservation wasteful

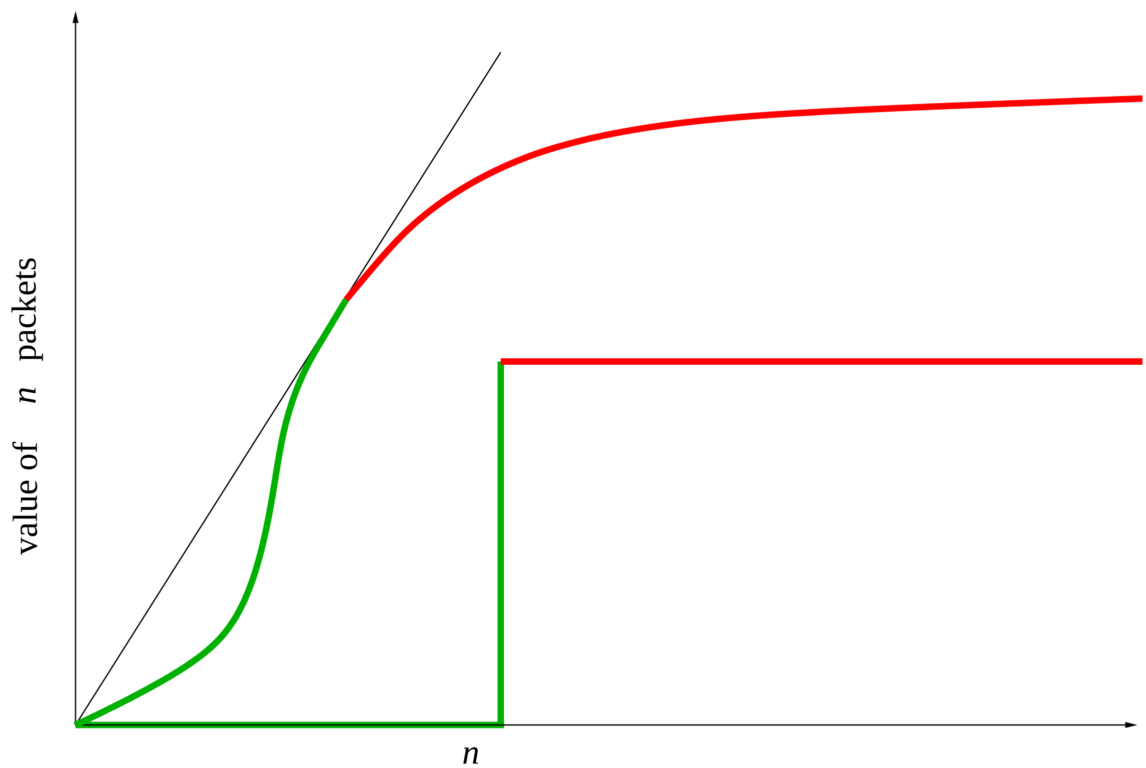
Value function: {super sub}linear



Value functions, (un)helpful reservation



Value functions
limited helpful reservation



Maximize $\pi(n)/n$?

Share a fixed capacity

Breslau & Shenker consider partitioning the capacity of the network among flows.

Their π applies to bandwidth,
not # of packets.

Considerations about shape of π are essentially the same.

What issue will determine reservations?

- maximize global communication utility
  - allocate investment in capacity vs. reservation system

- throughput times delay = packets in transit
  - limits granularity of end-to-end decisions

- reservations $\Rightarrow$ charges?

STUB: cover types of records in some detail.

DNS: names or handles?

- Shortcomings of IP numbers
  - hard to remember/guess
  - dependent on network topology/location

Choice: Introduce meaningful names, not meaningless handles?

DNS history

History:

1. 1980–82 global table of flat host names.

2. c. 1983 too many hosts.

