New and Not So New
Implications from PNA

John Day

Boston University[1]

IEEE CCW

2009

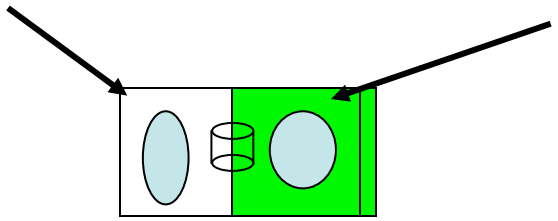
1 Supported in part by NSF
Oceans Observatory Initiative

Basis for the Results in PNA

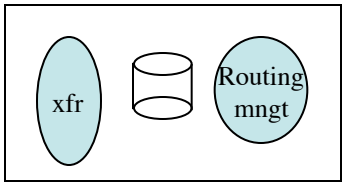
- Maximizing the Invariances and Minimizing the Discontinuities
- Separating mechanism and policy.
- Watson's result that bounding three timers are necessary and sufficient conditions for the synchronization req'd for reliable data transfer.
  - 3-way handshakes, SYNs and FINs are redundant.
- Saltzer [1982] shows that application names, node and point of attachment addresses are all required for a complete architecture.
- The distinction between Application Process and Application Entity
  - First uncovered by OSI (1985), first applied by the Web (1990).
- Connectionless is maximal shared sate, not minimal.
  - CO when traffic density is high, CL when it is low.
  - (in some sense nothing new)

How Did We Miss It?

- 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:



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.)

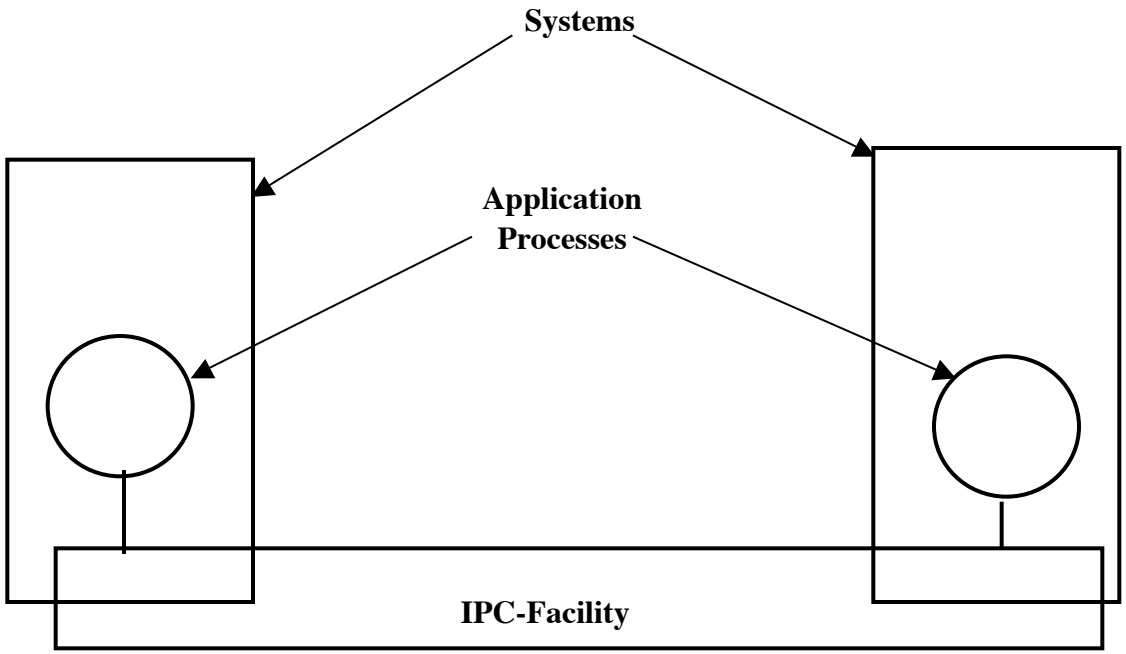
- The amount of state is about the same, although the amount of replication is different.
- We have been distributing connectivity information to every Node in a layer, but
- We have insisted in distributing resource allocation information only on a need to know basis, i.e. connection-like.
  - Even 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.)

