Networking is IPC and only IPC or How to Clean a Slate

John Day

2010

Good architecture, like good science, is maximizing the invariances and minimizing the discontinuities.

The Internet is Facing Severe Problems: I

- Security is essentially non-existent.
  - Excuse: No one considered it in the early days
    - Security wasn't a concern for a military-funded network?
  - Actual: Systematically weak design
- Router table size is growing exponentially
  - Excuse: Memory is cheap
  - Actual: No longer on Moore's Law, it is getting expensive and caused by
- No support for multihoming
  - Excuse: not that many hosts need it, and we can kludge it
    - A military-funded network doesn't care about redundancy?
  - Actual: Since when is 10^7 small, and the kludge doesn't scale.
    - It isn't 10^7, with Smart Grid it is more like 10^{10}.

The Internet is Facing Severe Problems: II

- Mobility is cumbersome and doesn’t scale
  - Excuse: What do you mean? It works. . . . . Sort of.
  - Actual: With only physical addresses, hard to do “re-locatable” addressing
- Congestion Control keeps Utilization around 30%
  - Excuse: There is great congestion control in TCP . . . Sort of. Bandwidth is Cheap don’t worry about it.
  - Actual: Any control theory book says put feedback as close to the resource as possible. That puts it as far away as possible.
- Quality of Service is difficult to do.
  - Excuse: Net neutrality requires that all traffic be treated equally
  - Actual: Net neutrality is political cover for their inability to do it.
    - Notice: Running out IP addresses was not listed
      - Not a problem. A global address space is not required

And if that Weren’t Bad Enough

Much of What is Believed
about the Internet is Myth

Myths of the Internet: I

- The Internet is an Engine of Innovation.
  - The Internet has in a real sense been stagnant since the late-70s
  - Living on Moore’s Law and Band-aids
    - Lots of Innovation on *top* of the Internet, but even that has begun to wane.
- The Internet has decentralized administration. No one owns the Internet.
  - Actually, Same as the global PSTN, just no sexy name.
- The IETF is a Grass-Roots Democracy.
  - Actually, It most closely Resembles the Communist Party.
    - IETF meeting is Party Congress; IESG is Politburo
- The Internet is based on the ARPANET
  - Actually, It is based on the French network CYCLADES

Myths of the Internet: II

- The Internet is not an internet, but a catenet.
  - Ceased to be an Internet on January 1, 1983.
- The Internet is a dumb network.
  - Actually, it keeps maximal state in the network, not minimal.
- The Internet has decentralized routing.
  - Actually, most ISPs routes are statically allocated.
- IP is the Internet Protocol.
  - That is what the letters stand for but it is really a subnet protocol.
- IP addresses name the host.
  - No, they name the interface to the host.
- TCP isn't perfect, but it is good enough.
  - Every design decision is not just wrong but makes something else worse. One thing it got right was destroyed creating IP.

The Reality is Quite Different

- If the Internet were an Operating System, it would have more in common with DOS, than UNIX.
- Imagine trying to use a modern computer with only physical memory addresses.

- That is the Internet today.

Need to Do Something About It

- A Decade ago, DARPA funded NEWARCH
  - All the “top brains” to come up with a new Internet architecture
  - Two years later, they came up dry. Nada. Nothing.
- At the time of NEWARCH, NRC told them they don’t know how to do research and don’t seem to have learned anything since.

> “A reviewer of an early draft of this report observed that this proposed framework – measure, develop theory, prototype new ideas – looks a lot like Research 101. . . . From the perspective of the outsiders, the insiders had not shown that they had managed to exercise the usual elements of a successful research program, so a back-to-basics message was fitting.”
> **Looking over the Fence at Networking,** Committee on Research Horizons in Networking, National Research Council, 2001.

- The field is no longer a science, but a craft.
  - They have been asking “What to build?”
  - Not asking, “What don’t we understand?”

But They Kept Trying

- When DARPA was unwilling to throw good money after bad, they went to NSF.
- FIND and GENI, massive projects to find the Answer.
  - At it for years, spending millions, and have nothing to show.
- Similar projects in Europe and Asia.
  - Same result. Isn't groupthink wonderful?
- No closer to an answer now than a decade ago.
  - "Rough consensus and running code" doesn't exactly foster the kind of thought required to uncover theory.

Guiding "Principles" Aren't Much Help

- Soft State/Hard State
  - All properly designed protocols are soft state; only some database operations are hard state. A specious distinction
- Loc/Id Split
  - Attempt to avoid the obvious solution. Mostly post IPng trauma.
  - Continues to route to the wrong place
- Fate Sharing
  - Mostly used as a rug . . . to sweep under.
- End-to-End Principle
  - At best a lemma. More a statement of desire, by focusing attention on the dichotomy of the middle vs the edge, it misses the point.
  - Hence becomes an impediment to finding a way forward.

When They Came Up Dry

- Both NEWARCH and FIND recommended involving fields outside computer science.
- The idea is that they need to consider more widely to understand how the network of the future will be used.
  - So they know what to *build*
  - Not to uncover what it is about networks they don't understand
- Concepts in other fields will shed light on network.
  - Can't possibly be that they don't understand what questions to ask

- They have been looking in the wrong direction
  - The answers aren't outside Computer Science, they are inside CS

The Myths and “Principles” are *the* Major Barrier to FINDing an Answer

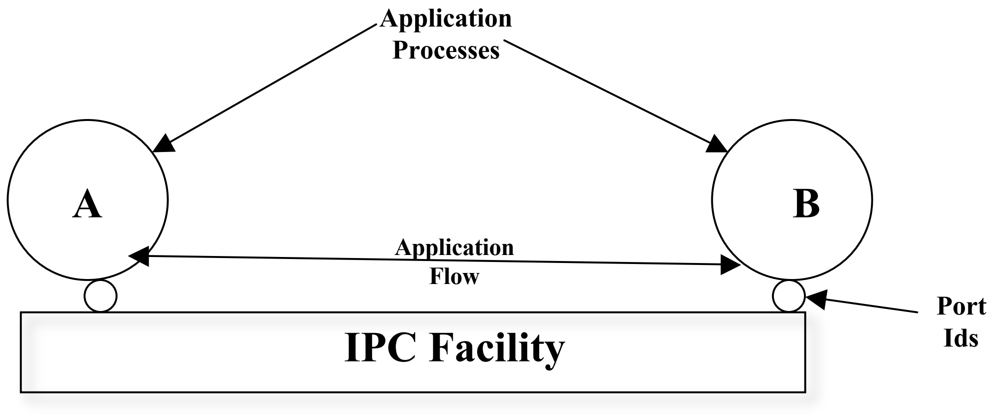
- The Answer has been clear since the mid-90s.
- And it is very simple:
- Networking is IPC and only IPC

Introduction

- Start simple and incrementally introduce new conditions.
  - Taking Imre Lakatos' Proofs and Refutations as a model.
- We are going to cover ground we all know.
  - Or at least we think we do. (I thought so)
- As we do, think about what is required when
  - May be interesting: the order is not what you might expect.
  - But the implications are even more interesting.

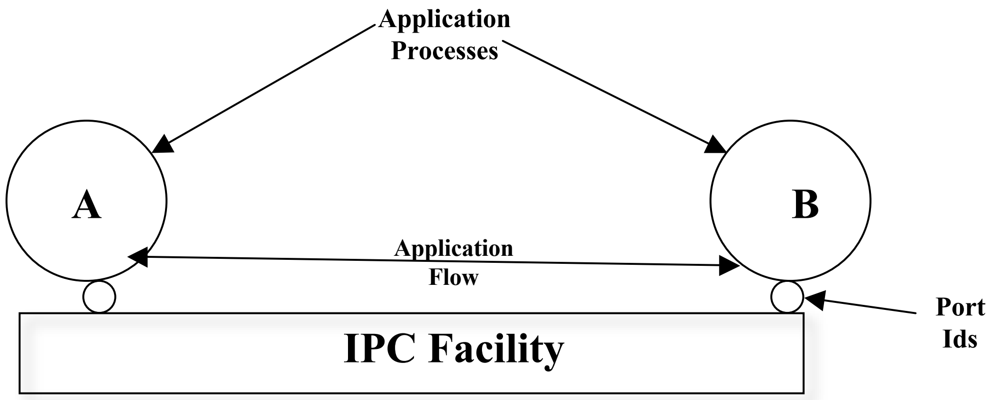
1: Start with the Basics

Two applications communicating in the same system.



Communication within
a Single Processing System

How Does It Work?



1) **A** invokes IPC to create a channel to **B**; **a = Allocate (B, QoS);**

2) IPC determines whether it has the resources to honor the request.

3) If so, IPC allocates port-id **a** and uses "search rules" to find **B** and determine whether A has access to B**.**

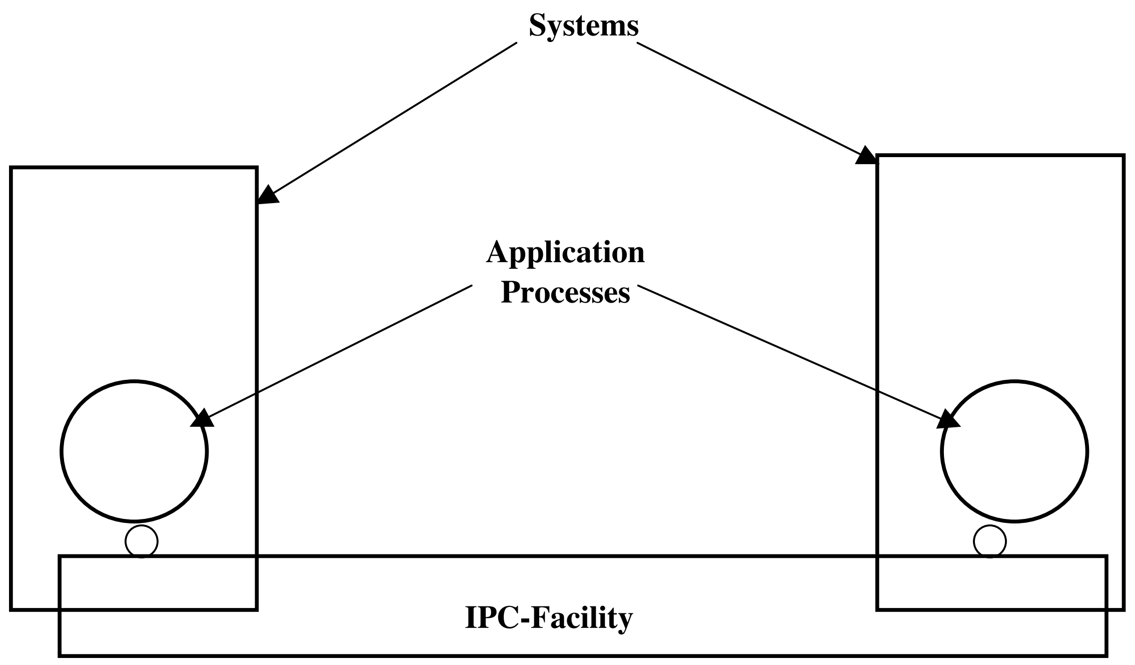
4) IPC may cause an instance of **B** to be created. **B** is notified of the IPC request from **A** and given a port-id, **b**.

5) If **B** responds positively, and IPC notifies **A** (the API could be blocking in which case the assignment of the port-id, **a** would be done now).

