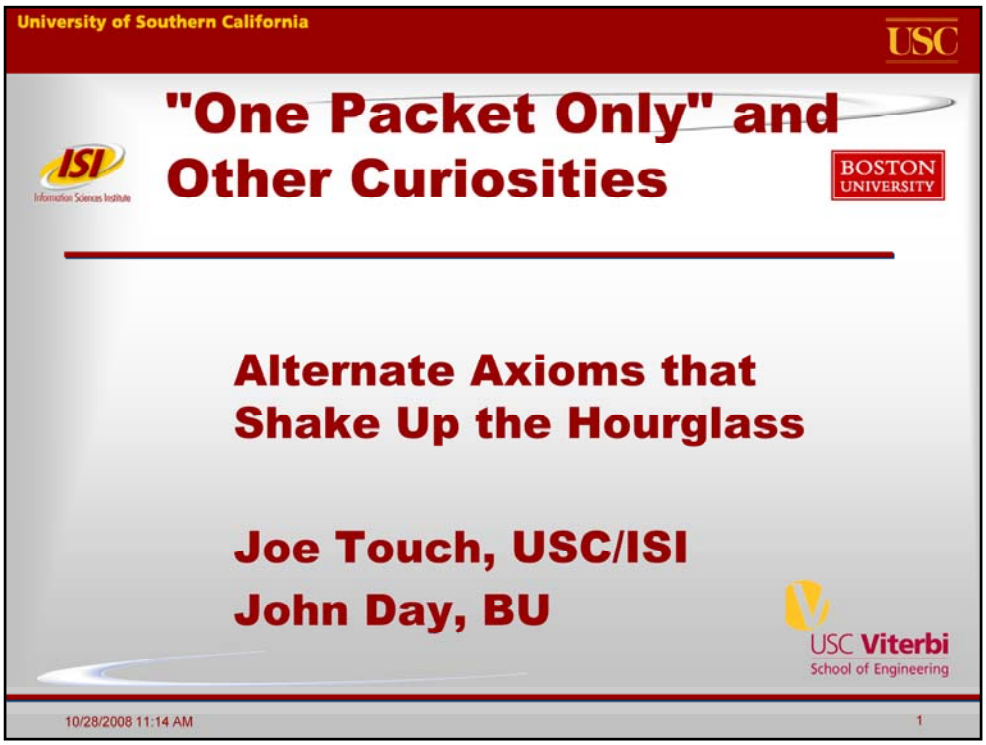


This is a 'tag-team' moderated discussion, in which Joe and John presented alternate issues to the attendees. The slides are labeled in the bottom right as to who presented each slide.

The presentation ended on slide #16, but the additional slides prepared are provided anyway.

Goal

- Perturbation analysis of network architecture
  - Change some assumptions
  - Explore the consequences
  - Categorize:
    - Fundamental network property
    - Desirable optimization
    - Unnecessary limitations

Axioms

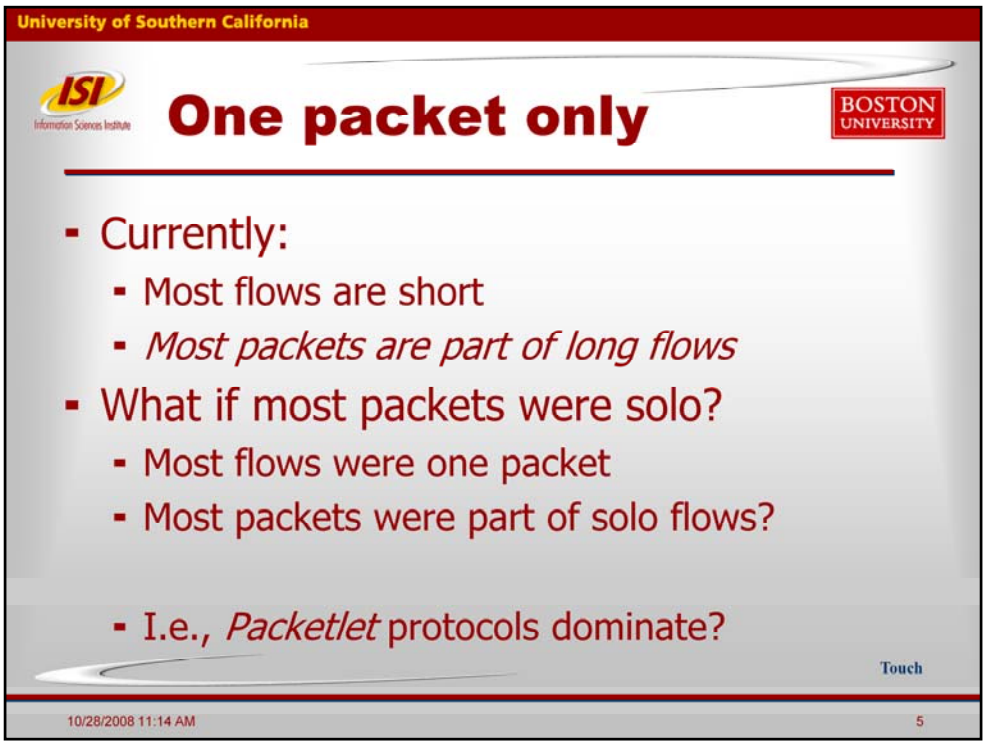
- Properties:
  - Assertion accepted as true without proof
  - Considered to be self-evident
- Vs. Tenet
  - Assertion accepted as true without proof
  - Not self-evident
- Goal: distinguish axioms from tenets!

Two Groups of Axioms

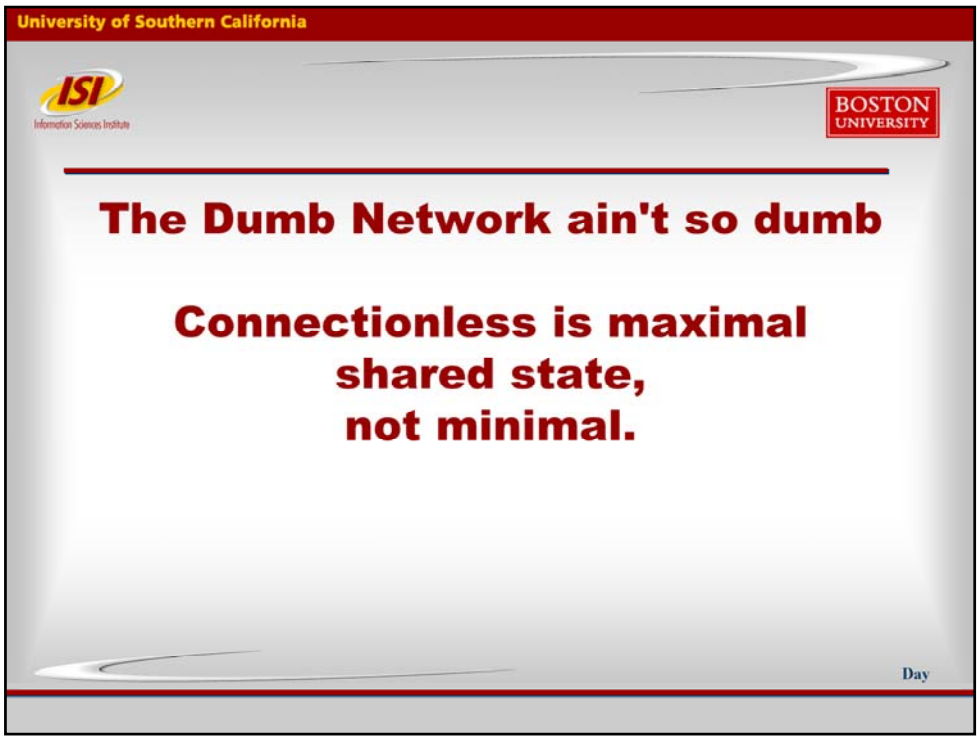
- Principles
  - Axioms of design
- Applicability
  - Axioms of context
- Both are architectural