Always Follow the Problem

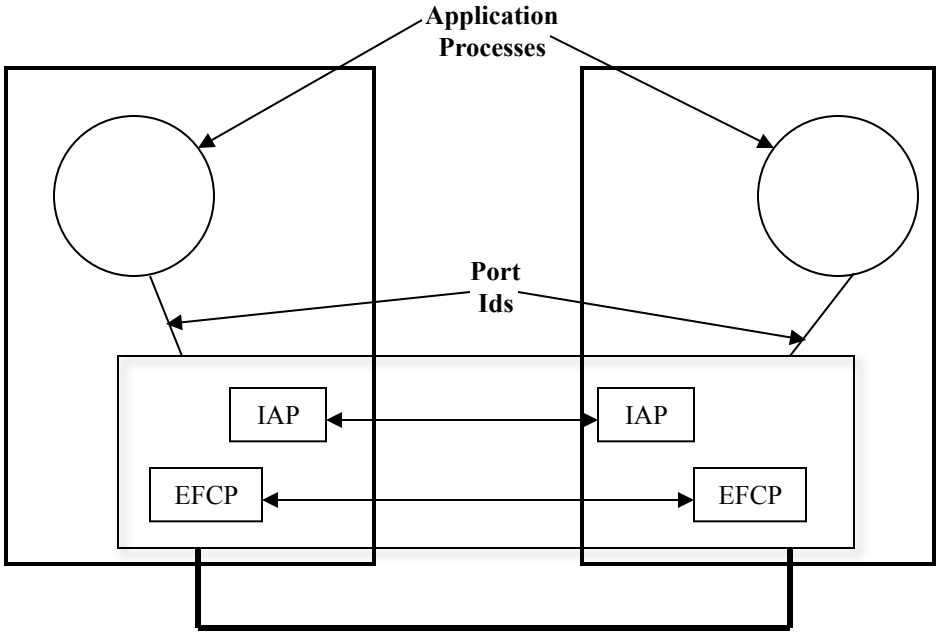
- Learning to Listen to the Problem is Not Easy.

- Starting around 1993, I had begun to see patterns I hadn't seen before, but . . .

We Go to IPC in Two Systems



That Means We Will Need

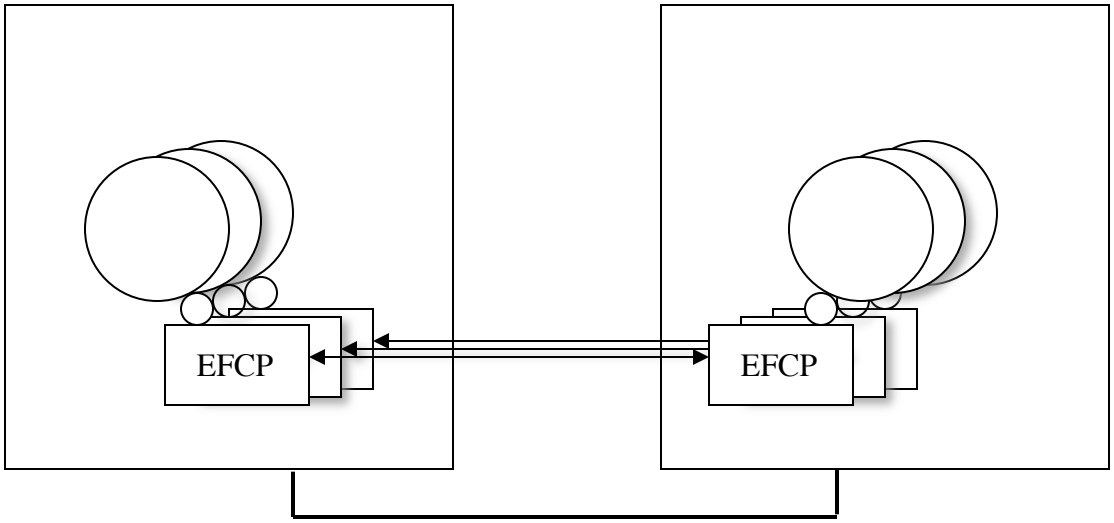


- A means to move data reliably and the means to find the requested application and determine access control.