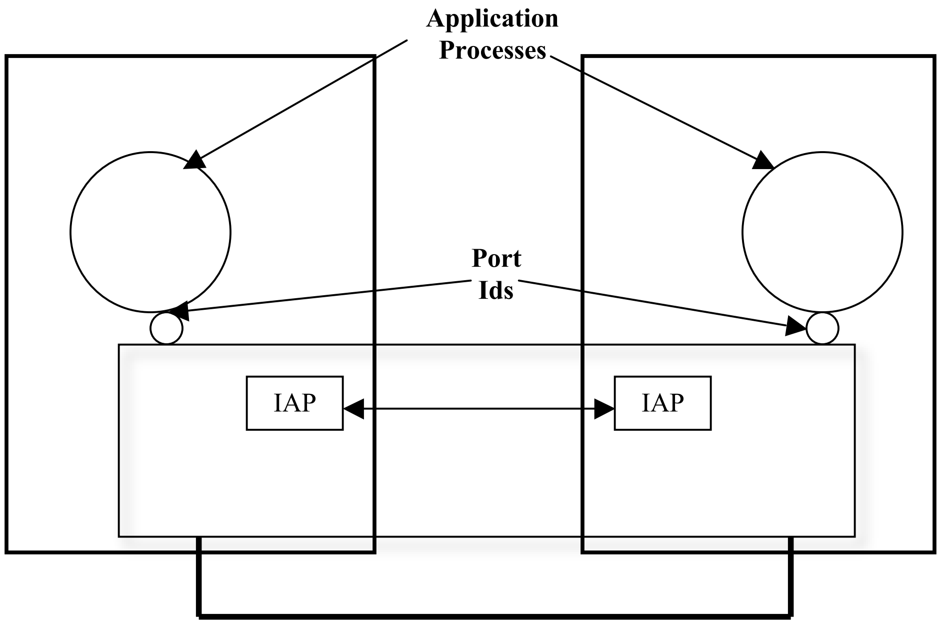
6) Thru n) Then using system calls **A** may send PDUs to **B** by calling **Write(a, buf),** which B receives by invoking **Read(b, read_buffer)**

7) When they are done one or both invoke **Deallocate** with the appropriate parameters

2: Two Application Communicating in Distinct Systems



How Does It Work Now?



1) A invokes IPC to create a channel to **B**; a = **Allocate (B, QoS);**

2) IPC determines whether it has the resources to honor the request.

3) If so, IPC uses “search rules” to find **B.** and determine if A has access to B.

- Management of name space is no longer under the control of a single system.
  - Each system no longer knows all available applications.
- Local Access Control can no longer be relied on to provide adequate authorization and authentication.

• Need a protocol to carry application names and access control information.

- Lets call it the IPC Access Protocol

What Does “IAP” Look Like?

- Simple Request/Response Protocol
  - IAP-Req(Dest-Appl-name, Src-Appl-name, QoS params, Src-Capability)
  - IAP-Resp(Dest-Appl-name, Src-Appl-name, QoS params, result)

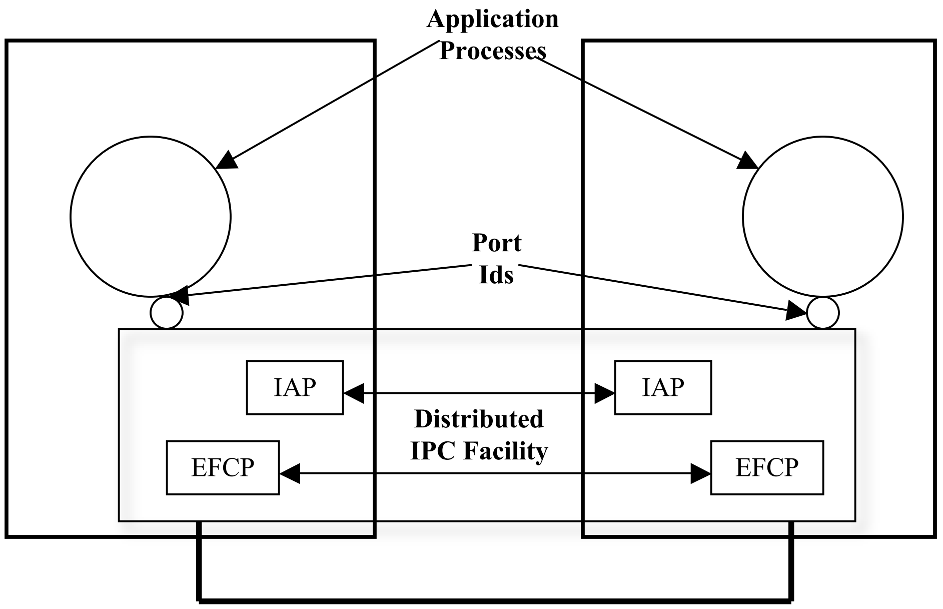
Op	Dest Appl Name	Src Appl Name	QoS & Policies	Capability

- How do we know when to use it?
  - If the application isn’t here, it must be there!
- But we have a problem. How do we get it there?

Getting from A to B

- We knew this was going to be a problem sooner or later.
- We need to be able to send information from A to B.
- And we know:
  - Bad things can happen to messages in transit. Protection against lost or corrupted messages
    - This can be expensive
  - Receiver must be able to tell sender, it is going too fast. We need Flow Control, and
  - We have lost our means of synchronization:
    - No common test and set means shared memory can no longer be used
    - Must create shared state between two systems. An explicit synchronization mechanism is required.
- We need some kind of Protocol for Error and Flow Control.

How Does It Work Now?



1) **A** invokes IPC to create a channel to **B**; a = **Allocate (B, QoS);**

2) IPC determines whether it has the resources to honor the request.

3) Send IAP Request to access B**,** creating an EFCP connection and determines if A has access to B.

4) IPC may cause **B** to be instantiated. **B** is notified of the IPC request from **A** and given a port-id, **b**.

5) If **B** responds positively, and IPC notifies **A** with a different port-id, **a**.

6) Thru n) Then using system calls **A** may send PDUs to **B** by calling **Write(a, buf),** which B receives by invoking **Read(b, read_buffer)** over the EFCP connection

7) When they are done one or both invoke Deallocate with the appropriate parameters

Just Like Before . . . .More or Less

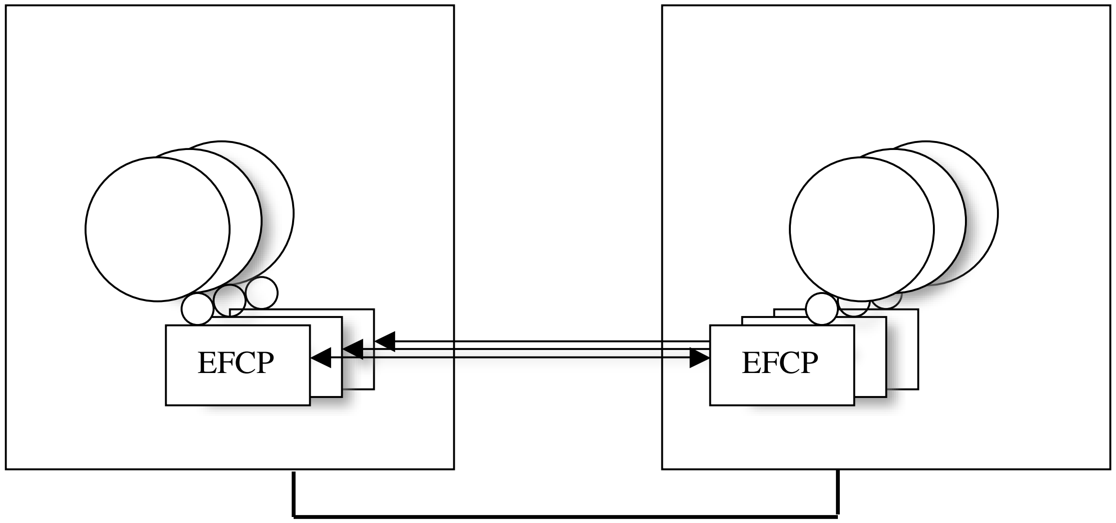
Two Applications in Two Systems Required Three New Concepts

- An Application Name Space that spans both systems. (not really new)
  - Should be location-independent in general so that applications can move.
- A Protocol to carry Application Names and access control info
  - Applications need to know with whom they are talking
  - IPC must know what Application is being requested to be able to find it.
    - For now, if the requested Application isn't local, it must in the other system.
- A Protocol that provides the IPC Mechanism and does Error and Flow Control.
  - To maintain shared state about the communication, i.e. synchronization
  - To detect errors and ensure order
  - To provide flow control
- Resource allocation can be handled for now by either end refusing service.

3: Simultaneous Communication Between Two Systems

i.e. multiple applications at the same time

- To support this we have multiple instances of the EFCP.



Will have to add the ability in EFCP to distinguish one flow from another.

Typically use the port-ids of the source and destination.

Connection-id

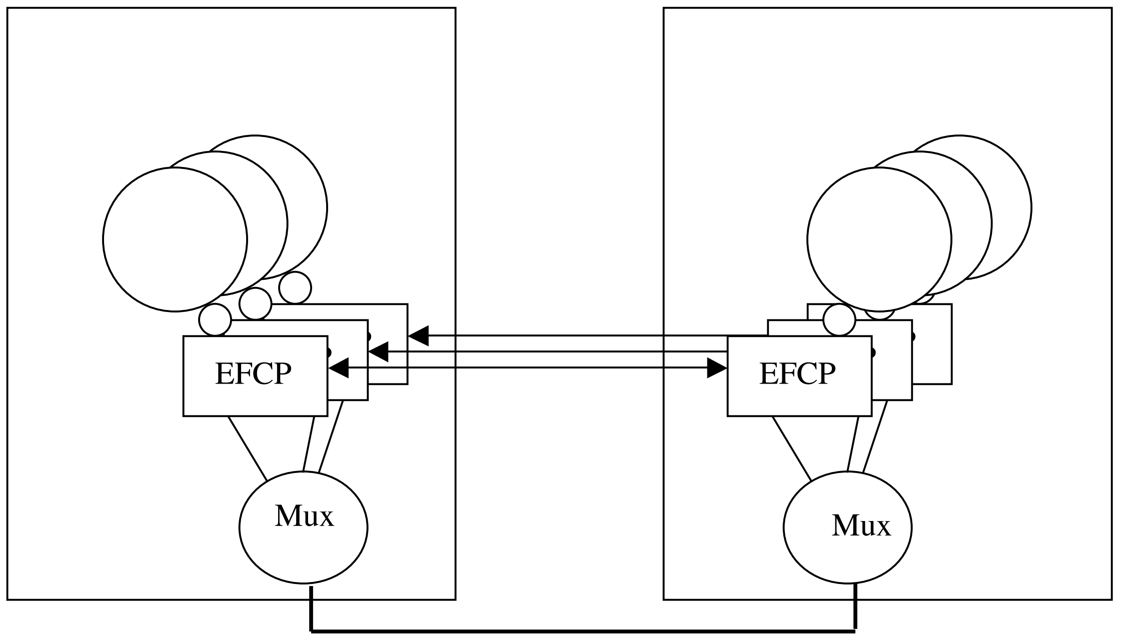
Dest-port	Src-port	Op	Seq #	CRC	Data

Also include the port-ids in the information sent in IAP to be used in EFCP synchronization (establishment).

Simultaneous Communication Between Two Systems

i.e. multiple applications at the same time

- In addition to multiple instances of the EFCP.