3. 1983–87 RFC 882, 883, 1034, 1035, DNS

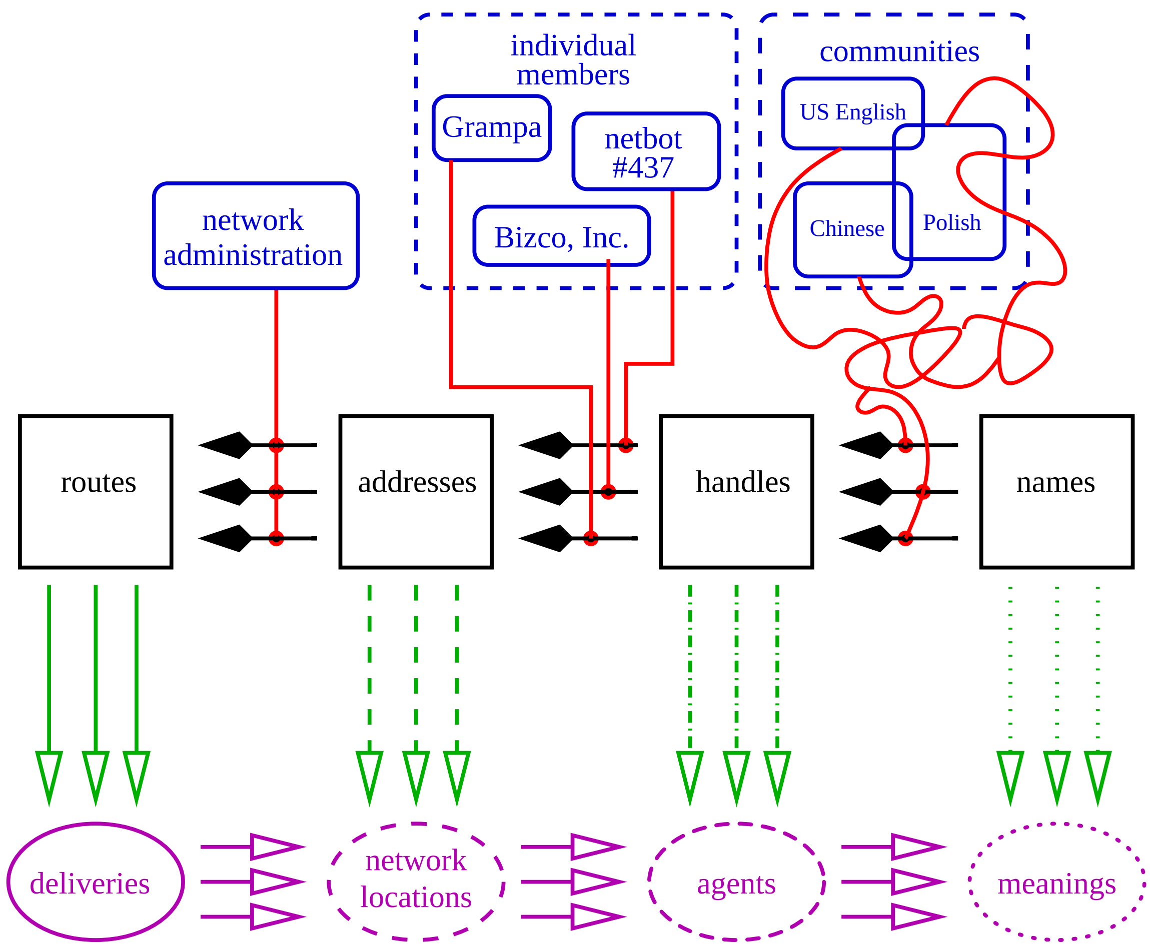
   - STUB for handle quote

   - STUB for name quote

Quibble: Names/handles refer to *agents*, not to *hosts*.
E.g., `www.cs.uchicago.edu`.

Principle: ***Everything* virtualizes.**
***Especially* in cyberspace.**

4 levels of identifiers



Reference between identifiers

- Identifiers *resolve* from right to left.

- Identifiers *refer* downward.

- Square paths should close $\begin{array}{ccc}\odot & \Leftarrow & \odot \\ \Downarrow & & \Uparrow \\ \odot & \Rightarrow & \odot\end{array}$

- All objects are virtual.
  - Meaning of objects derived from relations.
  - Correctness only in relations, not objects.
  - **Real design decision is who controls resolution.**

Pseudohistory of network identifiers

- Routes in UUCP: `host1!host2!...!hostn`
  - Sally sends to Grampa's candy store: `gargoyle!anubis!monkey!candy`
  - No good for Paul—he must use `juniper!elm!granite!arthur!candy`

- IP numbers: `128.72.98.4`
  - Same address reaches Grampa for both Sally and Paul.
  - No good when Grampa moves, network topology changes.

Pseudohistory, continued

- Domain names: `candy.com`
  - Same name works for everyone.
  - Grampa can update resolvent of `candy.com` when he moves.
  - Semantic value of `candy.com` leads to conflict.
    - Bigger candy company wants `candy.com`
    - C and Y company wants `candy.com`
  - Systemic conflict: Chinese alphabet, ...

