



# Reinventing the Internet: Is it time?

---

David D. Clark  
MIT CSAIL  
April 2006

# Some background

---

Internet was designed about 30 years ago.

Very bold and uncertain concept.

- Packet switching.
- Layering
- Protocol spec

Works amazingly well.

But it has persistent problems.

- Not basic packet forwarding...
-

# So what is wrong?

---

## Better security:

- A compelling justification.
- Been trying for almost 20 years--try differently?
- Define broadly--availability and robustness.
- Need an overarching framework.
- Must deal with end-nodes--what can and should the network do?

## Industry structure and economics:

- Making money from apps you don't own.
- Focus on interconnection.

## Larger social context:

---

# More requirements

---

## Better management:

- No architecture, but fresh ideas.
- Not a good poster child.
- Focus on the user, perhaps?

Better incorporation of network technology.

Both public and private networks.

Service in times of crisis.

---

# Support new computing

---

Our job is to connect computers.

- What will “computers” look like in ten years.
- Don't *start* with the network.

My prediction:

- The decade of cheap ubiquitous, low power.
    - *Not* performance.
    - The \$1 device, the ten cent network.
    - Getting “unstuck from 1”.
  - The decade of the application.
    - What do apps do? How do we support them?
-

# If processing is everywhere:

---

New management paradigms?

How do systems organize and how do parts find each other?

The crucial role of physical location.

New security issues.

Is infra open or integrated?

---

# The reality of applications

---

- They are not “end-to-end”.
  - Lots of Intermediate nodes. Why? Who?
- They are location-aware.
- They are the source of many security problems.
- They are the space of tussle.
- They are the driver of value.

Now: what will they look like in 10 years?

We must become application-centered.

---

# FIND: A challenge question

---

An NSF initiative, now being funded:

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.
-

# Why? (At a high level)

---

## External:

- The world will need a different Internet in 10
- years.
- To get it you have to envision it.
- To envision it you have to free your mind.

## Internal:

- Network research seem to be drifting.
  - Incremental, not goal directed.
  - A vision of a new network can motivate.
-

# FIND: it's not a new IP

---

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

Perhaps we focus on control, management and “other planes”.

- Data plane can fend for itself.
-

# 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...
-

# So the answer is...?

---

What are the architectural aspects of a new Internet?

- Well, we don't know yet. It is research.

Might be concepts closer to what the users want to do.

- Information dissemination.
  - Identity architecture.
  - Location management.
-

# Topics to discuss

---

## 2.5 case studies:

- Security
  - Economics and industry structure
  - Generality and service composition
-

# Security as an example

---

Define security generally.

- Not just disclosure control and integrity.
- Focus on availability
- Focus on user expectations of a safe and understandable experience.

Include the end node.

- Don't assume that bad end-node security is someone else's problem.
  - Don't assume the net will solve the whole problem.
  - Need a reasoned division of responsibility.
-

# More requirements

---

Security must be usable (so it gets used) and understandable.

There are different needs for security in different contexts.

We need an *architecture* for security.

- But note the point above.
- Architecture that supports late binding.

Security for tomorrow's devices.

Sensitivity to social context and needs.

---

# Resilience and availability

---

Security community has tradition of looking to resistance. Resilience may be a better path.

- Diverse failover modes
  - Reduced interdependence under attack
  - Integration with management
    - No silent failures
  - Support for variability
  - Resilient social structures
  - Other disciplines?
-

# Deterrence

---

Social form of question: what is the role of policing in the Internet?

Technical form of question: what should it be possible to see where?

Models of policing:

- Wait to be called.
    - Can end-node gather evidence? Witnesses?
    - Can application design prevent classes of crime?
  - Feet on the street, cameras.
  - CDC
  - Contract law and arbitration.
-

# Identity

---

At the packet level:

- If we separate identity from location, then what?
- Regional variation?
- Access to resources?

At the app (e2e) level.

- Cross application architecture?
  - Bottom up or top down?
  - Should “the net” play a part at the application level?
    - The “out-source” model of protection.
    - What about e2e at the application layer?
-

# Denial of Service attacks

---

Proposal: distinguish “public” and “closed” servers.

For public: must diffuse.

- Speculation: diffusion will be key part of future.

For closed, outsource protection.

- Who do you trust?

Possible research questions:

- Do private address spaces help?
  - Virtual nets?
    - Must protect the real assets underneath...
  - Re-architect protocols for these goals?
-

# Protecting the end node:

---

The negative availability principle

- Architecting the firewall.
- Relate to identity (“Danger, danger...”)

The virtual machine story

- Do virtual machines need virtual nets?

Helping the host protect itself.