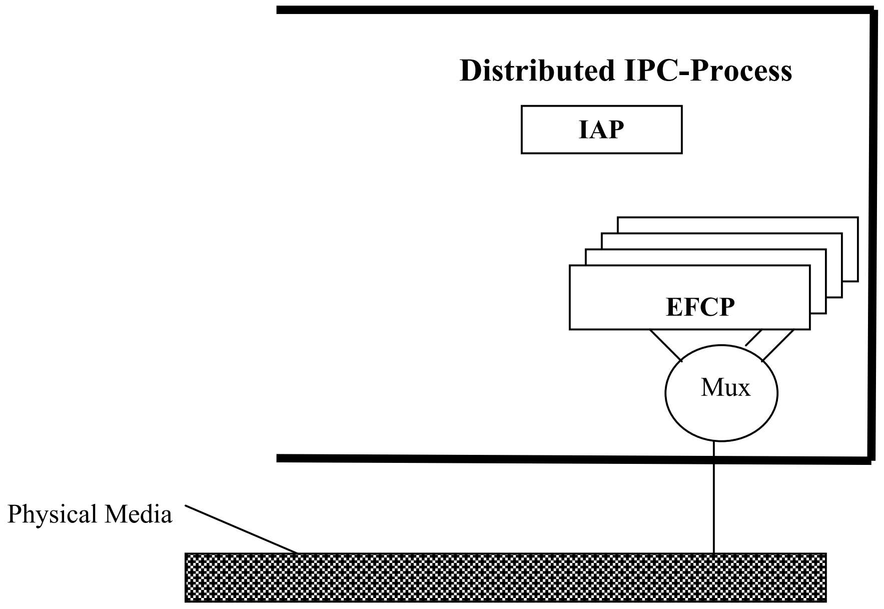


Will also need an application to manage multiple users of a single resource.

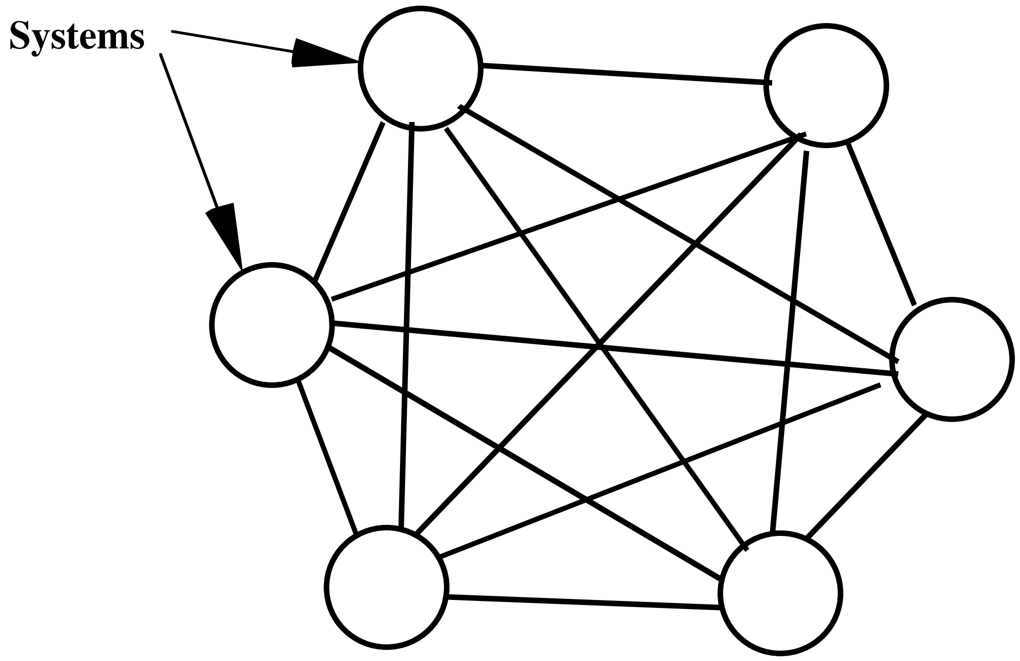
Multiple Instances of IPC

- New Concept: a multiplexing application to manage the single resource, the physical media.
- The multiplexing application will need to be fast, its functionality should be minimized, i.e. just the scheduling of messages to send.
  - To provide QoS, we use the EFCP and scheduling by the Mux.
- To do resource allocation, we will just let the other side refuse if it can’t satisfy the request.
- Application naming gets a bit more complicated than just multiple application-names.
  - Must allow multiple instances of the same process

IPC to Support Simultaneous Communication between Two Systems



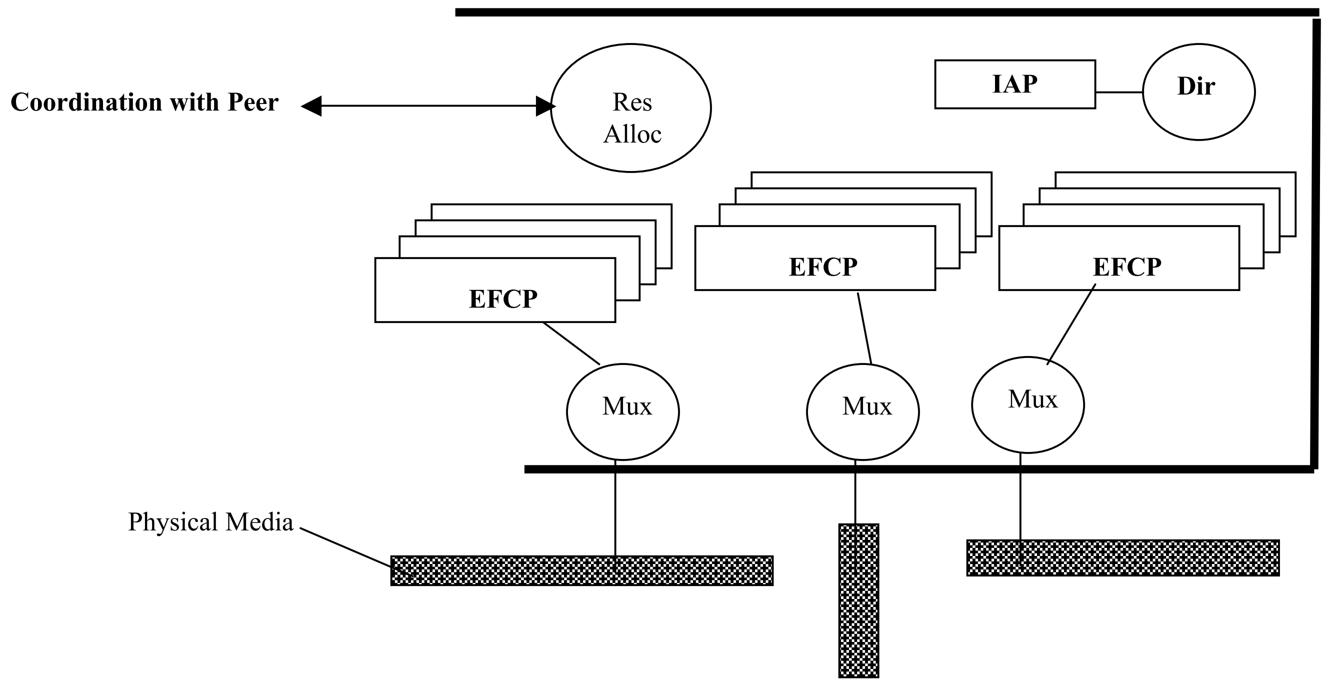
4: Communication with N Systems



Replication Entails More Management

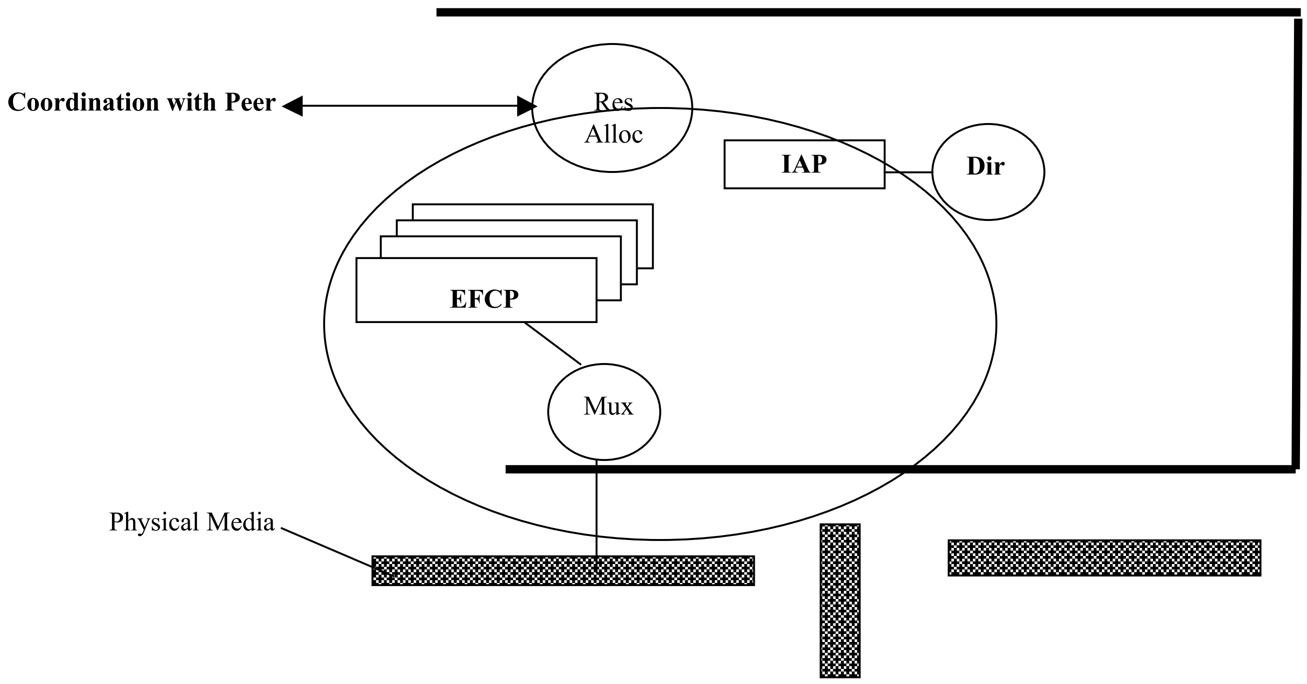
- Separate Multiplexing Application for each physical media interface.
- IPC can find the destination by choosing the appropriate interface.
- If enough applications, create a Directory to remember what is where, i.e. what application names are at the other end of which interfaces.
- Same local names can be used to keep track of which EFCP-instances (port-ids) are bound to which Multiplexing Application.
- With many destinations, may need to coordinate resource allocation information within our distributed IPC Facility.

