SOFTWARE ENGINEERING
FOR THE NEXT INTERNET

Pamela Zave

AT&T Laboratories—Research
Florham Park, New Jersey USA

direct descendant of Bell Labs
just outside New York City
now hiring

www.research.att.com
THE CURRENT INTERNET . . .

. . . IS AN ASTOUNDING SUCCESS:

IT HAS CHANGED THE WORLD

because of this success,

there is EXPLOSIVE GROWTH

in users, traffic, and applications
ACCORDING TO LEADERS OF THE NETWORKING COMMUNITY, WHAT IS WRONG WITH THE INTERNET?

THE INTERNET DOES NOT SATISFY CURRENT OR FUTURE REQUIREMENTS

[Rexford 08]

Three goals must be satisfied better than they are today:

- The Internet should have some assured global properties in areas such as availability, scalability, and quality of service (QoS).
- Service providers need autonomy and economic incentives to provide the network service that people need.
- Routing protocols must be flexible, evolvable, and manageable.

[Feldmann 07]

In addition, deficiencies in these areas deserve special attention:

- security
- device mobility
- failure diagnosis

[Clark et al. 05]

Almost every segment of society has a stake in the Internet. Stakeholders with conflicting interests use the available mechanisms to push toward their goals. This "tussle" distorts and damages unrelated aspects of the architecture.
WHAT ARE THEY DOING ABOUT IT?

"THE NEXT INTERNET"

IT IS A SWING OF THE PENDULUM . . .

... from "research is useless unless it can be adopted immediately within the existing Internet"

... to "the current Internet architecture makes solving some problems impossible, so we must have a clean slate"

IT IS MORE REALISTIC THAN IT SOUNDS

● many people understand it as an intellectual exercise that might lead to fresh ideas and a more scientific foundation for research

● the networking community is quite savvy about economic incentives and incremental or evolutionary adoption strategies

IT IS THE SOURCE OF A LOT OF RESEARCH FUNDING

for example:

● European Community umbrella programs: FEDERICA, 4Ward, EIFFEL

● US National Science Foundation umbrella programs: GENI and FIND (the first wave), Network Science and Engineering (the second wave)
ACCORDING TO THE PERSPECTIVE OF SOFTWARE ENGINEERING, WHAT IS WRONG WITH THE INTERNET?

IT IS MUCH TOO DIFFICULT TO BUILD, DEPLOY, AND MAINTAIN APPLICATIONS

This is true already.

In the future, application software will need to be . . .

. . . more complex
. . . more interoperable
. . . more trustworthy
. . . more robust
. . . easier to use
. . . easier to maintain

In four years of following network research and studying the literature, I have not seen any work I thought would help us much.

this emphatically includes work under the umbrella of the "Next Internet"
WHAT SHOULD WE BE DOING ABOUT IT?
GETTING INVOLVED IN "THE NEXT INTERNET"

TAKE AN INTEREST IN OUR RIGHTFUL TERRITORY

- the charter of the NSF's Network Science and Engineering program includes "rethink[ing] network functions, layers, and abstractions"

- the networking community does sometimes attempt to provide application-oriented functions, but does not do it well

LEARN TO WORK WITH THE NETWORKING COMMUNITY . . .

. . . which requires understanding the networking perspective

TAKE SOME OF THIS RESEARCH FUNDING!

1. What makes the Internet so hostile to today's applications?
2. How did the Internet get this way?
3. What can we, as software engineers, do about it?
THE "CLASSIC" INTERNET ARCHITECTURE

A five-layer stack is most often used to describe Internet architecture.

Middleware is becoming increasingly important.

Applications

Middleware

Transport layer (TCP, UDP)

Network layer (IPv4, IPv6)

Link layer

Physical layer

The Domain Name System (DNS) provides mnemonic, user-friendly names for nodes.

Port numbers identify applications and streams (according to application needs).

Nodes are identified, globally and uniquely, by Internet Protocol (IP) addresses.

IP provides best-effort packet delivery between nodes.

Some of the application problems are inherent in this architecture, . . .

