FIND and Architecture
A new NSF initiative

David D. Clark
MIT CSAIL
October 2005

Topics to discuss

The FIND initiative

- What is the motivation for this program?

Why we talk about architecture

- In general, and for Internet specifically

Some specific thoughts about design

FIND: A challenge question

1) What are the requirements for the global network of 10 or 15 years from now, and what should that network look like?

To conceive the future, it helps to let go of the present:

2) How would we re-conceive tomorrow's global network today, if we could design it from scratch?

- This is not change for the sake of change, but a chance to free our minds.

Isn’t today’s net good enough?

Security and robustness.

- As available as the phone system
- Been trying for 15 years--try differently?

Easier to manage.

- Really hard intellectual problem
- No framework in original design.

Recognize the importance of non-technical considerations

- Consider the economic landscape.
- Consider the social context.

What will be happening in 10 years

- New network technology.
  - Wireless
    - Mobility
    - Dynamic capacity allocation
    - Dynamic impairments
  - Advanced optics
    - Dynamic capacity allocation (again!)
- New computing paradigms
  - Embedded processor, sensors, everywhere
- Whatever computing is, that is what the Internet should support.
  - The Internet grew up in a stable “PC” time.

The scope of the challenge

Is it “Internet classic”? A cloud of routers with general purpose computers at the edges?

No! The scope of the question is much bigger than that.

Ask: what will “the edge” look like. That is where the action is.

- Sensors. Embedded computers.

Ask: what is it that users do? Try to conceptualize a network that supports that.

- Information access and dissemination.
- Location management and location-aware systems.
- Identity management systems.
- Conceptualize at a higher level (not higher layer).

What should we reconsider?

- For the moment, everything.
  - Packets, datagrams, circuits--everything.
  - Our religious beliefs
    - End to end, transparency, our model for layering.
- To conceive of a future, we have to let go of the present.
  - This does *not* mean that we cannot get there incrementally.

Timing

- This is a long term effort.
  - IPv6 started in 1990.
- It is less important when we start, more important that we do so.
  - We can and will do mid-course correction.
  - Adjust the objective as we get closer.
- Long term research has short-term fallout.
- Short term research never achieves a long-term objective.

Defining success

- We throw away the current Internet.
  - The most dramatic form of success.
- We set a goal, and the we realize we can get there incrementally.
  - Impose a bias or direction on change.
- Lots of fresh ideas leak into the present Internet.

The benefit

Today, we see erosion of clean design principles--architecture.

Clean architecture means clean interfaces, as well as better behavior.

Creates more opportunities for innovation.

- The NAT story.

If we don’t do this?

If we don’t step up to conceive of what networking will be in 10 years:

- A narrowing of the utility of the Internet to specific purposes. E-commerce?
- A pervasive loss of confidence in Internet.
- Limit our ability to exploit new technology.
- A loss of funding (inside NSF) to sectors that seem more relevant and vigorous.
  - A gentle glide into irrelevance for research.

Architecture

A process: putting components together to make an entity that serves a purpose.

A result: entities come to be defined by their architecture.

- Think about the original form of architecture.

A discipline: architects study past examples, learn patterns and approaches.

All of these apply to “real architects” and to computer science.

Some thoughts:

Putting components together:

- Modularity, interfaces, reuse, dependency

For a purpose:

- Successful architecture recognizes what a system *cannot* do.
- But we worship generality.

Design patterns, approaches and cautions.

- General: layering, abstraction, size of modules, second-system syndrome.
- Specific: end to end, transparency vs. conversion (spanning layer), the “hour-glass model”, soft/hard state.

The distinctive Internet

Most architecture leads to one instance.

- One building, one chip design, etc.

The Internet led to multiple implementations.

- Fundamental requirement: interoperation.
- Less is better: what is not architected is as important as what is.

So what do you architect?

That which is very helpful to agree on generally.

- Principle of architectural minimality.
- Applications have their own architectures.

That which is very helpful to reuse.

- The boundary between architecture and component.

But that is pretty abstract advice…

And your point is…?

Network architecture is first, a debate about what it is we (ought to/need to/want to) agree on.

- If we don't need to agree, then don't embed the concept in the architecture.
- As is true in general, architecture is first a debate about the purpose of the system.

Then it is a debate about to realize the agreement.