With Multiple Interfaces It Gets Messy



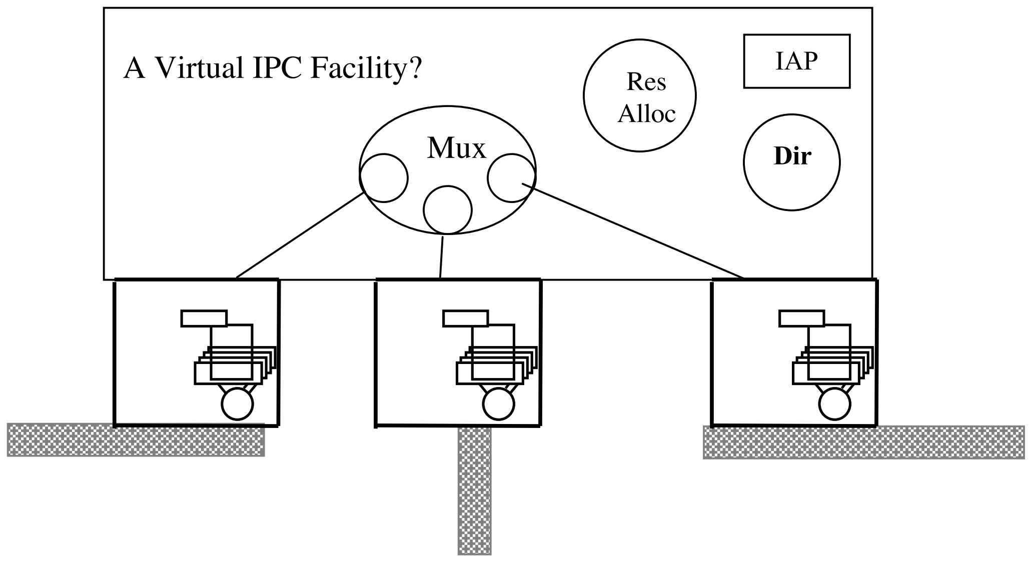
- So a task to manage the use of the interfaces and mask any differences.
  - A little organizing will help.

There is Some Common Structure



- We can organize interface IPC into modules of similar elements.
- Each one constitutes a Distributed IPC Facility of its own.
  - As required, consists of IAP, EFCP, Multiplexing Application, Directory, Per-Interface Resource Allocation
- Then we just need an application to manage their use and moderate user requests.

A Little Re-organizing

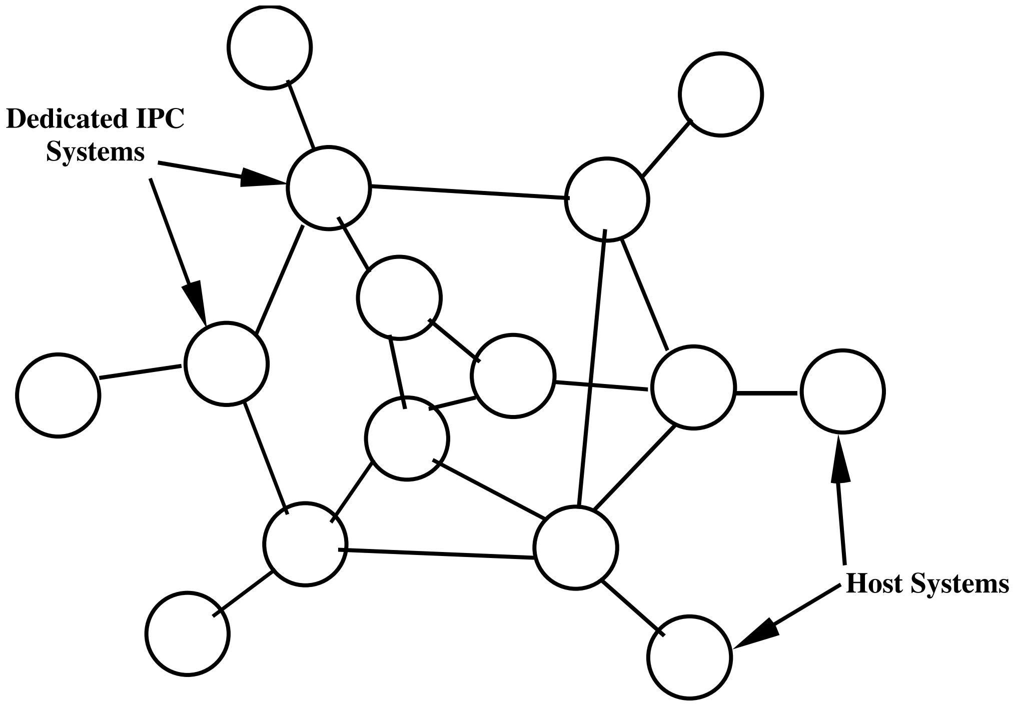


So we have a Distributed IPC Facility for each Interface and an application over all of them to manage their use and to give the user a common interface, a Virtual IPC Facility?

BUT

- This fully connected graph approach isn't going to scale very well and is going to get very expensive.
  - And not everyone needs to talk to everyone else all the time.
- Need to do something better.

5: Communicating with N Systems (On the Cheap)



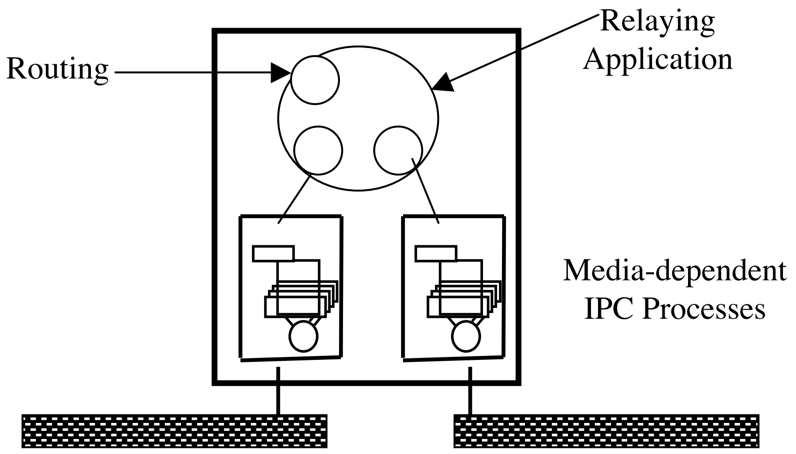
By dedicating systems to IPC, reduce the number of lines required and even out usage by recognizing that not everyone talks to everyone else the same amount.

Communications on the Cheap

- We will need systems dedicated relaying and multiplexing.
- That requires some new elements:
  - Globally accepted names for source and destination muxing apps.
  - And also for the relays. Relays require names for routing. Have to know where you are to determine where to go next.
  - Need routing applications too, which will need to exchange information on connectivity.
- Will need a header on all PDUs to carry the names for relaying and multiplexing.
  - Interface IPC Facilities will need one too if they are multiple access.

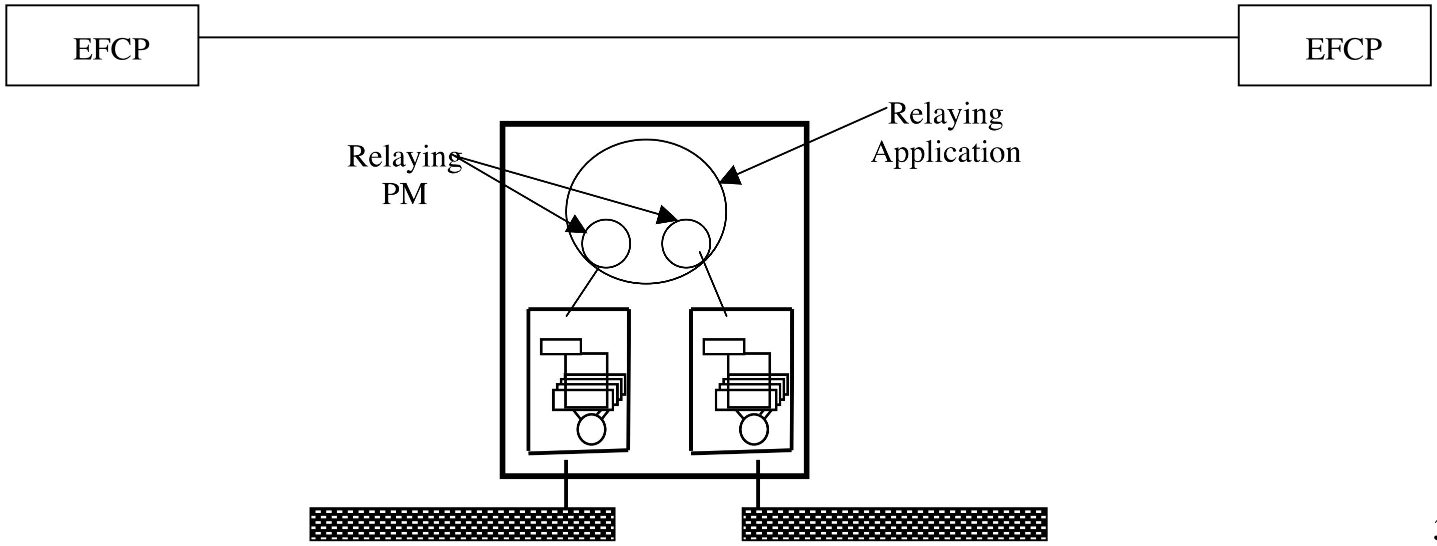
Dest Addr	Src Addr

Common Relaying and Multiplexing Application Header



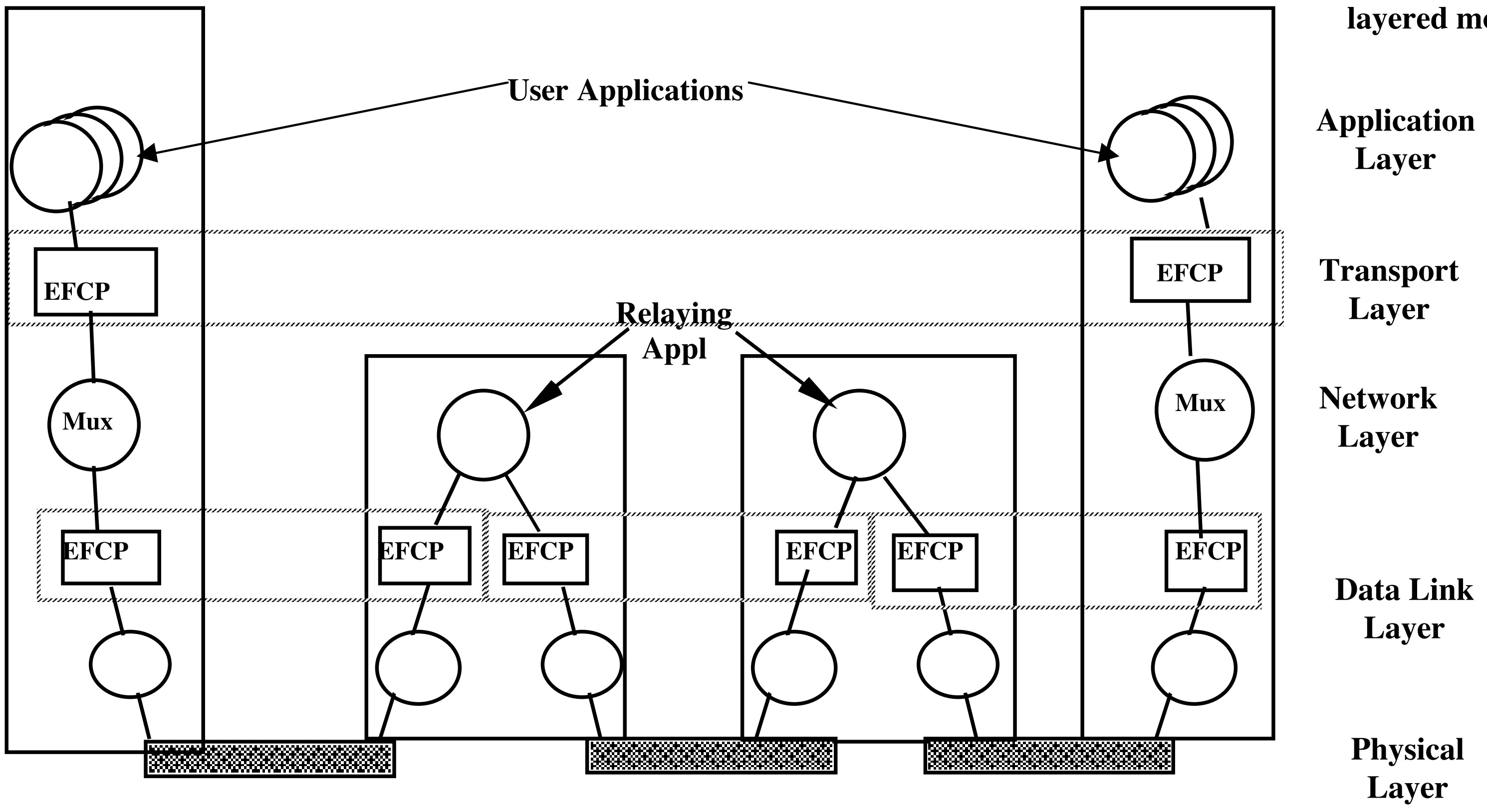
Communications on the Cheap

- But relaying systems create problems too.
  - Can’t avoid momentary congestion from time-to-time.
  - Annoying bit errors can occur in their memories.
- Will have to have an EFCP operating over the relays to ensure required QoS reliability parameters.
  - Our virtual IPC Facility isn’t very virtual.