. . . while others are due to the many warts that have grown on it.
A NETWORK SERVICE THAT APPLICATIONS NEED:
PUTTING APPLICATION SERVERS IN MESSAGE PATHS

- increases privacy and reduces annoyance, on behalf of employee
- blocks access to certain Web sites, on behalf of enterprise
- reduces bandwidth and improves latency, on behalf of Internet service provider

**AVAILABLE MECHANISMS**

- can configure each browser or proxy with the address of the next server in the desired path
  - administratively burdensome
  - some browsers and proxies are expected to act against their stakeholder's self-interest

- can configure routing of HTTP messages, in the underlying network, to enforce the desired path
  - administratively burdensome
  - not available to some stakeholders
A NETWORK SERVICE THAT APPLICATIONS NEED:
PEER-TO-PEER COMMUNICATION ACROSS FIREWALLS

A default firewall allows only sessions initiated from behind the firewall.

In peer-to-peer communication, peers both initiate and accept sessions.

Peers discover each other by initiating sessions with a rendezvous server.

Once peers agree to communicate, they should establish direct peer-to-peer sessions.

Fewer resources, better latency.

Until 2004, it was considered impossible to establish a TCP connection between peers behind default firewalls.

Even now, it works only 85-90% of the time. [Guha & Francis 08]

It is an ugly hack, depends on details of firewall behavior (which differ widely).
A NETWORK GUARANTEE THAT APPLICATIONS NEED: CONTROL OF THEIR OWN PORTS

PURPOSES OF NETWORK ADDRESS TRANSLATION (NAT)

- Some IP addresses can be re-used
- Private management of subnet address space
- Often combined with a firewall

NAT DOESN'T BOTHER YOUR WEB BROWSER OR EMAIL . . .

. . . because NAT boxes are crafted so that those applications will work

IP-based multimedia applications use a signaling channel to create multiple media channels

"solutions" are ugly and unreliable, because they depend on NAT details

IP node

NAT box

IP node

private subnet

public Internet

public address of NAT box

dynamic mapping encodes private address and port in public port

NAT box

because a NAT box changes addresses and ports, the signaled information is wrong

some IP addresses can be re-used

private management of subnet address space

often combined with a firewall
A NETWORK SERVICE THAT APPLICATIONS NEED:
CONTROL OVER LAYERING

DEVICE MOBILITY:
node and its identifier change location
and therefore IP address (because IP
addresses are location-dependent)

PERSONAL MOBILITY:
person and his identifier move from
node (device) to node (device)

in both cases, messages to the identifier
must be forwarded by a server that knows
the current binding of the identifier

a TCP connection is node-to-node because
all its state is kept in endpoint nodes

endpoints of a connection do not change

TCP

endpoints of a connection can change
during connection

personal mobility

TCP

it is difficult to
have this control
because . . .

. . . below TCP, names are rigid,
name spaces must be embedded
in IP address space, implementor
must have access to IP routers
OTHER NETWORK SERVICES FOR APPLICATIONS

QUALITY OF SERVICE
- monitor a data stream
- buffer to reduce jitter

SECURITY
- block unwanted messages
- authenticate source addresses
- provide encryption

BRIDGE HETEROGENEOUS SUBNETWORKS OR PROTOCOLS

all of these are implemented by inserting servers in message paths
1. What makes the Internet so hostile to today's applications?
2. How did the Internet get this way?
3. What can we, as software engineers, do about it?
THE END-TO-END PRINCIPLE

THE PRINCIPLE:

[Saltzer, Reed, & Clark 84]

FUNCTIONS SHOULD BE MOVED UPWARD IN A LAYERED NETWORK, AS CLOSE AS POSSIBLE TO THE APPLICATIONS THAT USE THEM

BECAUSE . . .

. . . many application functions can be performed exactly right only by the applications themselves, so similar functions in core* layers are redundant (at best) or conflicting (at worst)

. . . functions in the core* layers add complexity, resource costs, and performance degradation that burden all applications, whether they use the functions or not

HOW THE PRINCIPLE SHAPED THE INTERNET

[Clark 88]

originally IP and TCP were one protocol

a reliable, FIFO, duplicate-free stream is an excellent building-block for applications to use

but the TCP functions are detrimental to real-time traffic such as voice, which doesn't need them anyway

so we also have UDP

WHY IS IT CALLED "END-TO-END"?

* "core" layers are lower layers that cannot be circumvented

app
TCP
IP

all TCP state is in the endpoint nodes

if this router fails, the TCP connection is not disrupted
THE END-TO-END PRINCIPLE: ANTI-SERVER BIAS

THIS CONFIGURATION DOES NOT VIOLATE THE END-TO-END PRINCIPLE

BECAUSE OF ANTI-SERVER BIAS, WE HAVE . . .