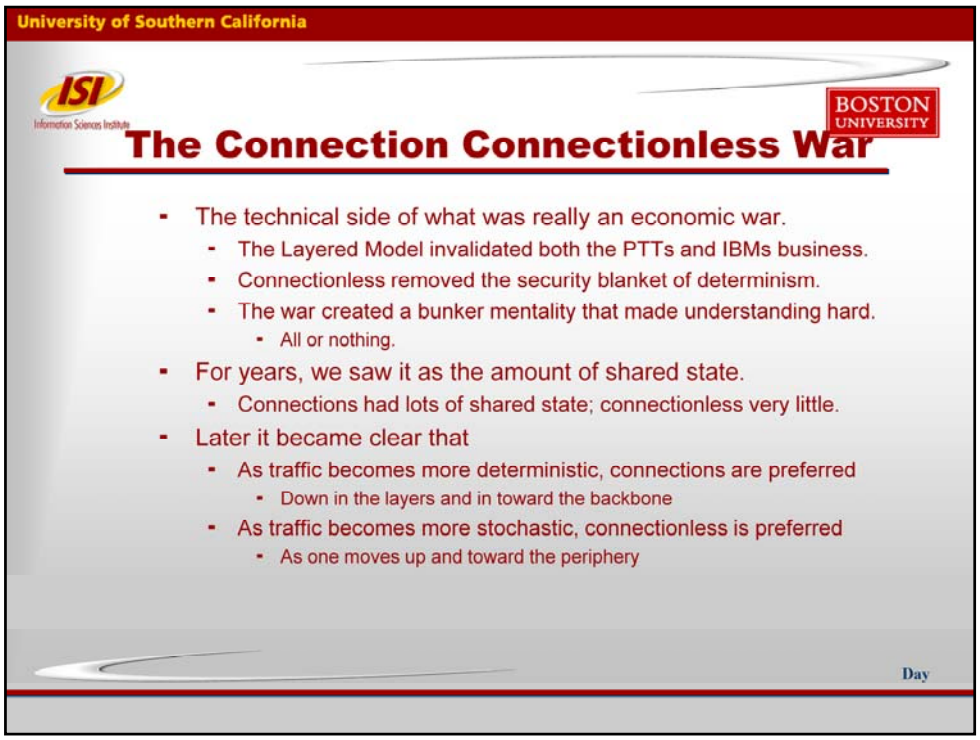
NB: distinction isn't critical, but helps John and me tag-team… ;-)

Touch



Just as a wavelet is a single wave in isolation, a packetlet is a single packet in isolation.


University of Southern California
ISI
Information Sciences Institute
BOSTON
UNIVERSITY
The Dumb Network ain't so dumb
Connectionless is maximal
shared state,
not minimal.
Day

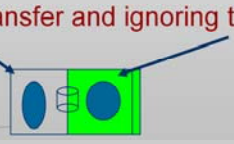


This is the real breakthrough in networking, not packet switching, but connectionless. The inspiration of Louis Pouzin and the basis for the CYCLADES network. This is what created the major threat to the phone companies. But we need a synthesis that removes the oil and water solutions. Like those insights that the ARPANet guys were so good at. It became clear from the mechanism and policy analysis that the amount of shared state didn't yield enough of a "continuum" to to make for a useful synthesis. What then?

University of Southern California

Finding a Synthesis is Hard

- Lets look at this very carefully
- What makes connection-oriented so brittle to failure?
  - When a failure occurs, no one knows what to do.
  - Have to go back to the edge to find out how to recover.
- What makes connectionless so resilient to failure?
  - Everyone knows how to route everything!
- Just a minute! That means!
  - Yes, connectionless isn't minimal state, but maximal state.
    - The dumb network ain't so dumb.
  - Where did we go wrong?
- We were focusing on the data transfer and ignoring the rest:

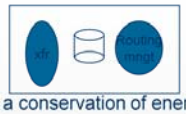


Day

For a long time we looked at it as a continuum on the amount of shared state with co/cl as the extremes: connectionless didn't have much, connections a lot. But there weren't many (any) points on the line.

University of Southern California

We Need to Look at the Whole Thing



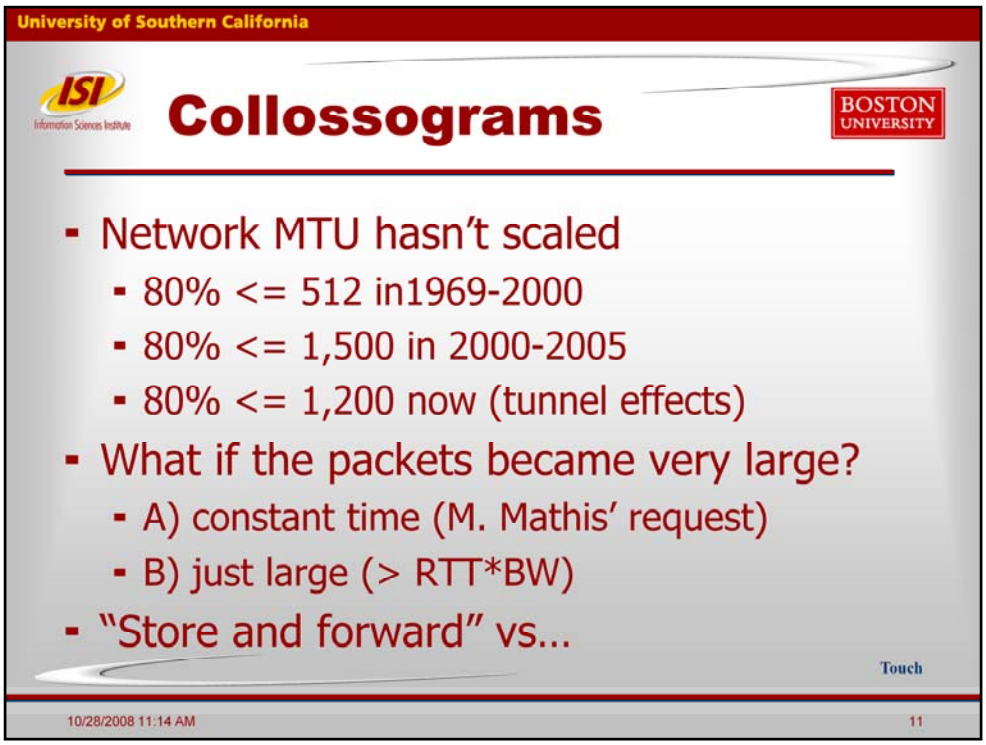
(A bit like doing a conservation of energy problem and getting the boundaries on the system wrong.)

- The amount of state is about the same, although the amount of replication is different.
- We have been distributing connectivity information to every Node in a layer, but
- We have insisted in distributing resource allocation information only on a need to know basis, i.e. connection-like.
  - Even if we aren't too sure who needs to know.
- Now we have to work out how to do resource allocation more like how we do routing. (Left as an exercise.)