The Big Picture

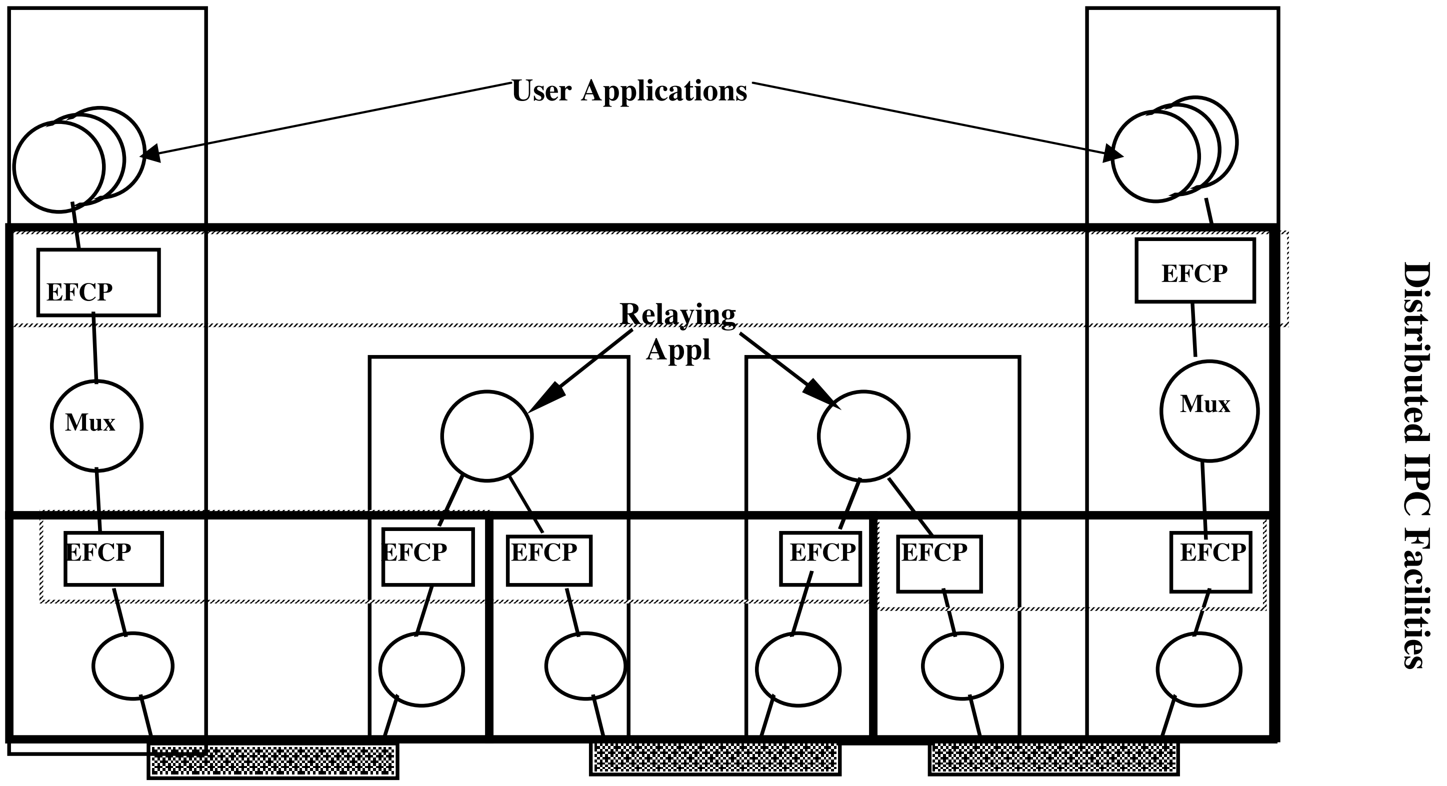
But this is half way between a bead-on-a-string model and a layered model



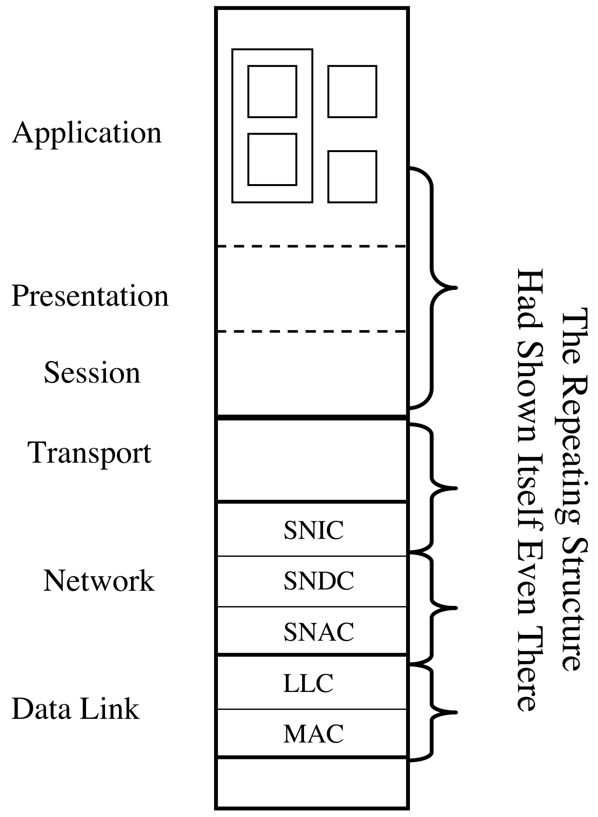
This should look familiar.

The IPC Model

(A Purely CS View)



OSI Should Have Seen the Pattern



Summary

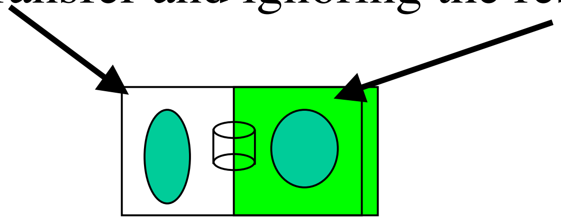
- "Networking is InterProcess Communication"
  - . . . . and only IPC.
    - The quote is Bob Metcalfe, 1972. (The rest is mine.)
- A layer is a distributed application that provides IPC consisting of a collection of SDU protection, muxing, EFCP, and their associated routing and resource management tasks.
- And this Distributed IPC Facility repeats.
- This constitutes a basis for a general theory of networking.
- Starting with a truly clean slate yields some interesting results, but before we can do that a couple of other results:

The Connection Connectionless War

- The technical side of what was really an economic war.
  - The Layered Model invalidated both the PTT and IBM business models.
  - Connectionless removed the security blanket of determinism.
  - The war created a bunker mentality that made understanding hard.
    - All or nothing.
- For years, we saw it as the amount of shared state.
  - Connections had lots of shared state; connectionless very little.
- A function of the layer, not a service. Not related to reliability
  - Not visible over the layer boundary.
- Later it became clear that
  - As traffic becomes more deterministic, connections are preferred
    - Down in the layers and in toward the backbone
  - As traffic becomes more stochastic, connectionless is preferred
    - As one moves up and toward the periphery

Resolving the CO/CL Problem

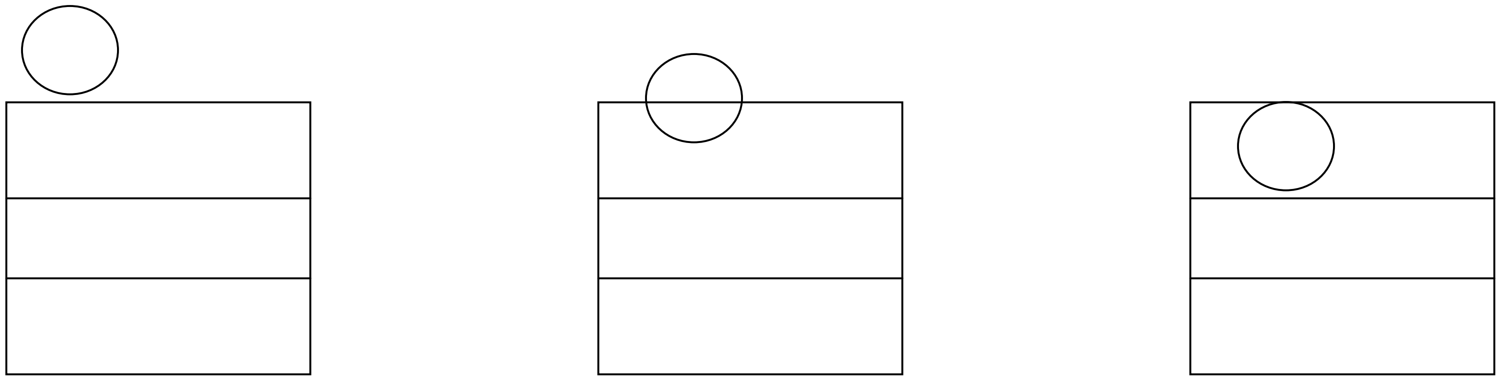
- Lets look at this very carefully
- What makes connection-oriented so brittle to failure?
  - When a failure occurs, no one knows what to do.
  - Have to go back to the edge to find out how to recover.
- What makes connectionless so resilient to failure?
  - Everyone knows how to route everything!
- 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:



Applications and Communication: I

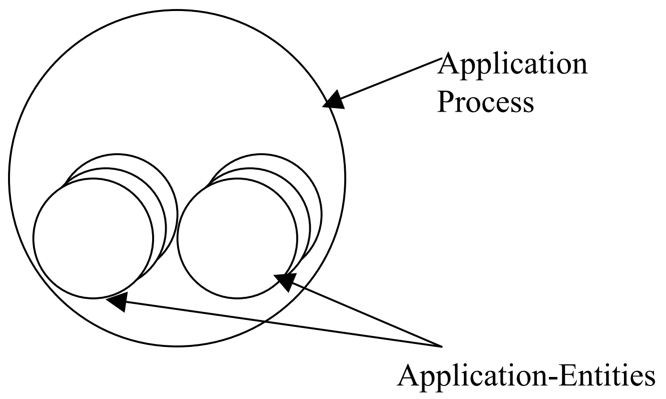
Is the Application in or out of the IPC environment?

- The early ARPANet/Internet didn't worry too much about it. They didn't need to. Only one FTP per system, only one remote login per system, etc.
- By 1985, OSI had tackled the problem, partly due to turf. Was the Application process inside or outside OSI?