- Trusted path, logging, reference monitor.

Avoid the two fates?

---

# Virtualization and security

---

What do virtual networks really buy us?

- Will we have a few or millions?

What do private networks really buy us?

What do overlay networks really buy us?

- What are the necessary security requirements for the underlay?
-

# Close links to management.

---

Protocol design for management and security.

No silent failure.

Make management systems secure.

- Hard--they must work when all else is failing...

Goof-proof tools

- Necessary to boost availability.

Security should slow events to the point where humans can intervene.

---

# Application level security

---

Provide design patterns for secure design.

- Design so that first interactions can be outsourced and diffused.
    - Validate identity, etc.
  - Design to gather evidence consistent with risk.
  - Give user control over selection of services “in the application”.
    - Outsourcing.
-

# Another case--making money

---

Why do ISPs exist in the form they have?

The protocols induce an industry structure.

- We did not fully grasp this in 1975...

What are the characteristics of their business?

- Sunk cost.
- Commodity.
- Competitive.

This (if correct) is not healthy.

---

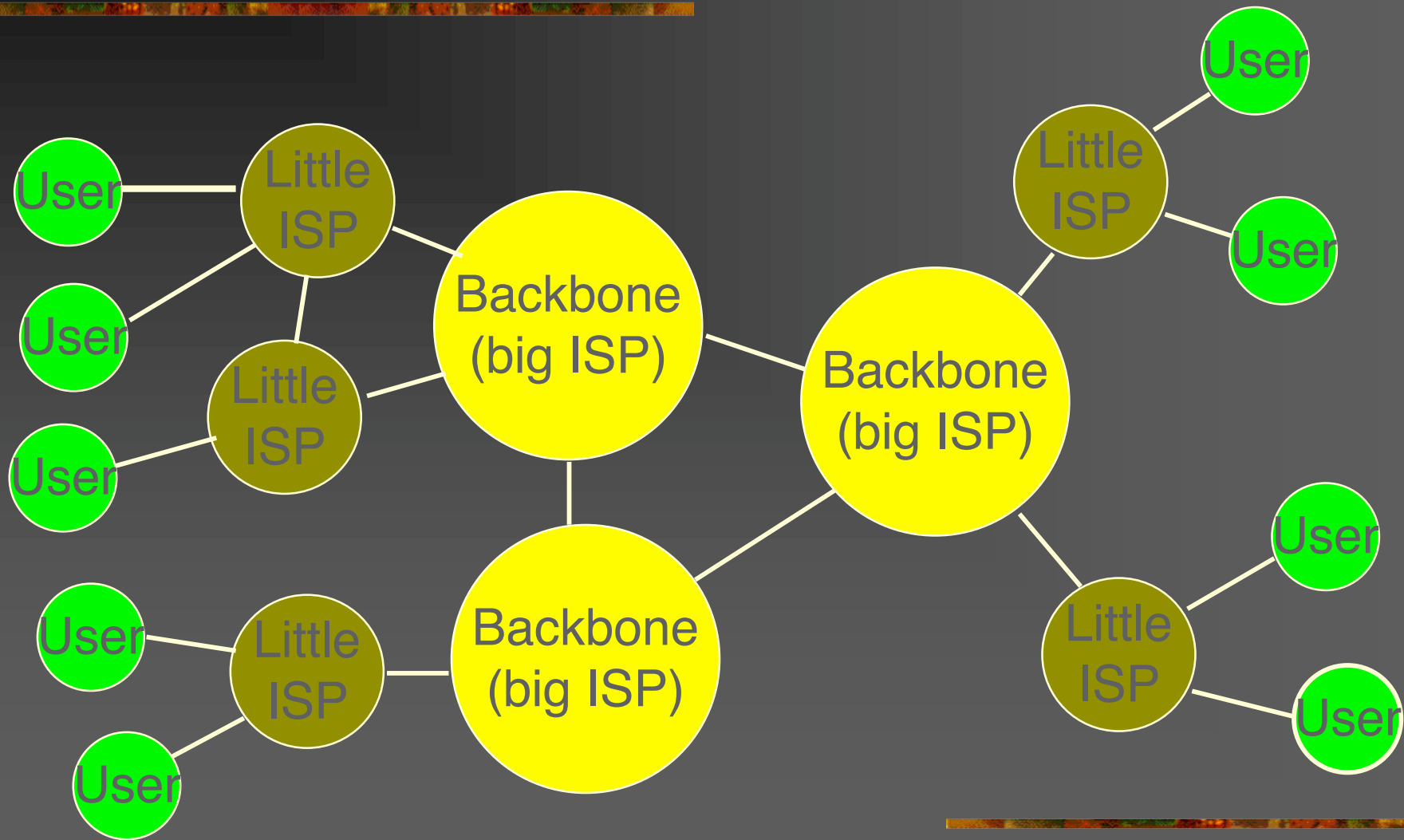
# A fundamental problem

---

On what basis should competing ISPs interconnect so that the global Internet can happen?