So What Do We Know About CO/CL

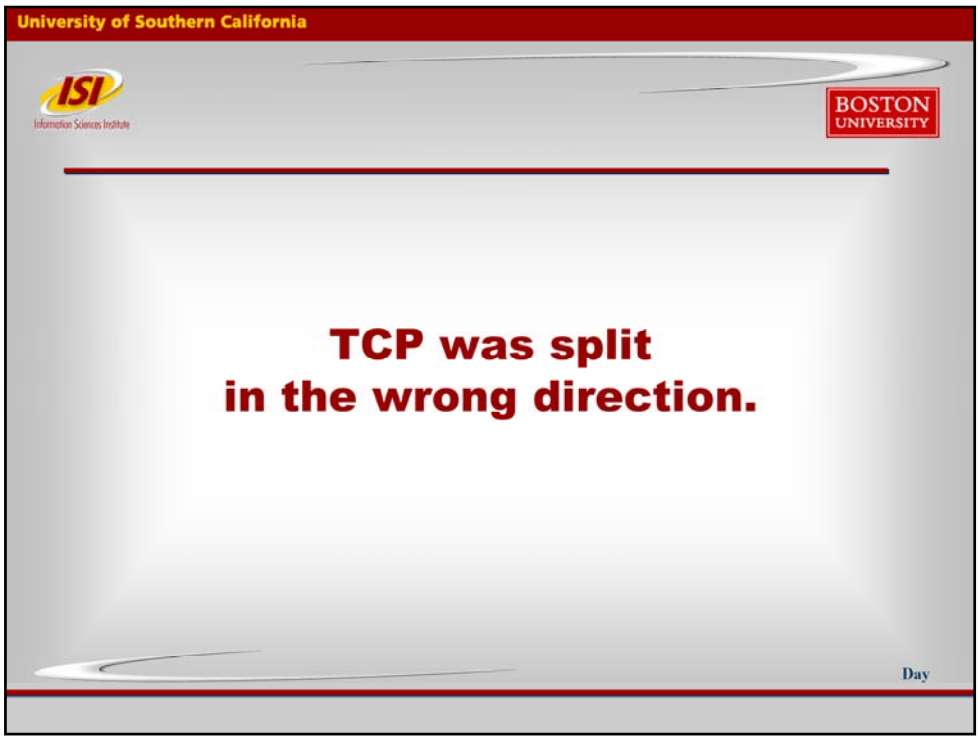
- It is a function of the layer. Should not be visible to applications.
- Connectionless is characterized by the maximal dissemination of state information and dynamic resource allocation.
- Connection-oriented mechanisms attempt to limit dissemination of state information and tends toward static resource allocation.
- Applications request the allocation of comm resources.
  - The layer determines what mechanisms and policies to use.
  - Tends toward CO when traffic density is high and deterministic.
  - CL when traffic density is low and stochastic.

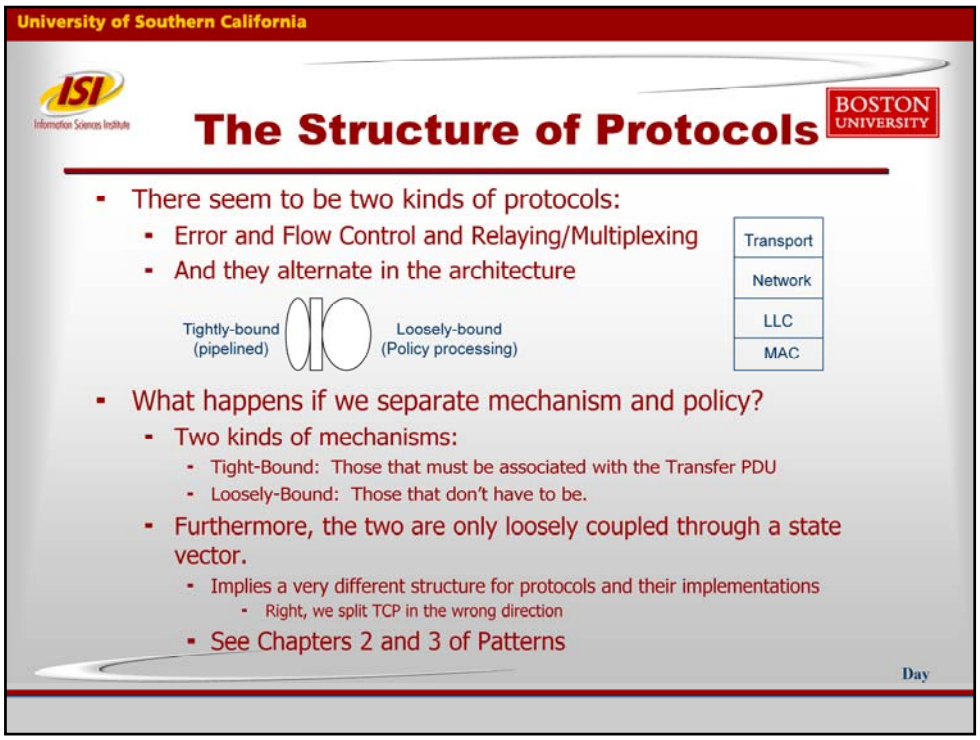
- See Chapter 3 of Patterns.



80% of the packets in the network were <= 512 bytes in the early Internet.

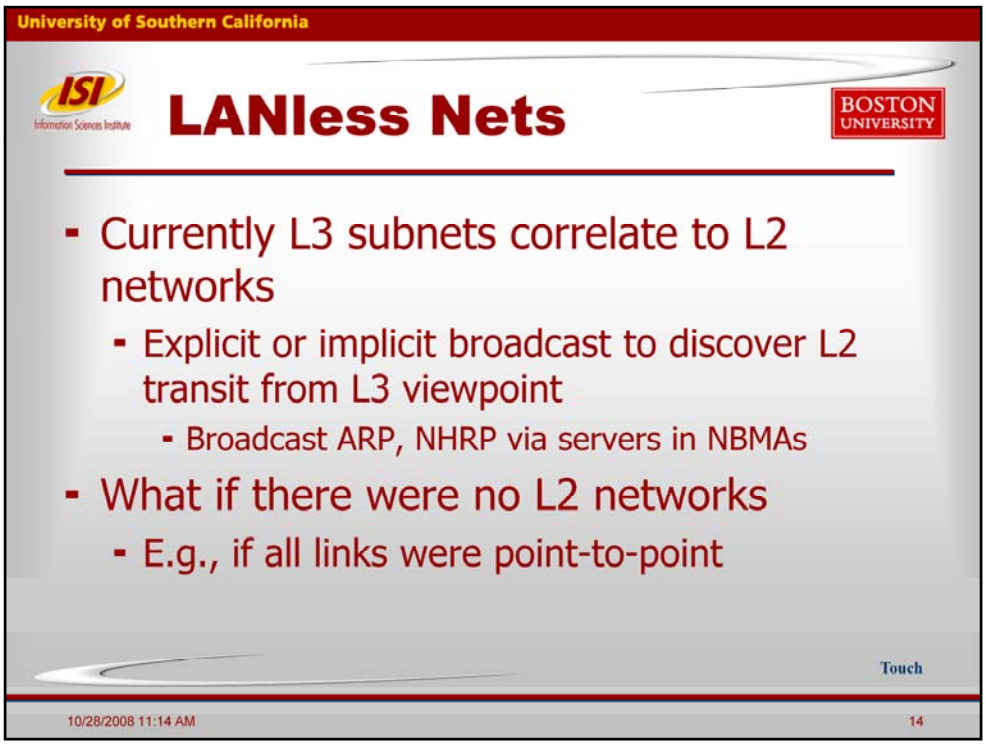
80% were under 1,500 for a while in the beginning of the decade, but dropped to anticipate room for VPN and tunnel headers (to avoid the need for ICMP "too big" feedback, i.e., path MTU discovery, which often fails because ICMPs are often blocked for security reasons). These numbers are from CAIDA studies.


University of Southern California
ISI
Information Sciences Institute
BOSTON
UNIVERSITY
TCP was split
in the wrong direction.
Day



Protocols like HDLC and TCP have a lot of similarities. As do the MAC protocols and IP and mail.

This tells us a lot about what the protocol should look like. Further, for protocols near the media we can expect the characteristics of the media to dominate the choice of policies; for protocols near the applications, applications dominate. Explains why we can do successful data link protocols but have never been satisfied with our transport protocols. Transport protocols support applications (many) while data link protocols are tailored to the media (one at a time). By not separating mechanism and policy we were implicitly expecting one point in a roughly 8 dimensional space to solve all our problems. Using delta-t as a guide yields further simplifications.



See RFC 3819 – advice for subnet designers, which showed that our protocols assumed certain L2 properties.