3: Simultaneous Communication Between Two Systems

i.e. multiple applications at the same time

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



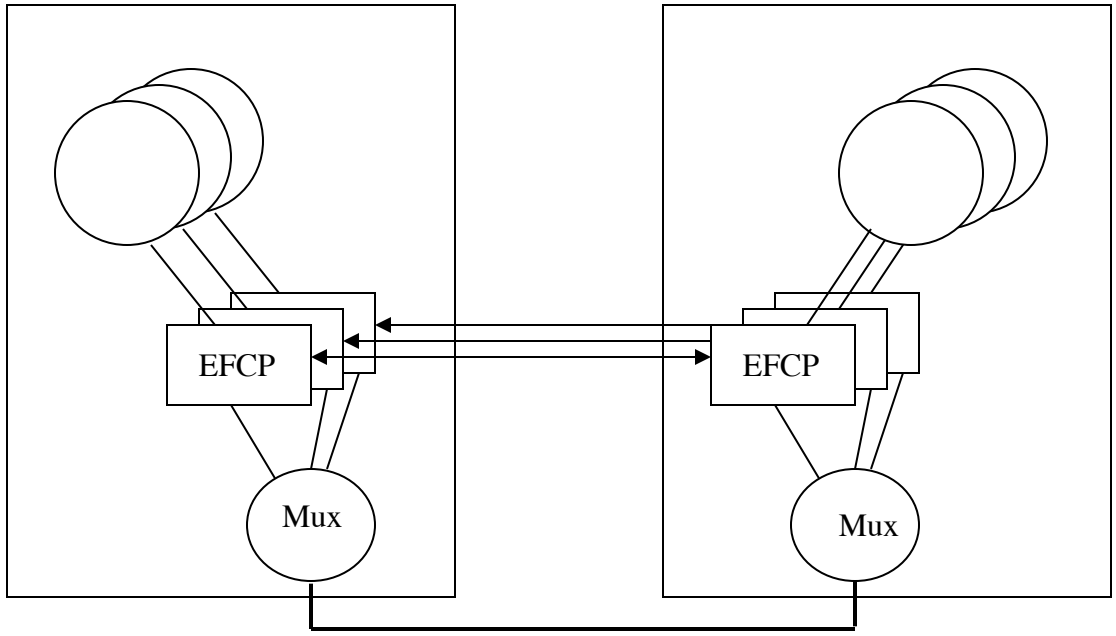
Will have to add the ability in EFCP to distinguish one flow from another.