- It wasn't until the web came along that we had an example that in general an application protocol might be part of many applications and an application might have many application protocols.

Applications and Communication: II



- The Application-Entity (AE) is that part of the application concerned with communication, i.e. shared state with its peer.
- The rest of the Application Process is concerned with the reason for the application in the first place.
- An Application Process may have multiple AEs, they assumed, for different application protocols.

BUT

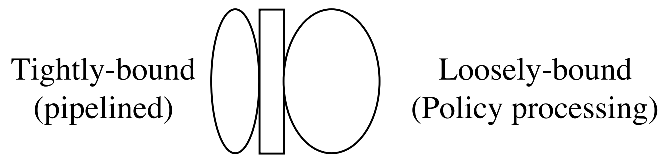
There is only one application protocol

- Huh!? Think about it. What can you do remotely?
  - Read/Write – Create/Delete – Start/Stop
  - On various objects. Everything is just an object outside the protocol.
    - Application protocols modify state outside the protocol.
- In that case, why do we need to name this application-entity stuff?
  - A particular collection of objects are required for an activity.
  - May be shared with others, but has its own access control.
  - One protocol, potentially shared objects, different state machines
  - Hence, all application protocols are stateless, the state is in the application.

Delta-t 1980

- Richard Watson develops delta-t, a unique approach.
  - Assumes all connections exist all the time.
  - TCBs are simply caches of state on ones with recent activity
- Watson proves that the necessary and sufficient conditions for synchronization are met if 3 timers are bounded:
    - Maximum Packet Lifetime
    - Maximum number of Retries
    - Maximum time before Ack
  - That no explicit state synchronization, i.e. hard state, is necessary.
    - Syns, Fins are unnecessary
- IOW, any properly designed data transfer protocol is soft-state.
- 1981 paper, Watson shows that TCP has all three timers and more.
- And PNA figures out that . . . .

The Structure of Protocols



- If we separate mechanism and policy, we find there are
- Two kinds of mechanisms:
  - Tightly-Bound: Those that must be associated with the Transfer PDU
    - policy is imposed by the sender.
  - Loosely-Bound: Those that don’t have to be.
    - policy is imposed by the receiver.
  - Furthermore, the two are only loosely coupled through a state vector.
    - Implies a very different structure for protocols and their implementations
      - Right, we split TCP in the wrong direction

Implications: Protocols I

- Data Transfer Protocols modify state *internal* to the Protocol. Application Protocols modify state *external* to the protocol.
- There are only two protocols (full stop):
  - A data transfer protocol, based on delta-t
  - An Application protocol that can perform 6 operations on objects:
  - There is no distinct protocol like IP.
    - Was just a common header fragment anyway.
- A Layer provides IPC to either another layer or to a Distributed Application using a programming model. The application protocol is the "assembly language" for distributed computing.

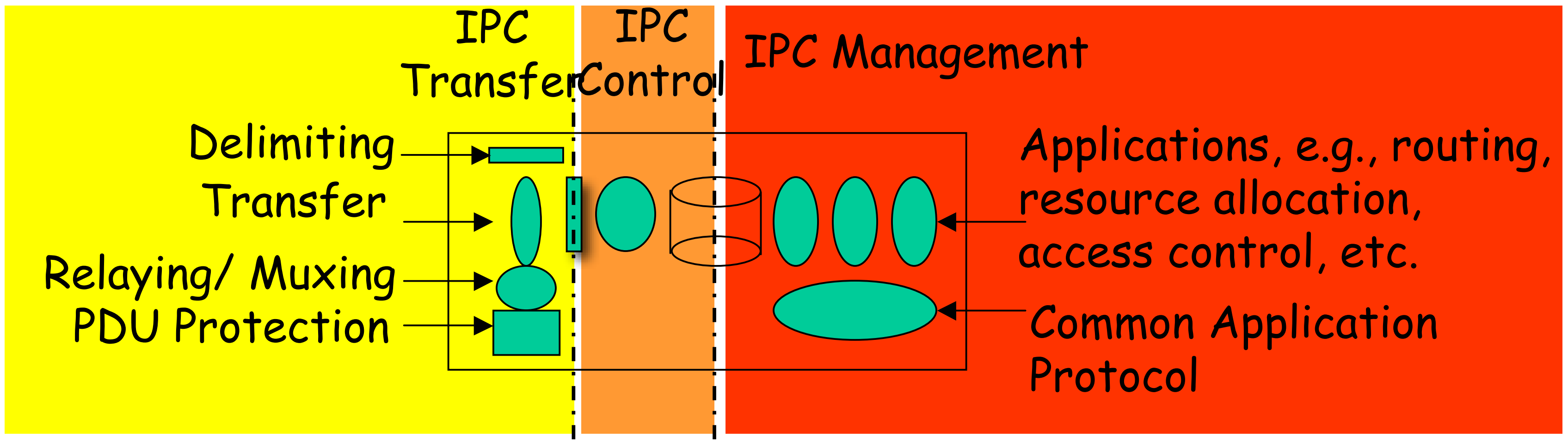
Implications: Protocols II

- “Hard state” only occurs for some uses of application protocols
  - Storing in a database *may* be hard-state. Everything else is soft-state.
  - Hence the “hard-state/soft-state” distinction at best states the obvious.
- Separating mechanism and policy in a delta-t like protocol will yield the entire range from UDP-like to TCP-like.
- Watson implies decoupling “port allocation” from synchronization.
  - Greatly simplifying security mechanisms, enabling multi-flow allocations of IPC, etc.
- And One Other Thing:

Fundamental Result

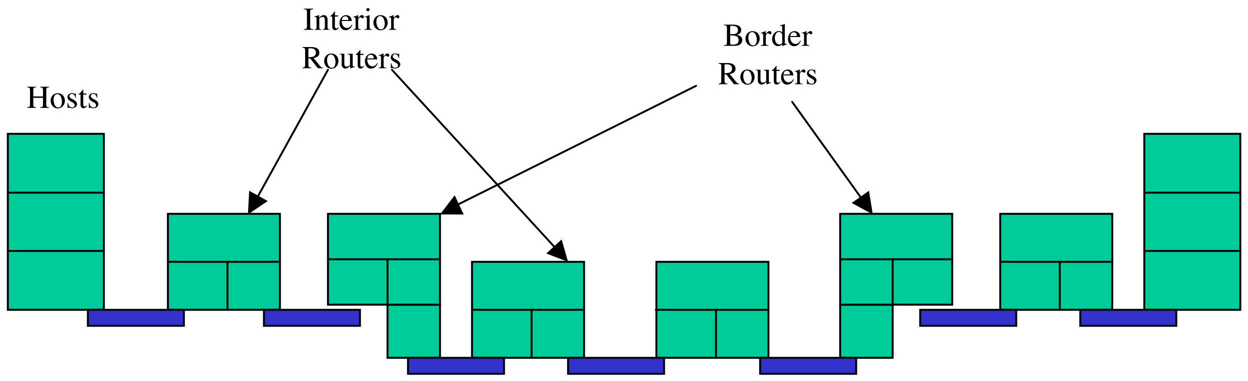
- Watson’s result also defines what is networking or IPC:
  - It is IPC if and only if Maximum Packet Lifetime can be bounded.
  - If MPL can’t be bounded, it is remote storage.

What a Layer Looks Like


IPC Transfer
IPC Control
IPC Management
Delimiting
Transfer
Relaying/ Muxing
PDU Protection
Applications, e.g., routing, resource allocation, access control, etc.
Common Application Protocol

- Processing at 3 timescales, decoupled by either a State Vector or a Resource Information Base
  - IPC Transfer actually moves the data (≈ IP + UDP)
  - IPC Control (optional) for retransmission (ack) and flow control, etc.
  - IPC Layer Management for routing, resource allocation, locating applications, access control, monitoring lower layer, etc.

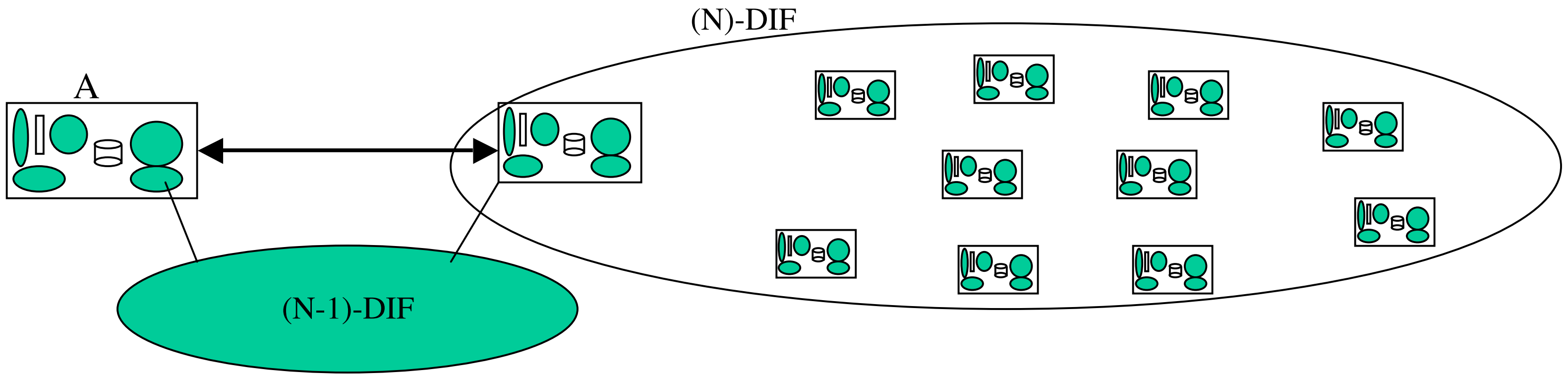
Only Three Kinds of Systems



- Middleboxes? We don't need no stinking middleboxes!
- NATs: either no where or everywhere,
    - NATs only break broken architectures
- The *Architecture* may have more layers, but no *box* need have more than the usual complement.
    – Hosts may have more layers, depending on what they do.

How Does It Work?

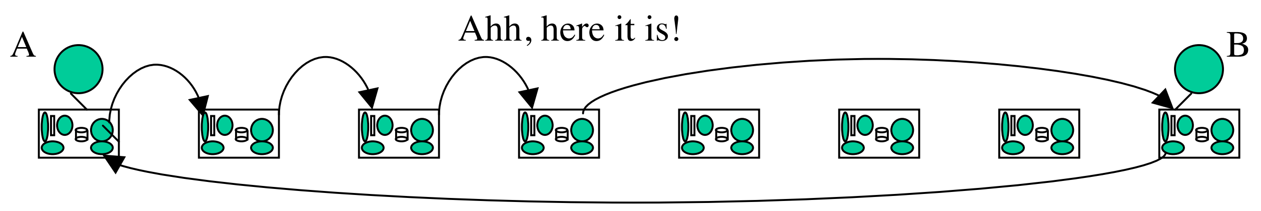
Joining a Layer



- Nothing more than Applications establishing communication (for management)
  - Authenticating that A is a valid member of the (N)-DIF
  - Initializing it with the current information on the DIF
  - Assigning it a synonym to facilitate finding IPC Processes in the DIF, i.e. an address

How Does It Work?