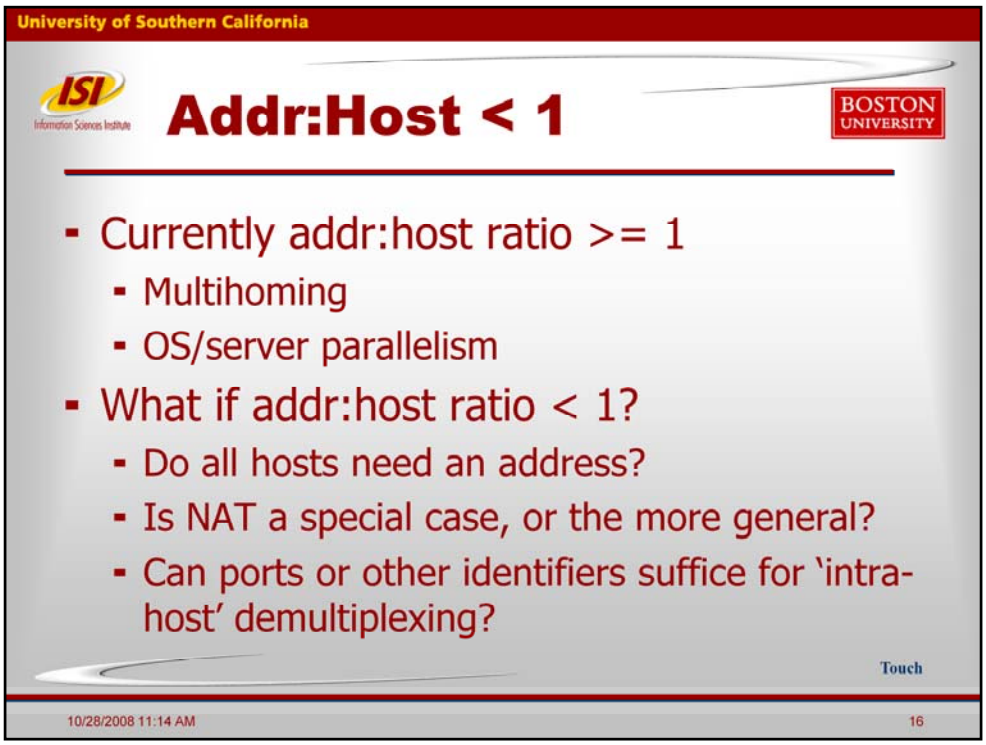
NHRP = next-hop resolution protocol (and ARP emulator)

NBMA = non-broadcast multiaccess networks, of which ATM LANE (LAN emulation) has been a primary example.

Unicast-only

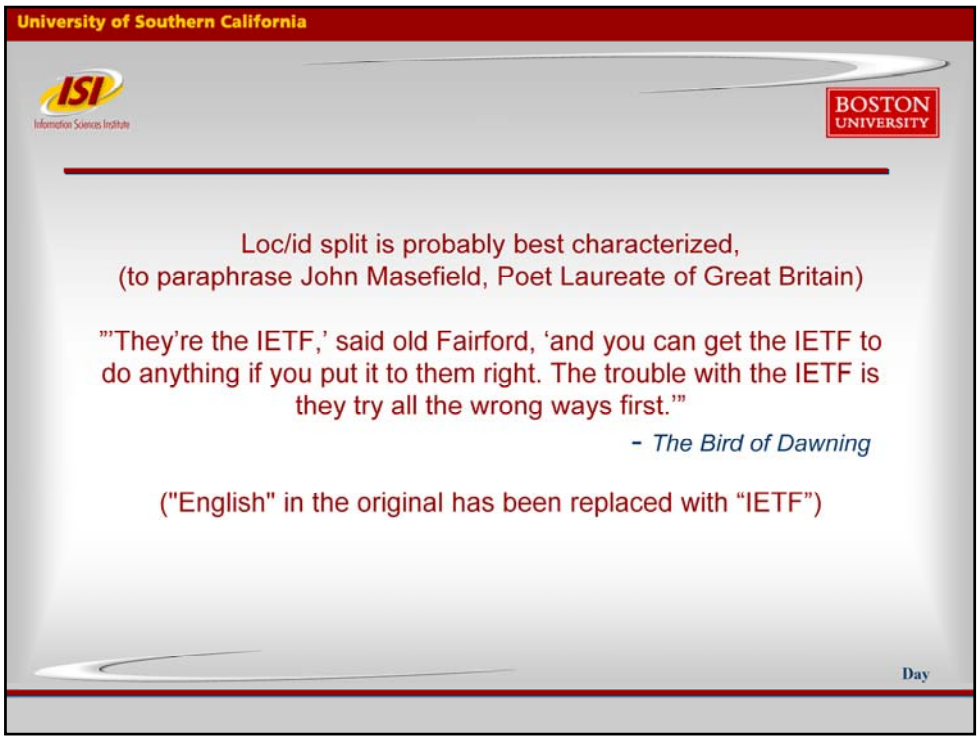
- Currently we rely on broadcast and/or multicast
  - L2 transit discovery
  - Device configuration
  - Resource discovery
  - Efficient multipoint data distribution
- What if there were no broadcast/multicast support at all (even emulated)?

Touch



NATs look like hosts to the rest of the world, but like “transparent” routers (they don’t decrement the TTL, but otherwise tend to follow router rules) to the NAT’d subnet.

NOTE: This was the last slide discusss

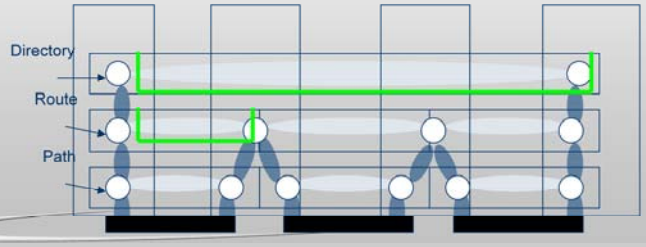


We have known about this problem since 1972. We immediately knew the answer, but Saltzer finally explained it in 1982.

University of Southern California

Saltzer Provided the Answer in 1982

- Directory maintains the mapping between Application-Names and the node addresses of all Applications reachable without an application relay.
- Routes are sequences of node addresses used to compute the next hop.
- Node to point of attachment mapping for all nearest neighbors to choose path to next hop. (Saltzer missed this because they hadn't occurred yet.)
- This last mapping and the Directory are the same:
  - Mapping of a name in the layer above to a name in the layer below of all nearest neighbors.