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

Connection-id

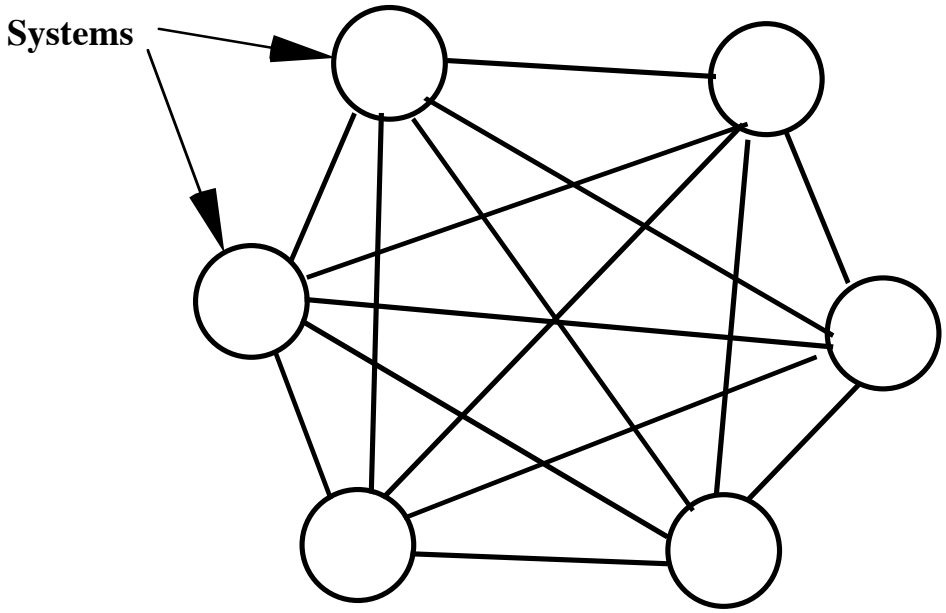
Dest-port	Src-port	Op	Seq #	CRC	Data

Then We Need Simultaneous Communication



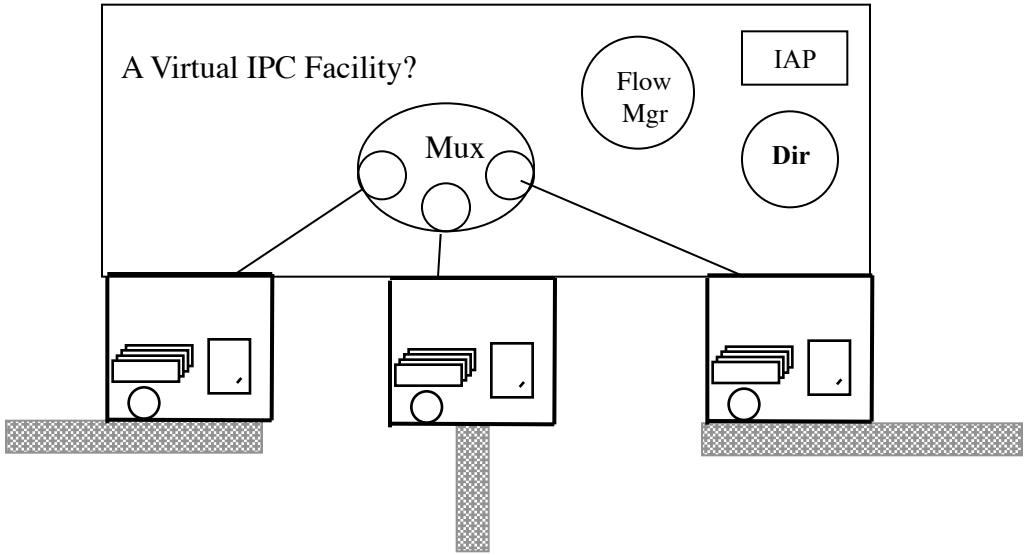
- This requires we add a connection (or flow) id and a task to manage the single resource, the wire.

Then We Move to N Systems



Remember Small Incremental Changes!
This leads to much more of the same,
so some organizing is worthwhile.