Possible future

- Handles: `h1g5k0061A38F9A3540B9`
  - Random looking, but not much worse than phone, credit card numbers.
  - Can be self assigned with no central authority.
  - Names resolve to handles which resolve to addresses.
  - We still fight over names.
  - Other ways to find handles besides DNS: Yahoo, Google, ...

Handles ubiquitous in computing systems

- Compiler:
  identifier $\longrightarrow$ sym table index $\longrightarrow$ address

- NEED MORE EXAMPLES

Names only:

- Principle: **When two services are equally easy to implement, choose the more valuable one.**
  - Names more valuable than handles.

Handles and names:

- Principle: **Put different services in different modules.**
  - Name/handle resolution depend on different authorities.

- Counterprinciple: **Valuable scarce resources are costly to defend.**

Security through cryptography

- Privacy/secrecy
- Authentication/signature

Secret-key cryptograpy (e.g. DES)

- Create one random-looking key K

- Key defines two functions, $\underline{K}$, $\overline{K}$, with $\overline{K}(\underline{K}(x)) = x$ (inverses)

- Share secret K among 2 or more communicants

- Important theory: given x, $\underline{K}(x)$, intractable to discover K or compute $\overline{K}$

Public-key cryptography (e.g. RSA)

- Create two random-looking keys K_o (open) and K_s (secret)
- Each key defines a function $\bar{K}$, with $\bar{K}_o(\bar{K}_s(x)) = \bar{K}_s(\bar{K}_o(x)) = x$ (inverses)
- Publish K_o, keep K_s secret
- Important theory: given K_o, intractable to discover K_s or compute $\bar{K}_s$

Using public-key cryptography

Secrecy: Send message x encoded as $\bar{K}_s(x)$.
Recipient decodes by $\bar{K}_s(\bar{K}_o(x)) = x$.
x can be symmetric key.

Signature: Send message x signed as $\bar{K}_s(x)$.
Recipient verifies by $\bar{K}_o(\bar{K}_s(x)) = x$.

Secure hashing (e.g. MD5, SHA)

- No key involved
- $H(x)$ much smaller than x
- H easy to compute
- Important theory: given x, intractable to find y with $H(y) = H(x)$

Using secure hash

Comparison: Test $H(x) = H(y)$
to verify $x = y$

Signature: Send $\bar{K}_s(H(x))$ instead of $\bar{K}_s(x)$.
Recipient verifies by
$\bar{K}_o(\bar{K}_s(H(x))) = H(x)$.

Performance of cryptography

RSA {en/de}crypt:	100 Kbps
DES {en/de}crypt:	100 Mbps
MD5 hash:	600 Mbps
SHA hash:	260 Mbps
RSA/MD5 signature: (1 Kb certificates)	100 Kbps
RSA/MD5 verify:	1000 Kbps

Infrastructure needed for PKC

Two technical system problems

- Manage private keys:
  use without revealing

- Distribute public keys:
  connect key to owner

Private key problem is difficult, but local.

Public key distribution is
network infrastructure.

Chain of trust in public keys

- Must distribute reliable <*name*, *key*> pairs.

- Distribute <R's *name*, R's *key*> out of band.

- R signs <A's *name*, A's *key*>.

- A signs <B's *name*, B's *key*>.

- Users collect <*name*, *key*> pairs.

Problem: A chain is as strong as its weakest link.

Web of trust (e.g. PGP)

- Distribute signed <*name*, *key*> pairs promiscuously.

- Users evaluate total trustworthiness from multiple signatures.

CoT bundles <*name*, *key*> distribution with evaluation.

WoT unbundles, allows arbitrary evaluation.

Unbundle further: distribute *key*s without *name*s.

End-to-end principle applied to authentication.

Identity: relation and object

identity

3. The quality or condition of being the same as something else.

4. The distinct personality of an individual regarded as a persisting entity; individuality.

—The American Heritage Dictionary of the English Language

Identities from authority (top-down)

CoT, WoT associate keys with predetermined names/identities.

Authenticity determined by authority, communicated by PKC.

(authentication by *provenance*)

Outside of net, identities are tricky.

Identities from experience (bottom-up)

Instead of assuming identities,
provide tools for constructing identities.

Identity: relation and object.

Derive identities from identity relation.

Flat key distribution

Idea: Support *identity relation*, not *identities*.

A cryptographic "signature" identifies signer only as the owner of the signing key.

Signature supports identity relation.

That's enough for the bottom-level infrastructure.