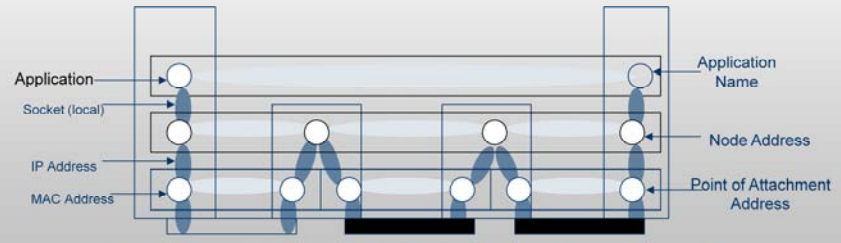


Day

University of Southern California

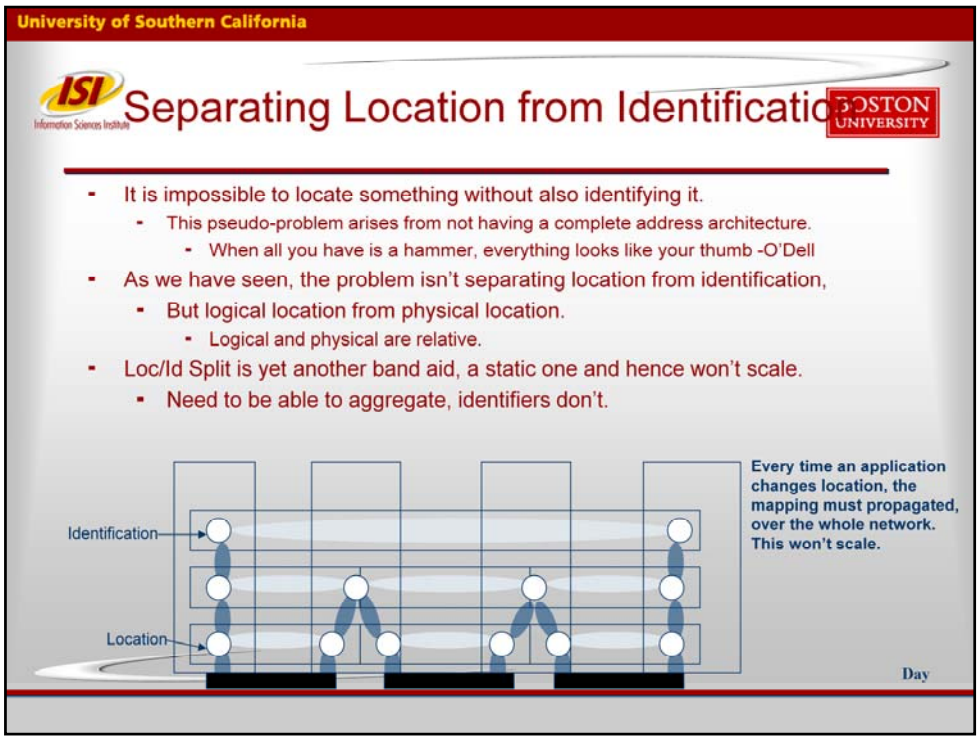
Applying Saltzer to the Internet

- The most striking feature is that half of the addressing architecture is missing.
  - No wonder there are addressing problems.
  - The only identifier we have for anything is the IP address.
- There are no node addresses and no application names.
  - And the point of attachment is named twice!
  - Domain Names are synonyms for IP addresses. URLs are pathnames through the stack and location dependent.



As if your computer worked only with absolute memory addresses.

Day



The identifer names an "endpoint." The endpoint can only be an application-entity-instance.

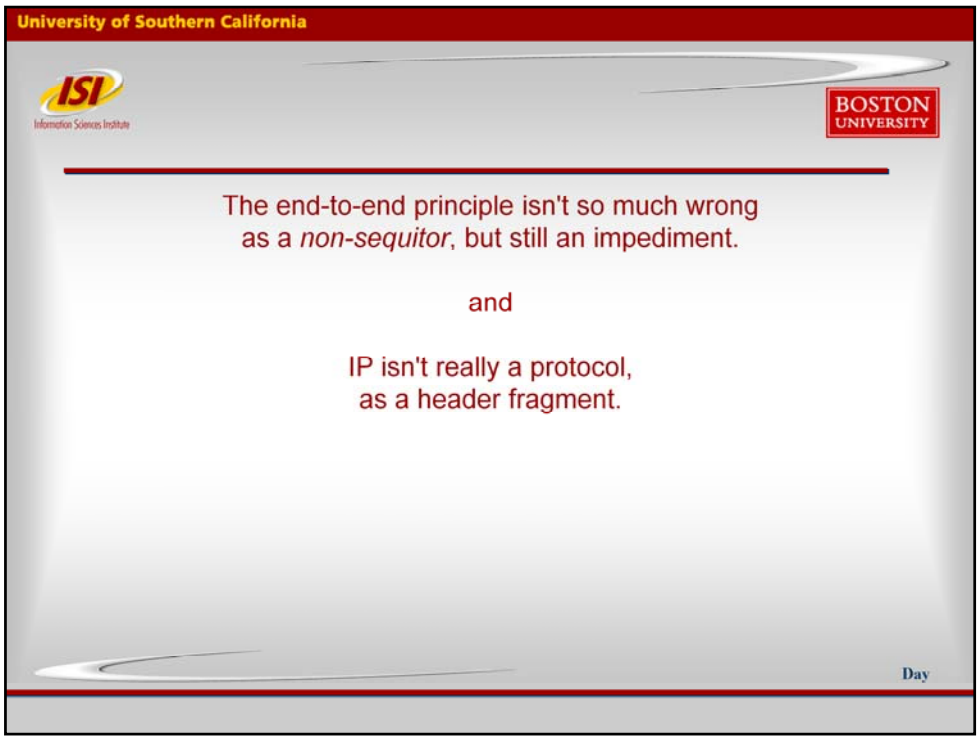
This takes us from a quarter of an architecture to a third an architecture.

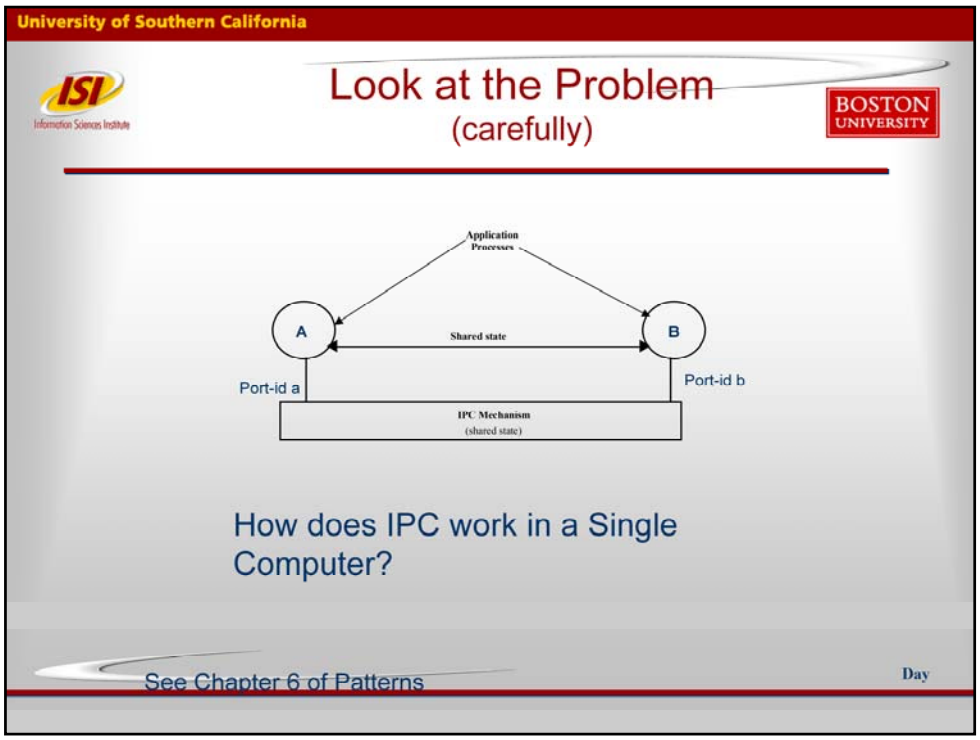
Multidimensional names

- Names today:
  - Static, global, uniform (single space)
- What if names were multidimensional?
  - Security level
  - VPN-ID
  - Time