Thinking architecturally

- Cool new technology is often first conceived outside of any proposal for its use.
  - And early proposals are usually wrong.
  - The laser story.
- But as technology matures, research fashions it to fit into a set of anticipated requirements.
  - This is “thinking architecturally” about a technology or component.
  - This is part of “architecture” research.

The “old” Internet

- Packet format.
  - Trying to replace that…
- Global addresses.
  - Broke that…
- Oblivious transport (end to end).
  - Eroding…
- Hosts are not routers (don’t run routing protocols)
  - Starting to break that…
- BGP (EBGP)
  - Talking about replacing that.
- DNS
  - Broke much of that…

FIND: a new IP?

Why do we need an “IP layer”? Prefer it?

- A preference for no flow setup.
- A preference for little state in routers.
- This is “oblivious forwarding”.

But…

- Today we do “stateless” address rewriting.
- Can do “stateless” label switching.
- Rewrite IPv4 <-> IPv6, encapsulate, etc.

Perhaps a header format is not the defining piece of a new architecture.

Internetwork architecture

To this point: some generalities about architecture, both classic and CS.

Return to what is special about Internet.

- The search for generality.
- The fear of commitment.

The search for generality

How do you make a “general” system?

Never commit to what it does.

- Commitment may “freeze” the system.

Design (architect) cool building blocks and hope someone can arrange them later.

- Run-time architecting.

We do this all the time.

QoS as an example

We fight over two approaches to specification.

- Per-hop behavior (PHB), composable along a flow to get overall semantics.
  - But does this provide actual isolation?
  - How is behavior composed? (Flow setup?)
- Defined end-to-end behaviors
  - TCP-friendly rate adaptation.

This tension between approaches is basic.

Security--another example

A firewall is a PHB.

- It tells you nothing about how you achieve good overall security, or what security you can achieve.

Alternative: some sort of “negative availability principle”.

- If one set of nodes doesn’t want some other set of nodes to talk to them, the network should enforce that (or “help” enforce).
- A bold, dangerous idea…

Consider economics

- What does an ISP sell? What do I buy?
- PHBs are (relatively) easy to create, but are they worth much?
- Selling an end-to-end service seems like more value, but is hard.
  - Have to agree on what the service is.
  - Requires cooperation on service creation, revenue allocation, etc.
- Consider the current work in ITU.

Deep debate (religion)

PHBs are “general purpose” building blocks

- Great idea if they are really reusable.
  - Implies mechanism for composition at run-time.
- Late binding defers painful debate.
- Different parts of net can be different.

End to end behaviors require *a priori* agreement on desired services.

- Favor some uses over others
- Further erodes generality of end-to-end.
- But services are what “real users” want, and real providers want to sell.

Design for today or tomorrow?

Design for today:

- Support the known apps.
- Make money.
- Embrace services.

Design for tomorrow:

- Don’t block innovation.
- Find cool building blocks.

Can we perhaps do both?

Design for change

End-to-end arguments.

- General mechanism “in the net”.

Run-time application adaptability.

Weak semantics.

Open interfaces.

- Can insert “new” behavior.

Time to attend to service composition?

- What mechanism might be useful here?

Architectural stability

In the past, change has implied erosion of architecture.

- Can we design *architecture* for change?

Look at management

Internet is very hard to manage.

- Affects everyone, big and little.

Management was not an initial high-priority concern.

We have no *architecture* for management.

- Not even sure what that might mean.

Much research in this area has floundered.

- Why?

An example-failures

Dealing with failures is hard because a basic tool, *abstraction*, is not useful.

When an abstraction fails, one must “dive through” and find the real cause.

- A soup of details.
- A layered, segregated world.
- Incompatible recovery actions.

An example: configuration

"Manage the net, not the box."

The opposite problem--need a higher layer abstraction.

- How expressive?
- How limiting?
- One or many?
- Can it be bypassed?

Regions or end-to-end?

It makes sense if each of us manages our "part" of the network.

But lots of problems manifest at the edges and across multiple parts.

How can these two ideas be reconciled?

- The "knowledge plane" was one ambitious cut at this problem.

Design for manageability

- Should our data-plane protocols be enhanced to facilitate management.
- Should components come out of the box ready to be managed?
  - Must eliminate silent failures.
  - But who to tell?
  - A business opportunity?

Some thoughts

- Should we design building blocks or management services?
- Is layering the enemy of coherent management?
- How can regions implement joint management of problems?

These, I believe, are examples of *management architecture*.