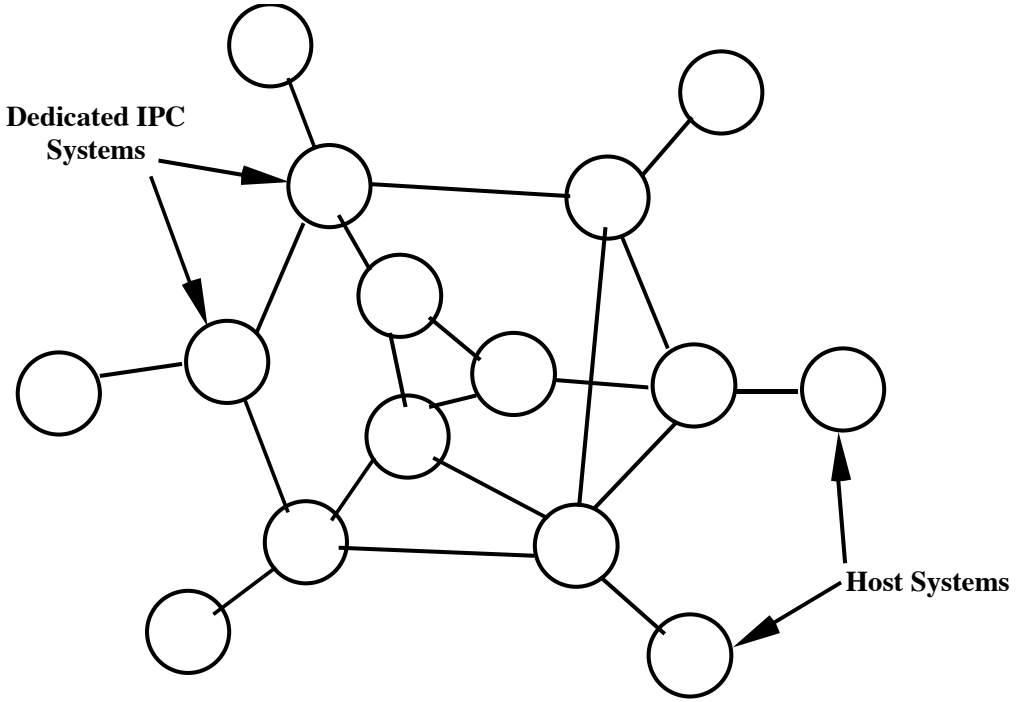
Need to Manage Multiple Interfaces



But we know this won't scale

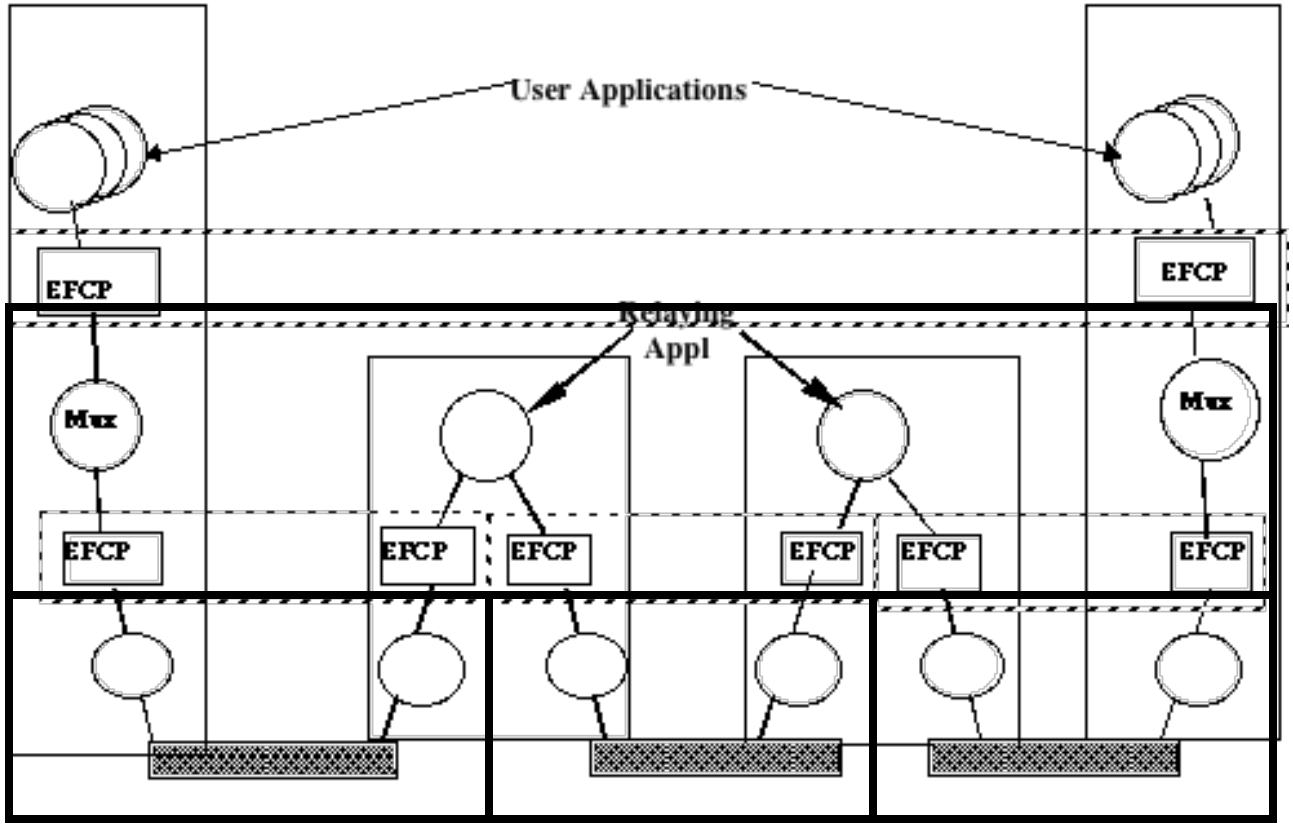
and will get very expensive.