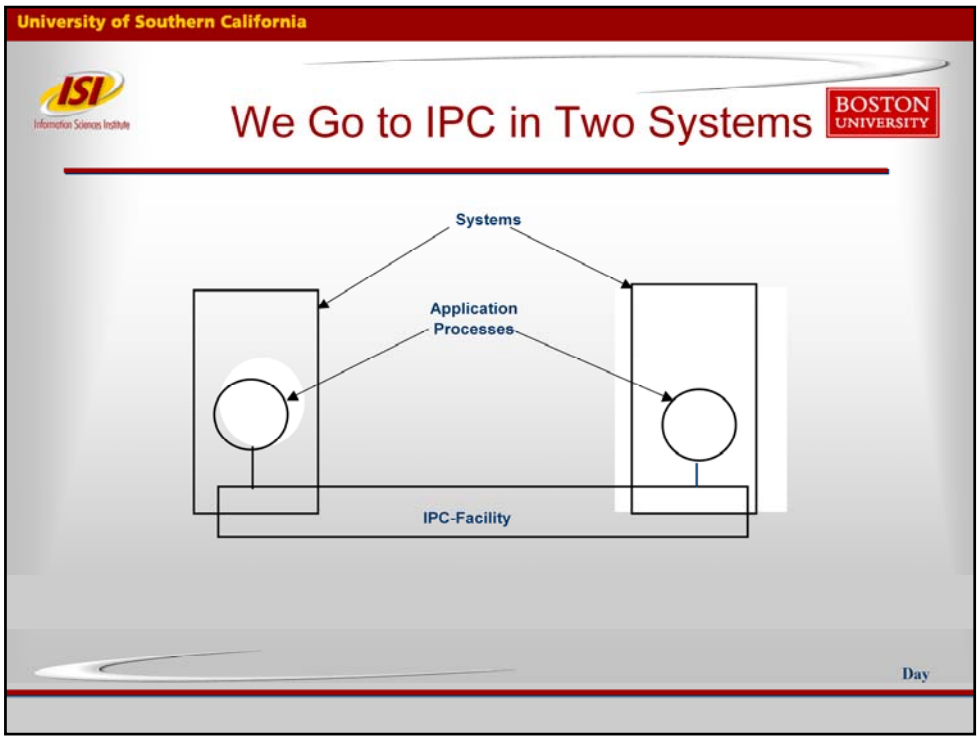
Multidimensional names

- Names today
  - Static
  - Global
  - Coarse structure (ucast/mcast)
- What if names were multidimensional?
  - VPN-IDs as more than a shim
  - Security level (recent IETF discussion)
  - Time (duration of validity)

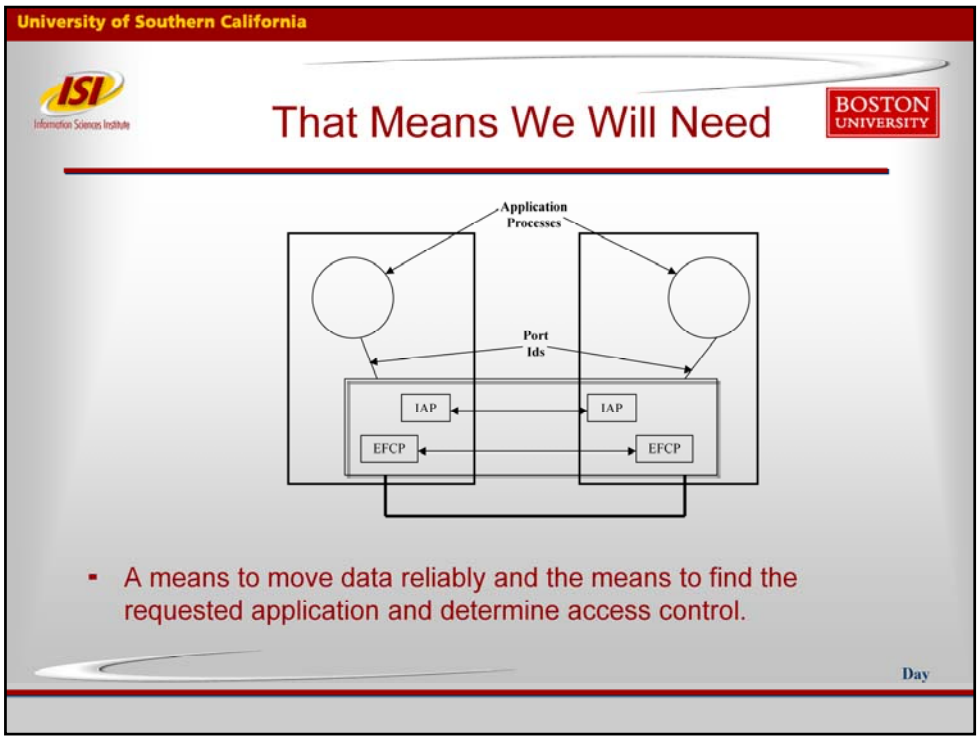

University of Southern California
ISI
Information Sciences Institute
BOSTON
UNIVERSITY
The end-to-end principle isn't so much wrong
as a *non-sequitor*, but still an impediment.
and
IP isn't really a protocol,
as a header fragment.
Day



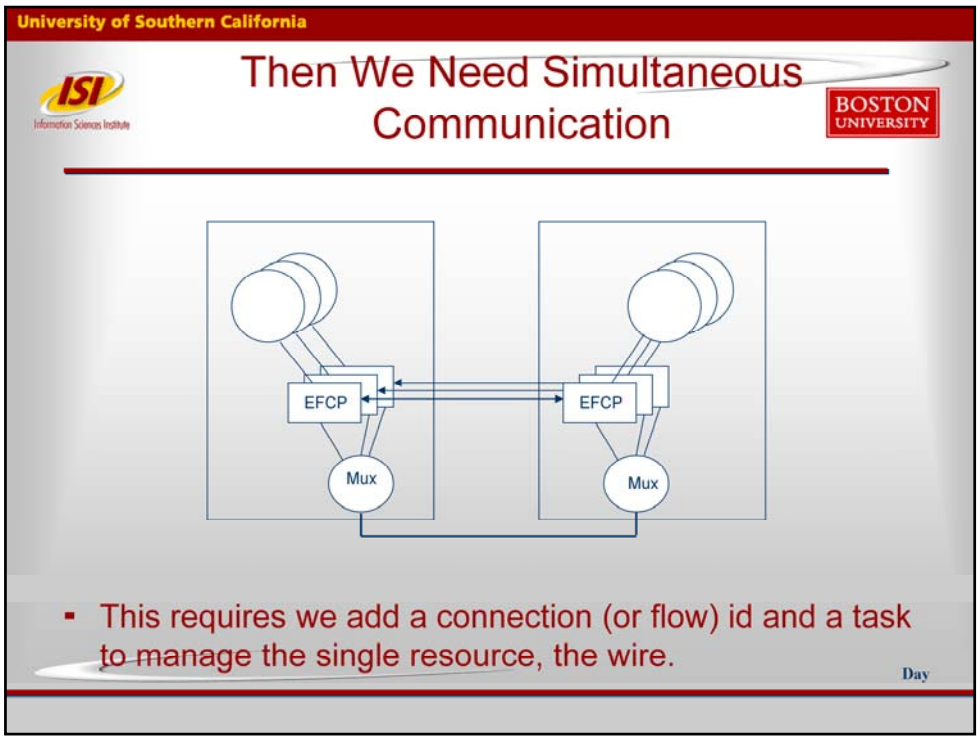
I wish I could tell you that I had this brilliant insight and my superior intellect immediately saw what we had been missing. Unfortunately, that is not the case. Someone asked me a question about protocols and I didn't like the answer I gave. So I came up with this story to explain it. Then I realized what it said. There is another presentation that goes through this in detail. Here we will just skim it.