Hash shortens key

Problem: Public keys are long
(128 bytes or more)

Solution: Use hash of key

Free tradeoff: long hash for security,
short hash for convenience.

Public key servers

Provide tuples $<$*key*, *hash*(es), *value*$>$
Lookup by key or hash.
value may have any type.

- no responsibility for authenticity (*end-to-end* authentication)

- only facilitates discovery of keys

Notice analogy to best-effort packet delivery.

Key owner has permanent control of database entries:

- may switch key server

- may use multiple servers

- may re-establish *same* key when server dies

Public-key protocol: datatypes

Use principles of distributed DB design. Network performance mostly involves lower level of implementation.

- *key* has form $<$*sigmethod*, *okey*$>$
  sigmethod gives the type of signature,
  okey the open key

- *hash* has form $<$*hashmethod*, *hashokey*$>$

- *value* type is free:
  typically address of owner

- *seqno* is an integer assigned by the owner
  larger number cancels smaller number

- *exp* is expiration date
  (crucial for distributed DB record)

Public-key protocol: messages

Owner signs and announces records:

bind(*key*, *hash*, *value*, *seqno*, *exp*)

delegate(*key*, *hash*, *rhash*, *seqno*, *exp*)

assign(*key*, *hash*, *rhash*, *time*)
(irrevocable)

compromised(*key*, *time*) (irrevocable)

Server answers queries by *hash*, returns most recent signed record (by *seqno*).

assign, **compromised** beat **bind**, **delegate**

Have both **assign** and **compromised**, return both. Querier judges validity.

Who signs records?

Important signature is signature of owner.

- E.g. **bind**(*key*, *hash*, *value*, *seqno*, *exp*) signed by *key*

Server signature adds nothing to positive query response.

Server may sign negative query responses.

Trusted 3d party may re-sign **assign**, **compromised** to certify timestamp.

Purpose of messages

bind allows owner to (re)associate arbitrary *value* with *hash*.

Queries allow others to find *key* and *value* associated with a known hash.
Using *key*, querier authenticates response independent of server.

delegate allows owner to share one address for mutiple distinguished roles.
Share with own or other's address.
Split as resources/requirements dictate.

assign allows owner to upgrade cryptographic method, pass on or sell a role to another owner.

compromised allows owner to disavow a compromised key.

How queriers use records

bind—use *value* until expiration, query again as desired.

delegate—requery *rhash* (after checking signature).
Server returns **delegate** record, not record for *rhash*, so querier may authenticate.

assign—requery *rhash*, and replace *hash* by *rhash* in local records.
Local replacement is crucial, since **assign** record will eventually disappear or be compromised.

Flat key distribution provides an infrastructure analogous to IP level of communication.

CoT, WoT, ... built on top of flat keys.

Vulnerabilities

As long as owner protects private key, cryptosystem not cracked, querier who verifies cannot be fooled.

Loss of key $\Rightarrow$ loss of updatability.

If key cracked, owner announces **compromised** $\Rightarrow$ loss of that key.

Recipient of **assign** vulnerable to **compromised** announcement by previous owner.
Remedy: pay trusted 3d party to certify transfer with time stamp. (This makes time stamp on **compromised** crucial.)

Extra services

Key owners may negotiate extra services (at a price) outside of the basic key server.

- Semantic services
  - association of *key* with *identity*, signed by trusted authority (CoT, WoT, ...).
  - time stamp signed by trusted authority.
- Performance services
  - **assign** record res-signed by trusted authority to survive obsolescence of cryptographic method.

Associating keys with other identifiers

The basic PKI can leverage other systems of identifiers, by encoding hashes/keys into another sort of identifier. Then the hash/key inherits services from the other sort of identifier.

- public-key handles as domain names (STUB: CREDIT)
- Open Privacy Initiative *nym*s
- host names in secure NFS
- STUB: MIT security project

Hashes as domain names

`h1g`*<mmm>*`k`*<n...n>*.*<PKIroot>*

`h1g5k0061A38F9A3540B9.handleroot.example.org`

key tag

hashed key

nameserver for PKI

code for RSA/SHA1

algorithm tag

code for public key system

handle prefix

DNSSEC as PKI

Most of PKI functionality already supported by DNSSEC.

PKI	DNSSEC	note
bind	RR, SIG	signed by key owner
delegate	A, SIG	
assign	DNAME, SIG	
compromised	revocation list	not in DNS records

Add code to check owner signatures on receipt of records.
Not necessary for authenticity, but prevents attack by misinformation.

Hierarchical name structure allows interleaving with conventional names.