N Systems on the Cheap



This requires dedicating systems to IPC (routers). Since there are losses in the routers, a higher level error control protocol.

The IPC Model



A General Theory of Networking

Distributed IPC Facilities

Implications: Layers

- Networking is IPC and only IPC. Hence a layer is the machinery for IPC for a given scope and range of operation.
- Layers are not a partitioning of functions. All layers have the same functions.
  - Not all instances of layers may need all functions
- A Layer is a Distributed Application that performs and manages IPC.
  - A Distributed IPC Facility (DIF)
- There is a single DIF that repeats with different scope and range of operation (quality of service).
- This yields an architecture that scales indefinitely,
  - i.e. any bounds imposed are not a property of the architecture itself.

The End-to-End Principle?

- It never came up.
  - It isn't a principle anyway, at best a lemma.
  - Really a statement of desire.
    - Original paper written to counter PTT contention Transport was unnecessary.
    - If there is a principle, it is relaying may require error control above it
- It is not really *wrong*, but it *is* an impediment.
- Its focus on the distinction of ends vs the middle, hosts vs the network has two deleterious effects:
  - It still has vestiges of "beads-on-a-string."
  - It has kept us from seeing the repeating nature of the structure.

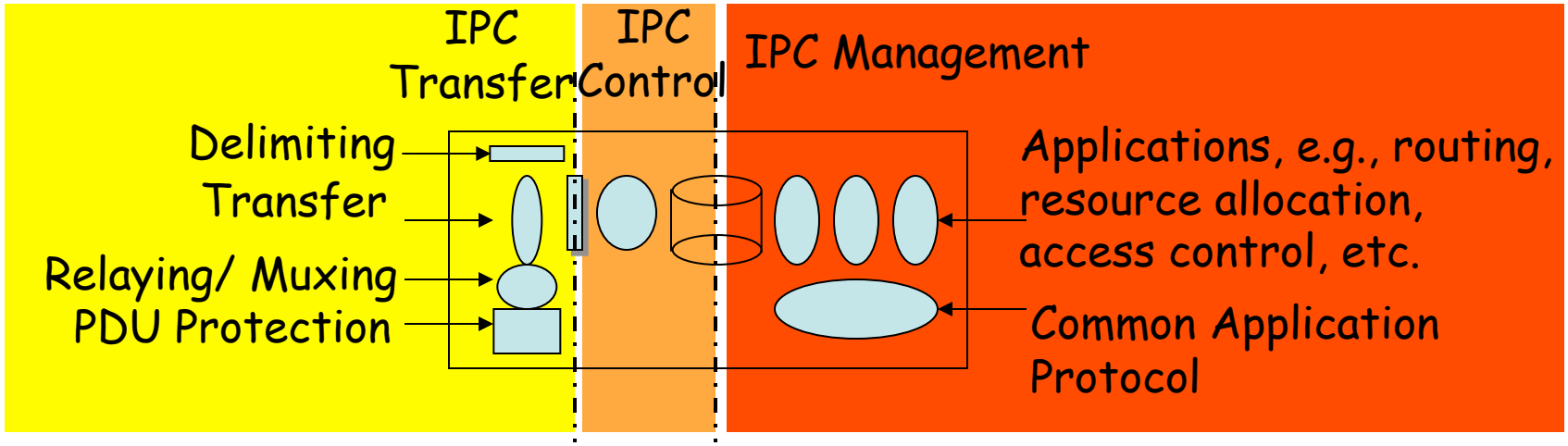
Implications: Protocols I

- 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:
    - Create/delete, Read/Write, Start/Stop
    - Good examples would be HEMS or CMIP
  - There is no need for a protocol like IP.