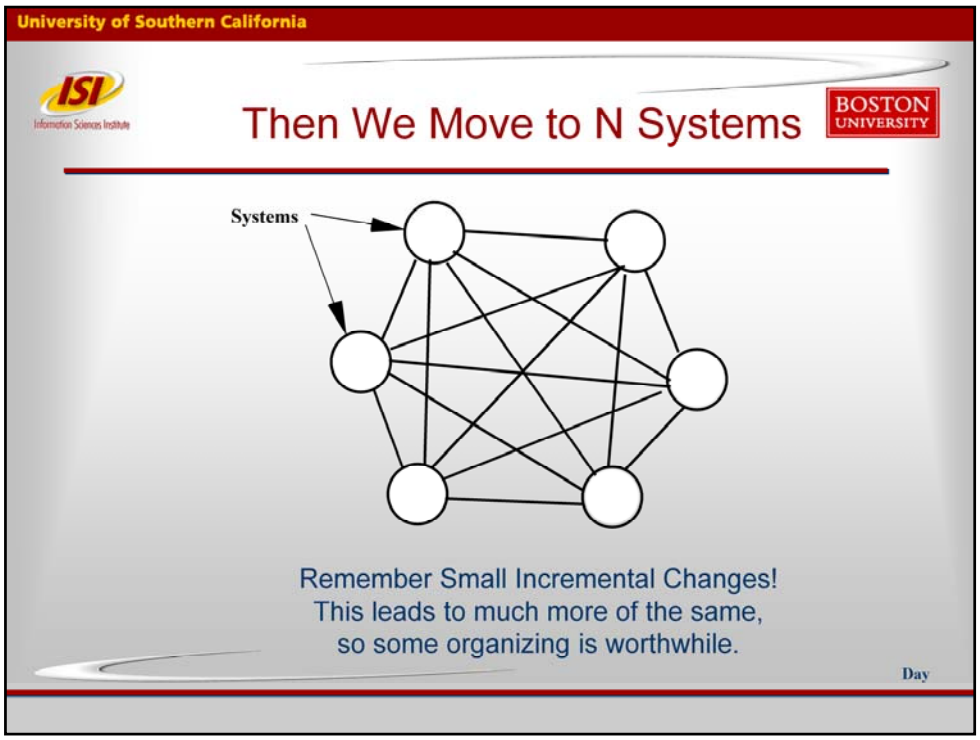
We have lost several capabilities and must create functions to do them. We can no longer see all of the available applications, we can no longer rely on OS access control for everything, and we need some means to transfer data between machines reliably.



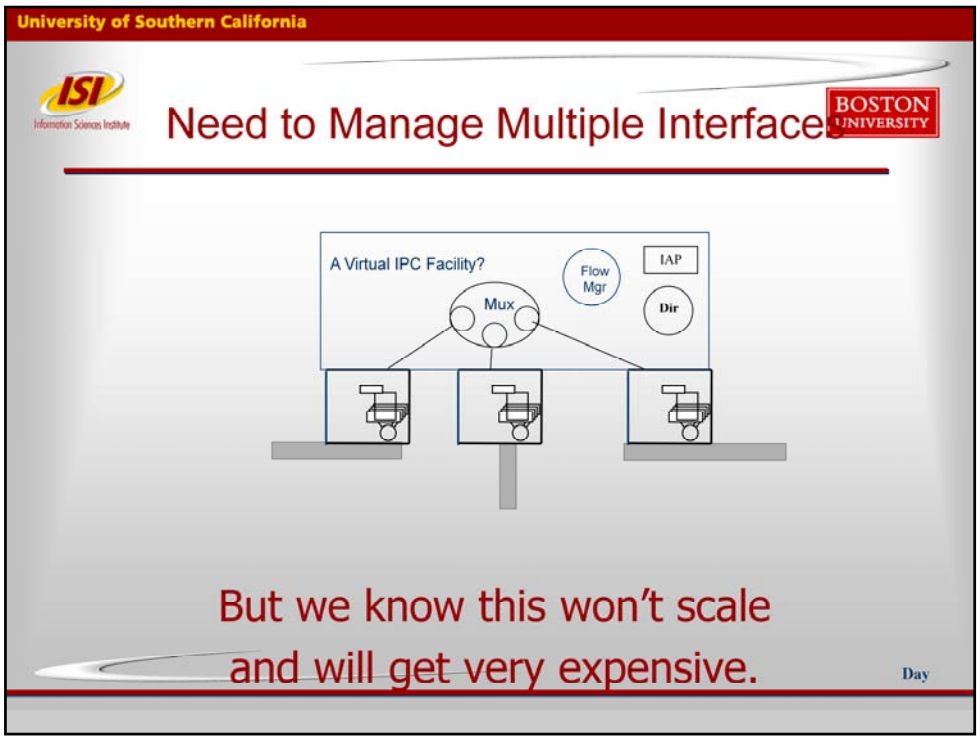
First we need a protocol for finding if the application is on the other system and whether we have access to it (an IPC Access Protocol). But then we need a protocol to get that information there (some sort of error and flow control protocol). Once we have that then, . . .



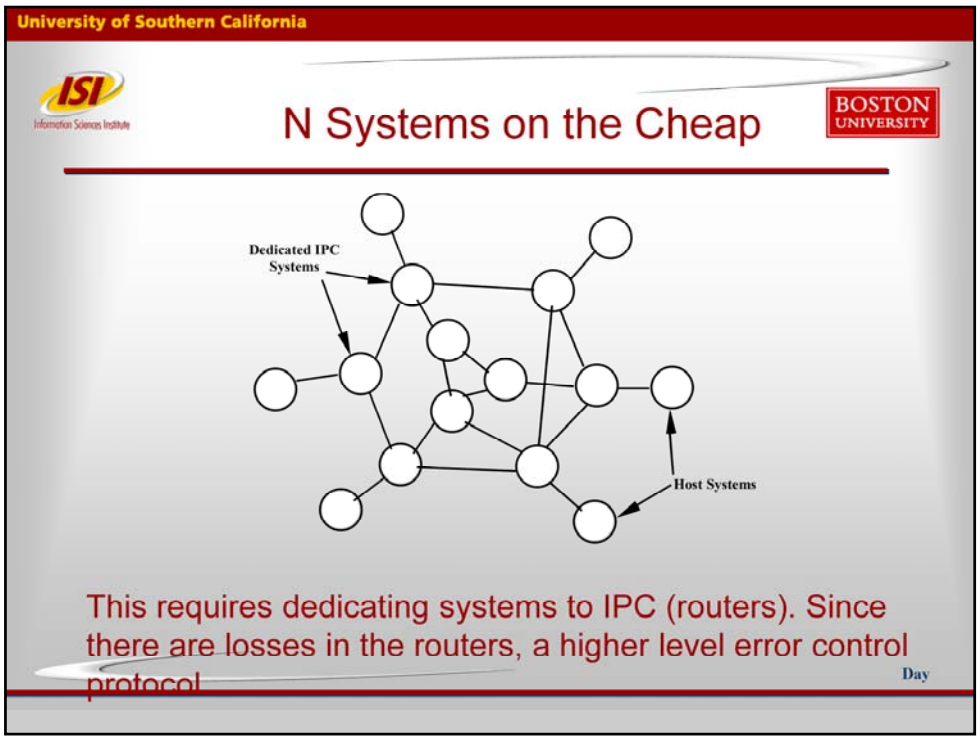
Connection-ids have traditionally been formed by concatenating the local port-ids. We must add a connection-id field to our EFCP.



This will require each system to have the stuff we just created for each wire. But,



But it has provided something we can build on. We create a second level simply to hide the complexity from the user.



Now we need addresses and a higher level error control protocol

University of Southern California

Communications on the Cheap

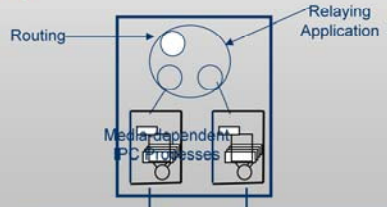
BOSTON UNIVERSITY

- We will need relaying and multiplexing.
- That requires some new elements:
  - Globally accepted names for source and destination muxing apps.
  - Need routing applications too, which will need to exchange information on connectivity.