- They have to interconnect.
  - They are fierce competitors.
-

# The traditional Internet picture



# What constrains that picture?

---

Money routing.

- Packets are an excuse to make money...

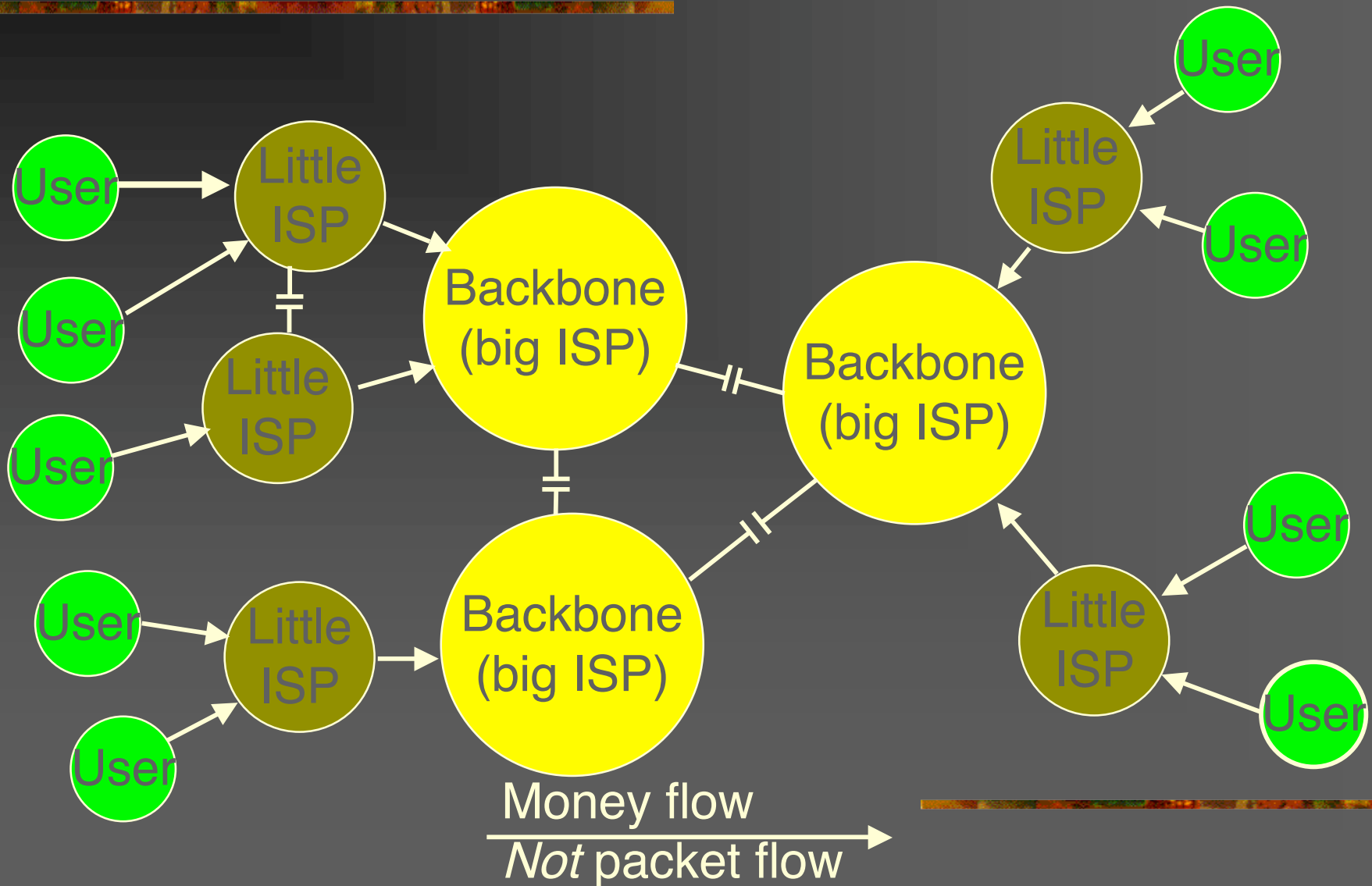
And old and possibly true story.

- “I thought *you* were going to pay *me* money.”
- Or: how not to trade in a car.

The result: revenue-neutral peering, or “money insulators”.