Establishing Communication



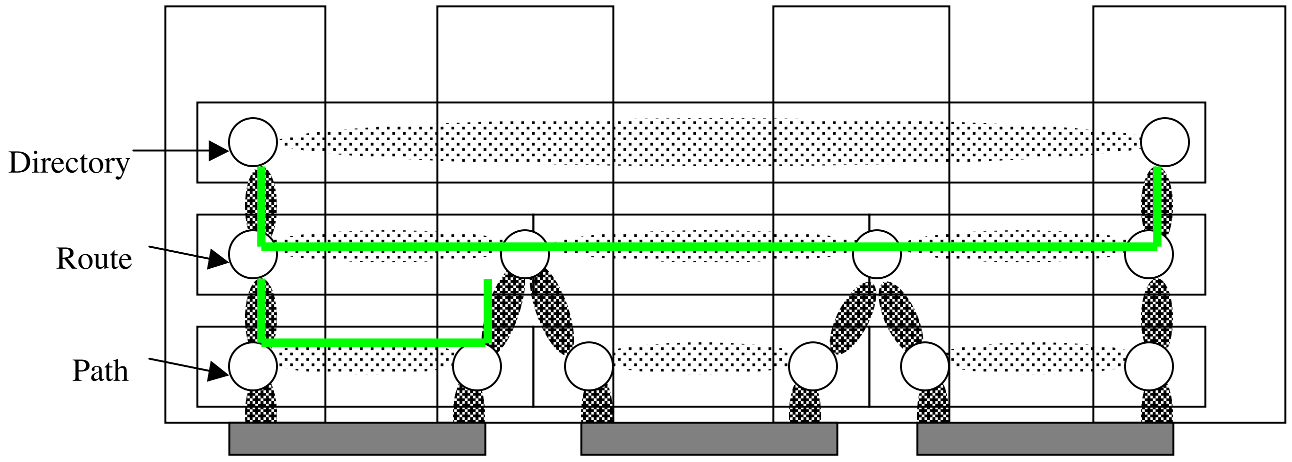
- Simple: do what IPC tells us to do.
  - A asks IPC to allocate comm resources to B
  - Determine that B is not local to A use search rules to find B
  - Keep looking until we find an entry for it.
  - Then go see if it is really there and whether we have access.
  - Then tell A the result.
- This has multiple advantages.
  - We know it is really there.
  - We can enforce access control
  - We can return B's policy and port-id choices
  - If B's has moved, we find out and keep searching

Implications: Naming and Addressing I

- IPC Processes are application processes that are members of a DIF.
- Hence, they have application-process-names.
    - These are probably quite long and unwieldy to use for forwarding among the members of the DIF.
- Assign them synonyms from a name space with less scope, i.e. they are shorter.
    - Call it a "Forwarding-Identifier" or F-id.
- Application Process Names (and their synonyms or sets of them) are the only non-local names required in an architecture.
    - All other names are local in scope.
        - Port-ids
        - Connection-endpoint-ids

Generalizing Saltzer to Networks

- 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.



Implications: Naming and Addressing II

- If the DIF has a sufficiently large number of members (or maybe even if it doesn't) we would like to make using F-ids even more useful.
  - Structure them to reflect which elements are near each other for some idea of "near."
  - This is an "address" in the same sense as its normal usage. It expresses “where” without indicating “how to get there” (see Shoch's paper).
    - It is considering this idea of "nearness" where topology comes into the picture. Topology is the study of functions that preserve "nearness."
- A node address is an (N)-F-id and
- A Point of Attachment address is an (N-1)-F-id.
  - However, we want to deal with (N-1)-port-ids

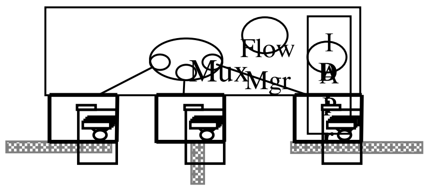
Implications: Naming and Addressing III

- A Layer routes on the addresses in its layer.
  - (Sounds obvious but IP routes on someone else’s address)
- Routing table size can be bounded by recursion.
- Deep packet Inspection is unnecessary.

- A Global Address space is Unnecessary.
- Huh!?

Choosing a Layer

- In building the IPC Model, the first time multiple DIFs are encountered (data link layers in that case), it was found useful to have a task to determine which DIF to use.

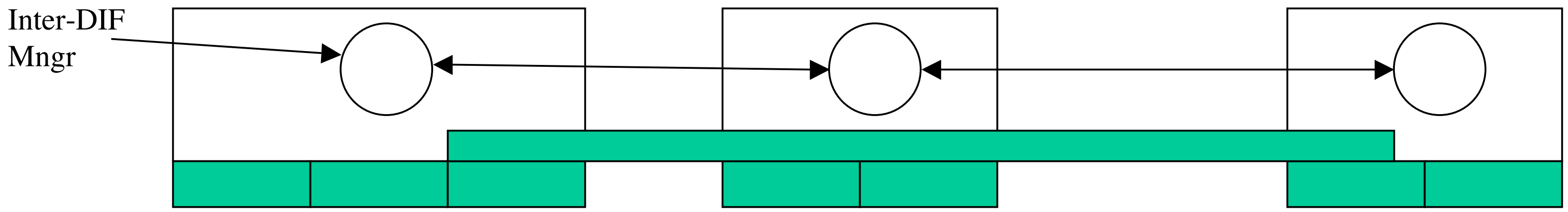


  - So user didn't have to see all of the wires
  - But the user shouldn't have to see all of the "Nets" either.

- This not only generalizes but has major implications.

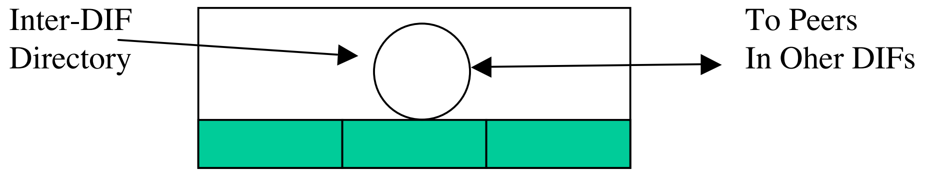
An Inter-Net Directory?

- Inter-DIF Manager determines what Net an application is on.
  - If this system is a not member, it either joins the net as before
  - or creates a new one.



- Which Implies a global address space is not necessary
  - the largest address space has to be only large enough for the largest DIF
  - Given the structure, 32 or 48 bits is probably more than enough.
- You mean?
  - Right. IPv6 was a waste of time . . .
  - Twice.

A Global Application Name Space is Useful, but a Global Address Space is Not Necessary



Actually one could still have distinct names spaces within a DIFs (synonyms) with its own directory database.

- Not all Names Need to be in the Global Directory
  - A DIF could have its own parallel application name space and directory of distributed databases. (no single DNS)
  - Alternate name spaces are not only possible but useful. (and scary)

Implications: Naming and Addressing IV

- Multihoming at no cost as a consequence of the structure
- Old result (1982): Don't embed an (N-1)-address in an (N)-address.
    - Makes it a *pathname*; defeats the point of making addresses location-dependent but route-independent.
        - Amazing that people still do it.
- Mobility is dynamic multihoming with expected failures.
    - Only difference is the frequency of changes
- Multicast and anycast are degenerate cases of whatevercast:
    - A whatevercast name is the name of a set of addresses with a rule for selecting members of the set when the whatevercast name is referenced.
    - Multicast folds into unicast and vice versa.
    - A multicast or anycast *address* is an oxymoron.

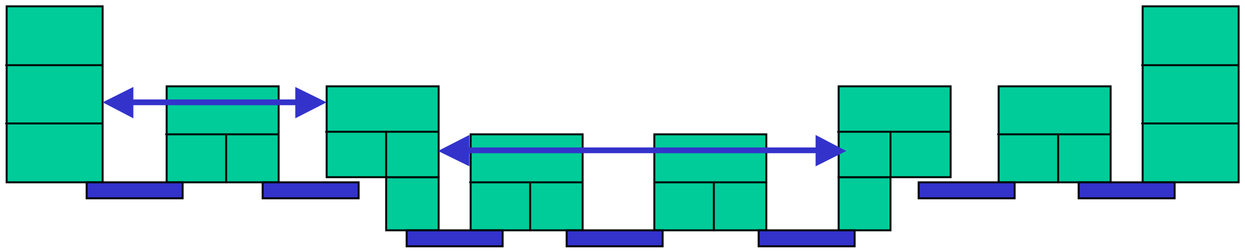
What You Don’t Need to Know

about Naming and Addressing

- What has been called a “Locator” is just an (N-1)-F-id.
- It is always been vague (and a subject of debate) what “identifier” names.
  - An Endpoint
    - Yes, but of what?
  - A Transport Address?
    - There is no such thing.
  - An Internet Address?
    - Suppose to be flat, not good for routing.
  - Some kind of Application Name?
    - Perhaps, but that isn’t our problem here.
  - RRG: Maybe endpoint should be aggregatable. Not an identifier?
    - You mean it is a locator? a node address?

How Does It Work?

“Congestion Control”

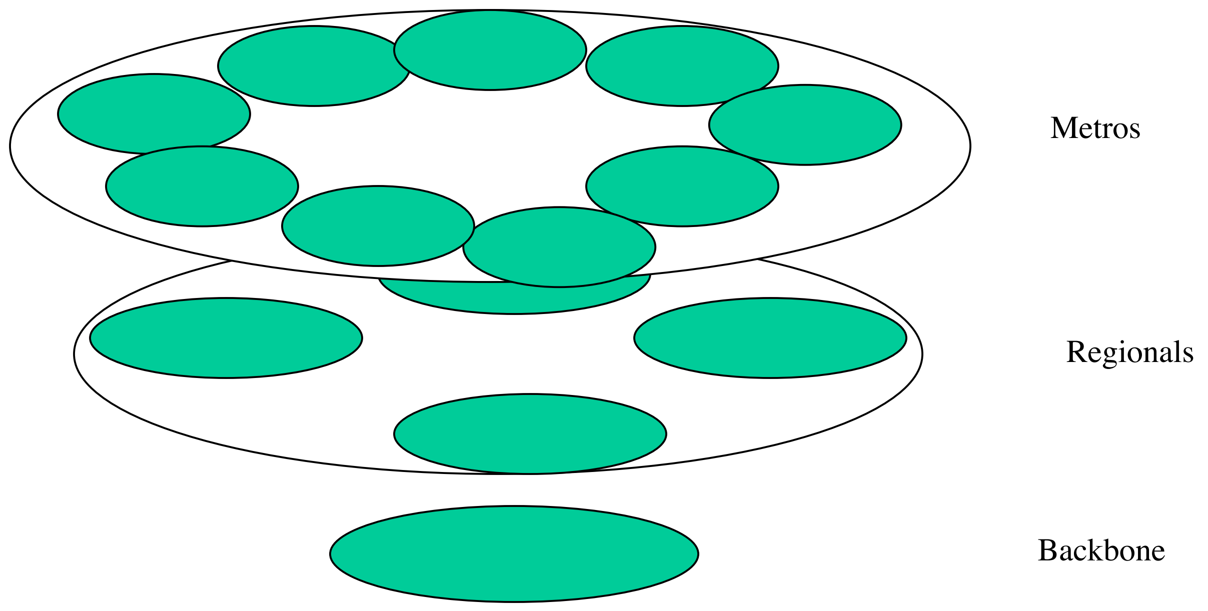


- Congestion Control in TCP was always known to be a stop-gap.
- A DIF always has the potential for the full capability of functions.
- Do flow control (without retransmissions) between intermediate points.
  - Better congestion control, really flow control
  - Allocate different resources to different e-malls.
  - Allows provider much more effective management of resources.
  - Provides means to throttle flows being used for denial of service attacks
  - All of these places? Doubtful. Research topic..