YET IT IS ALMOST ALWAYS CONSIDERED A VIOLATION!

it is true that the application server introduces an additional point of failure

but this must be weighed against the valuable functions that cannot be performed by user nodes, for example:

○ protect users from denial-of-service attacks
○ act as a persistent network presence for mobile, handheld devices
○ run software that is difficult to install on user nodes, or meets goals in conflict with user's

[Blumenthal & Clark 01]

. . . but no idea of what it might be

. . . no mechanisms for putting application servers in message paths
. . . poorly-designed "middleboxes" (firewalls, NAT boxes) being deployed everywhere, but ignored by researchers

THERE IS A CALL FOR A SUCCESSOR TO THE END-TO-END PRINCIPLE . . .
TALES OF SIP (THE SESSION INITIATION PROTOCOL)

SIP IS THE DOMINANT SIGNALING PROTOCOL FOR IP-BASED MULTIMEDIA APPLICATIONS

- telecommunications
  - voice-over-IP
    - video chat
  - large-scale conferencing
    - telemonitoring

- computer-supported cooperative work
  - embedded telecommunications
    - distance learning
  - emergency services
    - virtual reality

- computer-supported cooperative play
  - multiplayer games
    - collaborative television
  - networked music performance

SIP IS STANDARDIZED BY THE INTERNET ENGINEERING TASK FORCE (IETF)

ALTHOUGH IT IS AN APPLICATION PROTOCOL, SIP CAME FROM THE NETWORKING COMMUNITY, AND EXEMPLIFIES MANY OF ITS CHARACTERISTICS
TALES OF SIP: THE PROTOCOL SPECIFICATION

SIP HAS BEEN, AND IS BEING, DEFINED IN RESPONSE TO A SERIES OF USE CASES

The specification is an ever-growing series of English standards documents (IETF RFCs), illustrated by message sequence charts.

"A Hitchhiker's Guide to SIP" gives a snapshot of SIP RFCs and drafts . . .

. . . which lists 142 documents, totaling many thousands of pages

this is an extreme case of bottom-up thinking

[S Rosenberg 08]

SOME CONSEQUENCES

• frequent "bake-offs" where vendors meet face-to-face to make their equipment interoperate

• everyone wants "simple SIP", and everyone has a different idea of what it should be

• a false opposition between generality and simplicity

no conception that a protocol based on better abstractions could be both more general and simpler

• really high message overhead in SIP applications

in contrast to the Distributed Feature Composition (DFC) architecture

[Jackson & Zave 98]
TALES OF SIP: THE IETF

THE "CORNER CASE"

Definition:

A corner case is a possible behavior that emerges from the interaction of anticipated behaviors and constraints.

It is undesirable, and is declared to be rare and unimportant without evidence.

THE VENDORS

The IETF is dominated by equipment manufacturers.

Successful equipment manufacturers do not want standards.

They adhere to standards only under pressure from their customers.

The participate in standards bodies like the IETF as a highly competitive game.

THE TECHNOLOGY

The IETF philosophy is to standardize based on "rough consensus and working code".

A finite-state machine is considered exotic.

I built a Promela model of some core aspects of SIP, then did some basic validation and verification with the Spin model checker.

- there is a SIP RFC (four years in the making) that describes 7 race conditions
- my model reveals these and 42 others in the same category

The model is public, serves as complete, centralized, executable documentation.

Yet it is an uphill battle to get the SIP Working Group to consider it.
TALES OF SIP: ANTI-SERVER BIAS

IN BUILDING ENTERPRISE SERVICES, THIS IS ONE OF THE MOST BASIC THINGS WE DO:

1. application server
   - SIP
   - How may I direct your call?
   - media server

2. application server
   - SIP

YET BECAUSE SIP IS DEFINED END-TO-END . . .

. . . THIS USE CASE IS NOT PROPERLY SUPPORTED IN SIP
THE NETWORK-LAYER PILE-UP

MANY (MOST?) NETWORK FUNCTIONS ARE PROVIDED BY CRAMMING THEM INTO THE IP LAYER

For example:

- security based on firewalls
- address management by NAT
- mobility
- multicast
- virtual private networks

Reasons:

- efficiency always takes precedence over separation of concerns
- this is the only readily available mechanism

CONSEQUENCES

- correct configuration of IP routers is a major challenge
- vendors of IP routers have complex proprietary configuration languages
- stakeholders without access to IP routers are largely disenfranchised
1. What makes the Internet so hostile to today's applications?