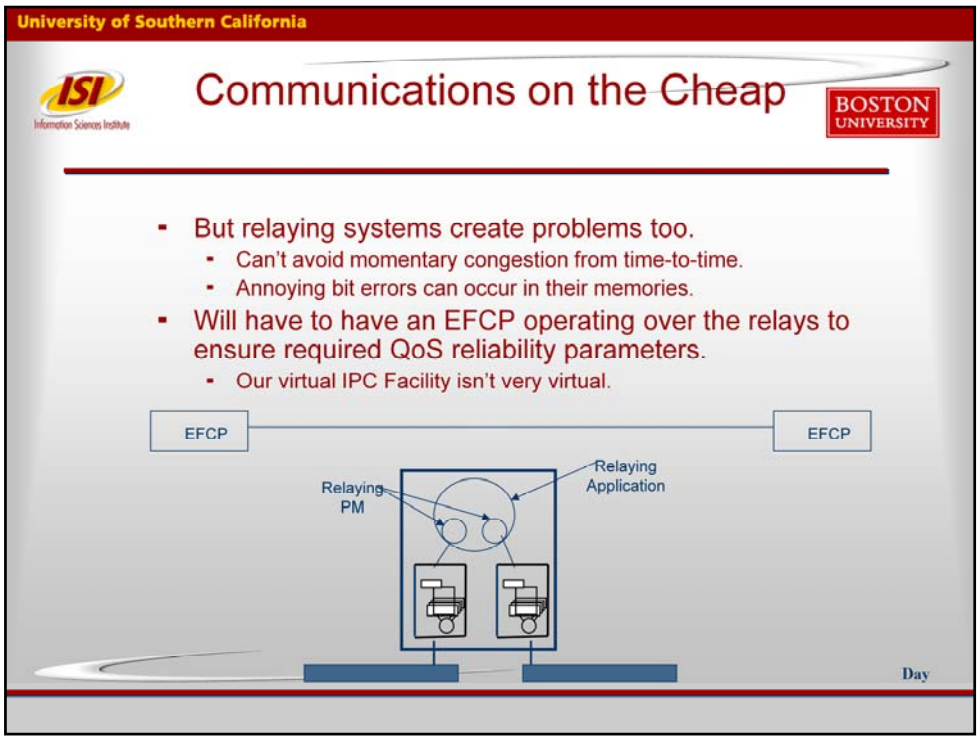
Dest Addr	Src Addr

Common Relaying and Multiplexing Application Header

- Will need a header on all PDUs to carry the names for relaying and multiplexing.



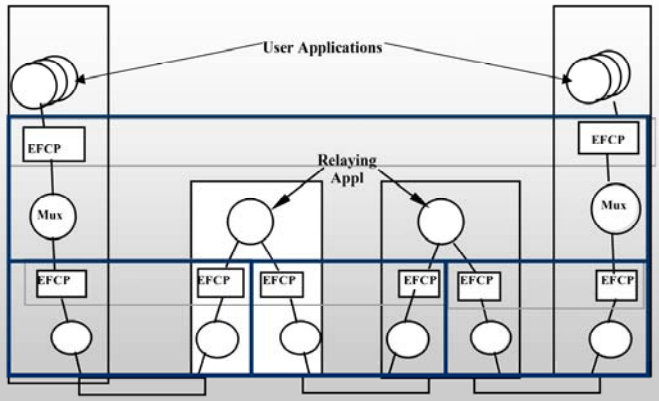
Day



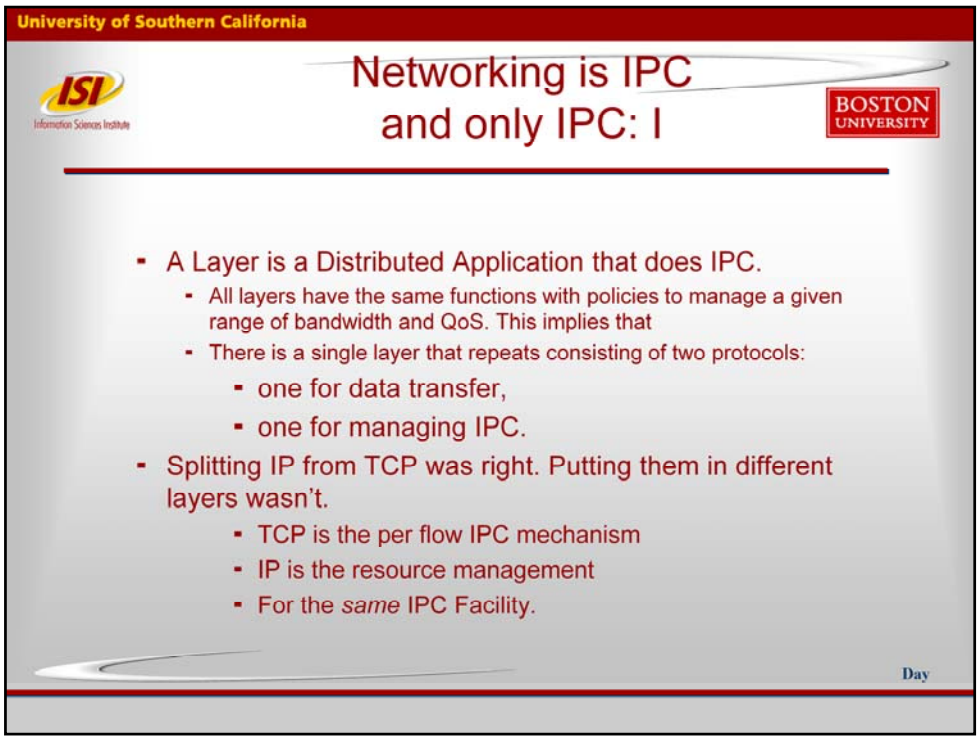
Lets step back and look at the larger picture we are constructing.

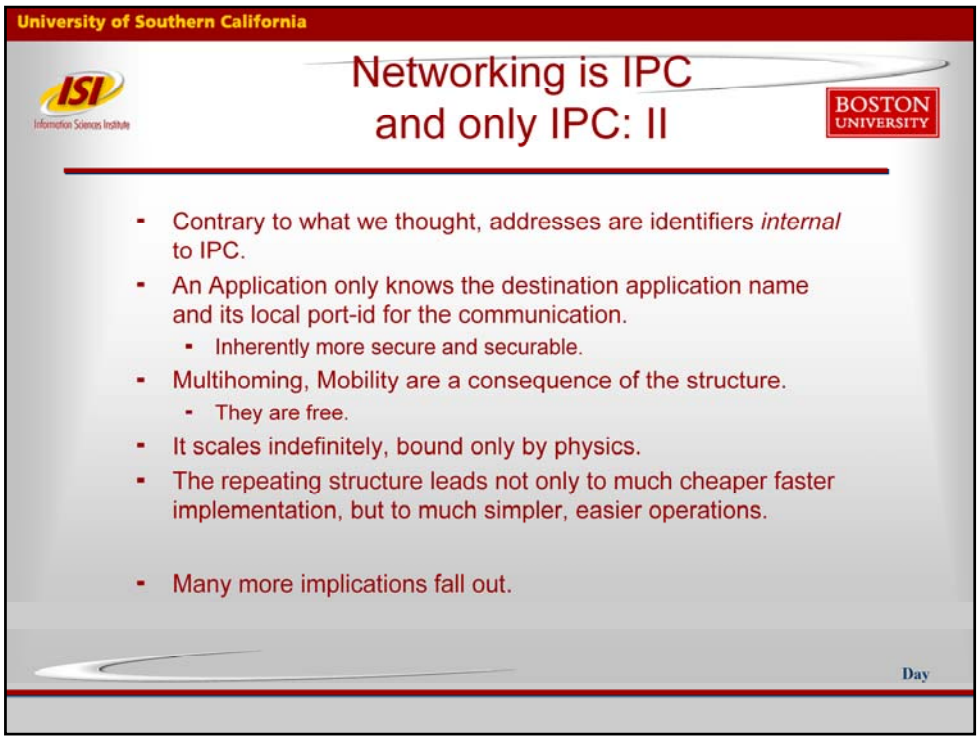
University of Southern California

The IPC Model



Day


University of Southern California
ISI
Information Sciences Institute
Networking is IPC
and only IPC: I
BOSTON
UNIVERSITY
A Layer is a Distributed Application that does IPC.
All layers have the same functions with policies to manage a given range of bandwidth and QoS. This implies that
There is a single layer that repeats consisting of two protocols:
one for data transfer,
one for managing IPC.
Splitting IP from TCP was right. Putting them in different layers wasn't.
TCP is the per flow IPC mechanism
IP is the resource management
For the same IPC Facility.
Day



This structure and that it repeats is more secure, even before we add specific security mechanisms.