---

# The money picture



# The two modes

---

Transit: small ISP pays large ISP to deliver packets to/from anywhere.

Peering: two ISP agree to exchange packets for free.

- Normally, only packets destined for each ISP and its transit customers.
  - Normally, no payment.
-

# Dig deeper--why?

---

Internet has no expression of value flow.

- No “800” numbers.
- Packet flow not the same as value flow.
- (No concept of a “call”.)
  - We were proud of that.

So, two rough arguments.

- 1) You were going to get the traffic anyway.
  - 2) Some sort of symmetry.
-

# What is *really* wrong?

---

It is not just the inefficiency of peering.

- But note the recent posturing from SBC

It is the inability to create and offer new services.

Evidence:

- The (non-)history of Quality of Service.
  - Akamai
-

# The phone company story

---

Very different history.

- Interconnection is regulated.
- Simple, well understood service.
- Different revenue model (sort of).
  - Access charges and settlements.

Question for discussion: should we regulate Internet interconnection?

---

# The “consumer problem”

---

Specific focus on consumer broadband.

Today: fixed pricing. \$40/month (or so) independent of usage.

This creates cost pressures and perverse incentives.

New applications → higher usage → higher cost, but constant revenues, so lower profits. Why would an ISP love a new app?

---

# Some *rough* estimates

---

Of that \$40/month, \$2 might be for usage.

Heavy users may generate 10x the traffic of average users.

So a heavy user may cost an ISP ~ \$20/month. Solution: block heavy users!

- These usage costs are for wide area traffic.

This behavior is a barrier to some innovation.

---

# Death of distance--not

---

Distance did not die. It is not even badly injured. It is only sleeping.

For some content (such as IPTV) current costs do not permit large-scale wide area transport.

- Content will “have to be” staged locally.
- Then costs go to zero.

Raises new forms of blocking:

- Logical or physical co-location (hosting).
-

# Three futures

---

Bad future 1: rising costs → blocking of new apps  
→ reduced innovation → stagnation of the value  
proposition → no growth.

Bad future 1a: Vertical reintegration → restricted  
innovation → etc.

Bad future 2: open access → commoditization of  
ISP → no facilities investment → stagnation in  
capacity → no growth.

Good future: avoid both those fates.

Make money from apps ISP does not own.

---

# How might that happen?

---

Sell enhanced network-level services such as enhanced QoS.

- Is this better than vertical integration with apps?
- Sell to whom?

Find a new pricing model for packet carriage.

- Usage tiers? Weekend splurge pricing?

Sell applications that are “naturally” local.

- Backup

Sell application-support services.

- Caching, security. *Actively open.*
-

# Barriers to being *actively open*

---

## Application design:

- Design to benefit from services
- Design to avoid monopoly.

## Indirect sources of revenue:

- Consumers don't usually pay today

## Coordination problems to offer the service platform.

- Architecture, revenue sharing, performance
-

# Who let Akamai in?

---

Akamai is a third-party provider of content distribution.

- Global set of managed Web caches.
- Proprietary methods to improve performance
- Service commitments.

Akamai monetized what ISPs were doing for free.

- Identified the content producer as customer.
- One stop shopping.

Illustrates barriers for ISP entry.

- Coordination, indirect source of revenue.
-

# Good or bad?

---

Bad: ISPs could not exploit opportunity, left revenues on table.

- Dive to commodity packet carriage.

Good: Akamai moves money across the Internet.

- Jumps peering points (revenue neutral boundaries).
  - ISPs get (or save) money, without having to solve coordination problems.
-

# What could a new Internet do?

---

Express direction of value flow.

Re-design the ISP interconnection protocols.

Offer design patterns for the “economic design” of applications.

Provide tools for service composition.

---

# Generality vs. specificity

---

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.

High-level requirements for any FIND architecture.

---

# 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

---

Two approaches to specification.

- Per-hop behavior (PHB), composable along a flow to get overall semantics.
  - 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 Per Hop Behavior.

- 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.
  - We have lots of honed tools for this purpose:
    - End-to-end arguments.
    - Run-time application adaptability.
    - Weak semantics.
    - Open interfaces.
-

# Fundamental challenges

---

## Mixing today and tomorrow

- Can we perhaps do both?
- Design building blocks, and methods to assemble them?

## Architectural stability

In the past, change has implied erosion of architecture.

- Can we design *architecture* for change?
  - This goal is somewhere between FIND architecture and science of design.
-

# My proposal for a design goal

---

We should design for tomorrow--design for change and for the application we have not seen yet.

We should pay more attention to how building blocks are composed into services.

We should think about how an architecture can itself survive change.

---