2. How did the Internet get this way?

3. What can we, as software engineers, do about it?
### OVERLAYS

#### ANATOMY OF AN OVERLAY

<table>
<thead>
<tr>
<th>SPECIFICATION</th>
<th>autonomous computing elements (member nodes)</th>
<th>specification of a communication service among its nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>users</td>
<td>may include . . .</td>
</tr>
<tr>
<td></td>
<td>servers</td>
<td>. . . compound communication primitives</td>
</tr>
<tr>
<td></td>
<td>both</td>
<td>. . . security properties</td>
</tr>
<tr>
<td></td>
<td></td>
<td>. . . abstract names and their dynamic binding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>. . . path guarantees</td>
</tr>
</tbody>
</table>

#### IMPLEMENTATION

- protocols used by nodes to distribute control information
- algorithms and data structures in nodes
- multi-hop protocols for data transport

#### ASSUMPTIONS

- assumptions about the lower-layer communication service that will implement each hop

Overlays are used by the networking community for clean-slate design, and usually implemented on top of IP.
assumptions of this overlay must be satisfied by specification of this overlay

these underlying overlays (underlays) are bridged by their overlay

composition can be implemented efficiently, even if dynamic

[Ma et al. 08]
THE SUCCESSOR TO THE END-TO-END PRINCIPLE:

EACH APPLICATION SHOULD RUN ON A STACK OF OVERLAYS THAT EXACTLY MEETS ITS REQUIREMENTS . . .

. . . so that it does not incur the overhead of unnecessary functions

there is a hierarchy of optional, composable overlays

overlay in which application processes are user nodes

application may choose type, instance, or parameters of next overlay

application may make or delegate the choices below

application choices must be constrained or pre-configured for the requirements of other stakeholders

the bottom underlay runs on a virtual network, which gets a predictable slice of the real network resources

[Anderson et al. 05]
FUTURE WORK: DESIGN OF THE OVERLAY HIERARCHY

- Higher overlays overlap (and should be integrated) with middleware
- Many instances of these types, for different purposes, with different costs
- Middleware functions
- Create a new name space, with dynamic binding and source authentication
- Insertion of servers in message paths
- DFC router is a feature-rich solution to this problem [Jackson & Zave 98]
- Compound communication primitives (sessions, connections, multicast), possibly with quality-of-service monitoring
- Security by encryption
- Security by constrained reachability, with capabilities
  - Major goal: routing constraint
- Management and extension of IP address space
  - Major goal: routing freedom

How can we design each overlay with maximum generality, simplicity, scalability, and separation of concerns?

How do overlays interact, and how does that constrain their order?

Which compositions should be static, and which dynamic?

Does the "tussle" among stakeholders occur within overlays, between overlays, or both?
FUTURE WORK: TECHNOLOGY FOR OVERLAYS

here we are on solid ground:
applying our expertise in a new area

OVERLAY IMPLEMENTATION AND COMPOSITION

- overlay specification
- verification of overlay implementations
- verification of overlay compositions
- mechanisms for overlay choice
- secure, efficient dynamic composition

OVERLAY DEPLOYMENT

- help with configuring and initializing overlays
- help with deploying overlays
- empirical evaluation and tuning

much of the "Next Internet" funding is being spent on new experimental test beds, so there will be platforms to work on
THE NEXT INTERNET WANTS YOU!

. . . TO SOLVE THE PROBLEMS THAT MAKE APPLICATIONS SO HARD TO BUILD AND DEPLOY

   which no one else is working on in the right way

. . . TO PROVIDE THE DUAL PERSPECTIVE THAT IS SO BADLY NEEDED

   top-down, from the software-engineering community

   bottom-up, from the networking community

. . . TO EDUCATE THE CENTERS OF INTERNET POWER ABOUT THE NEED FOR CHANGE

DOMAIN-SPECIFIC RESEARCH . . .

. . . is the best way to make progress and have fun

THE BIBLIOGRAPHY . . .

. . . was chosen to provide a good introduction to networking research for software engineers

- I will keep an updated version on my Web site
- if you want to understand the Internet, read any paper by David D. Clark

SPECIAL THANKS TO . . .

. . . Jennifer Rexford of Princeton University, who is my networking mentor

. . . the organizers of ICSE