How Does It Work?

The Internet and ISPs

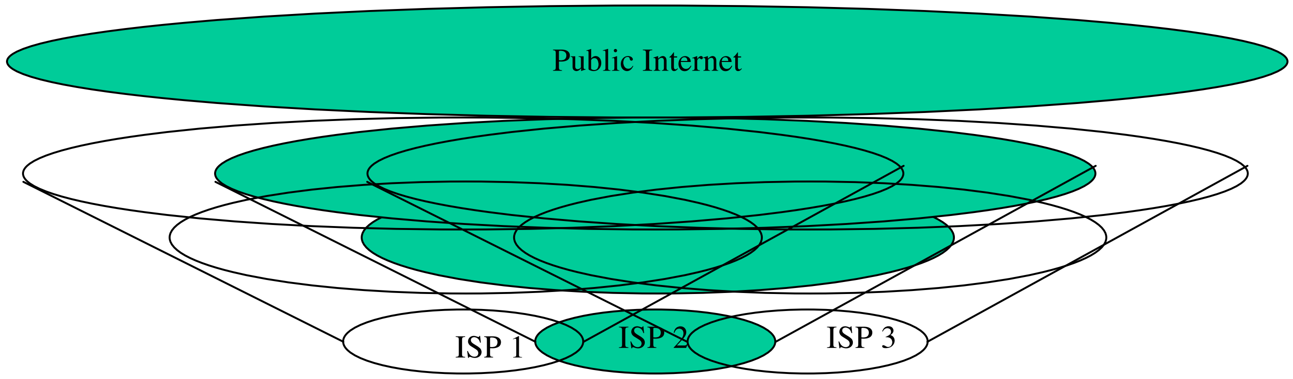
- ISPs have as many layers as they need to best manage their resources.



How Does It Work?

The Internet and ISPs

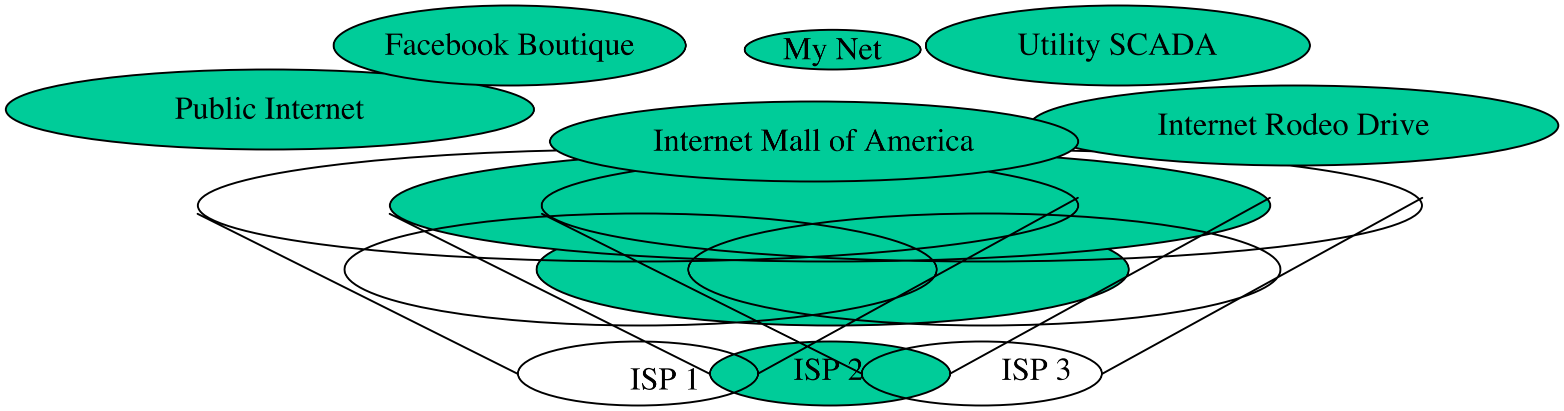
- The Internet floats on top of ISPs, a “e-mall.”
  - One in the seedy part of town, but an “e-mall”
  - Not the only emall and not one you always have to be connected to.



How Does It Work?

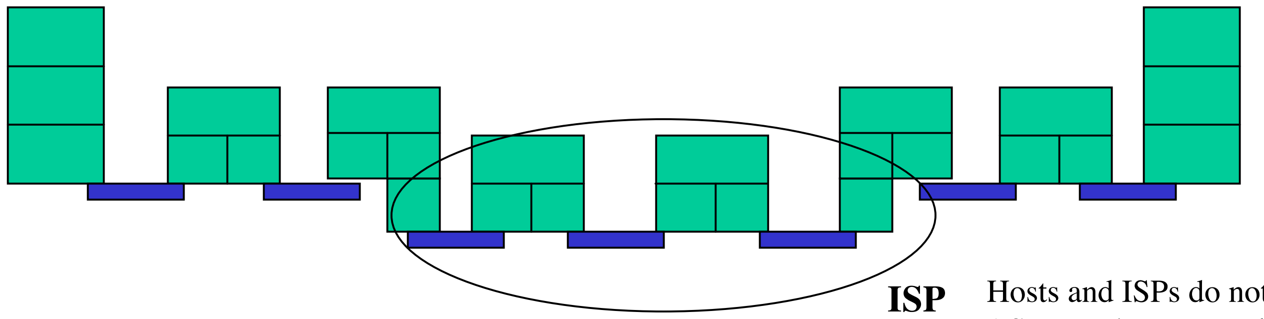
The Internet and ISPs

- But there does not need to be ONE e-mall.
  - You mean!
    - Yes, it is really an INTERnet!



How Does It Work?

Security

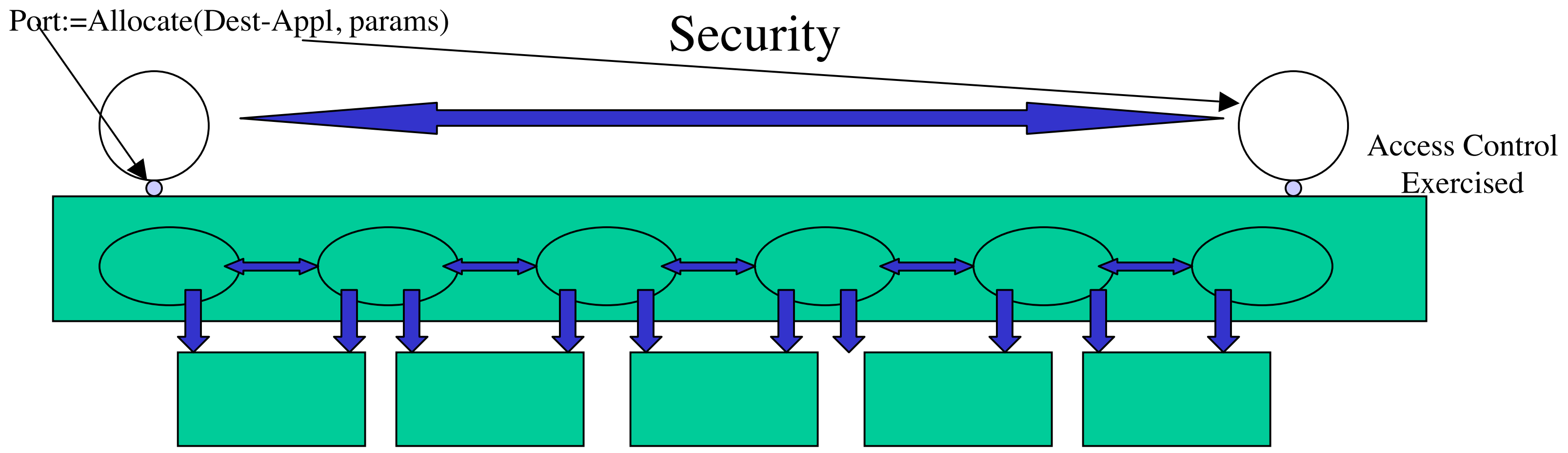


Hosts and ISPs do not share DIFS.
(ISP may have more layers

- Security by isolation, (not obscurity)
- Hosts can not address any element of the ISP.
- No user hacker can compromise ISP assets.
    - Unless ISP is physically compromised.

How Does It Work?

Security

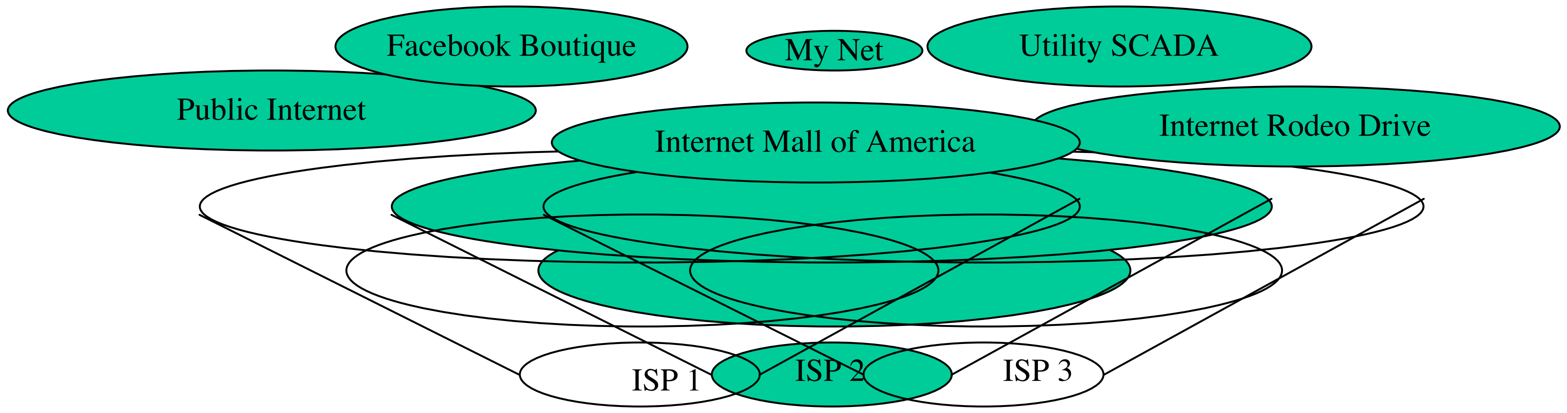


- The DIF is a securable container. DIF is secured not each component separately.
- Application only knows Destination Application name and its local port.
- The layer ensures that Source has access to the Destination
  - Application must ensure Destination is who it purports to be.
- All members of the layer are authenticated within policy.
- Minimal trust: Only that the lower layer will deliver something to someone.
- PDU Protection can provide protection from eavesdropping, etc.
  - Complete architecture does not require a security connection, a la IPsec.

How Does It Work?

Security

- A Hacker in the Public Internet cannot connect to an Application in another DIF without either joining the DIF, or creating a new DIF spanning both. Either requires authentication and access control.
  - Non-IPC applications that can access two DIFs are a potential security problem.



There is Much More, And Much More to Discover!

- A Claim: One will not find a structure that is both as rich and as simple as this that is not equivalent to it. Prove me wrong! ;-)

- An Invitation: Come explore it with us.
  - There is much to explore:
    - Working out the common object models for management
    - How it applies to different environments, especially wireless.
    - What are the dynamic properties?
      - Routing, congestion control
- Start with Patterns in Network Architecture, Prentice Hall
  - Then the “Reference Model” (3 sections) and
  - The protocol specifications all available for review
  - At www.pouzinsociety.org or
  - csr.bu.edu/rina