  - TCP was split in the wrong direction.
    - Every facet of TCP prevents a simpler solution to something else, usually because of overloading.
- 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

- Watson's results imply that all data transfer protocols are "soft-state."
  - At least all that are properly designed.
- "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 is 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.
- Watson's result also defines what is networking or IPC:
  - It is IPC if Maximum Packet Lifetime can be bounded.
  - It is remote storage, if MPL can't be bounded.
    - A very fundamental result

What a Layer Looks Like



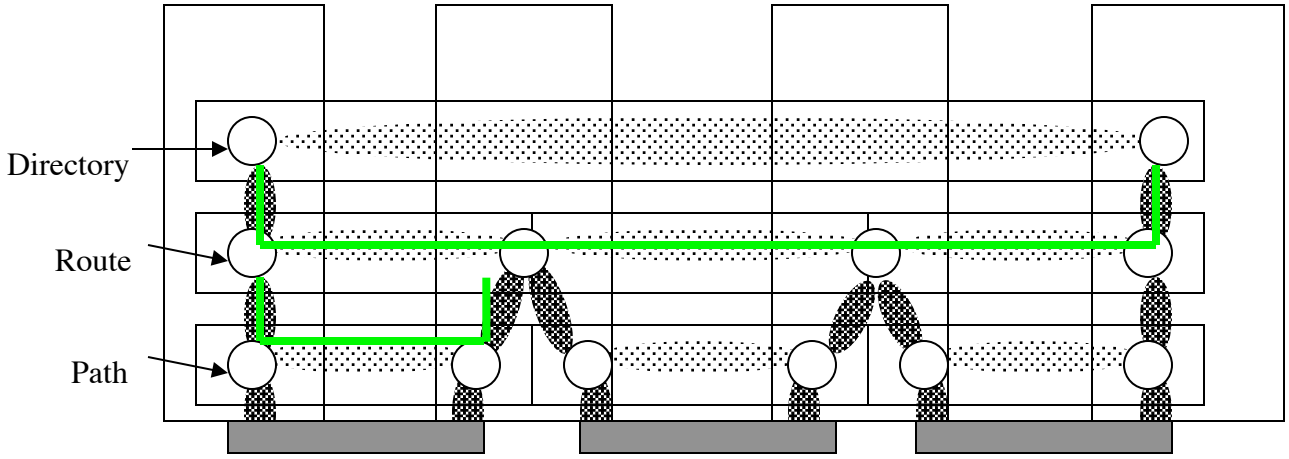
- Processing at 3 timescales, decoupled by either a State Vector or a Resource Information Base
  - IPC Transfer actually moves the data (≈ IP + UDP)
  - IPC Control (optional) for retransmission (ack) and flow control, etc.
  - IPC Layer Management for routing, resource allocation, locating applications,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

Implications: Naming and Addressing I

- The elements of a layer are cooperating application processes dedicated to performing and managing IPC.
- Addresses are synonyms of the Application Process Names that are members of a layer. Scope is limited to the layer and may be structured to facilitate use within the layer.
    - Can change an address without disturbing existing flows.
- 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
- Saltzer [1982] worked out the naming and addressing architecture, but missed a crucial point.

Directory, Routes and Paths

- 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

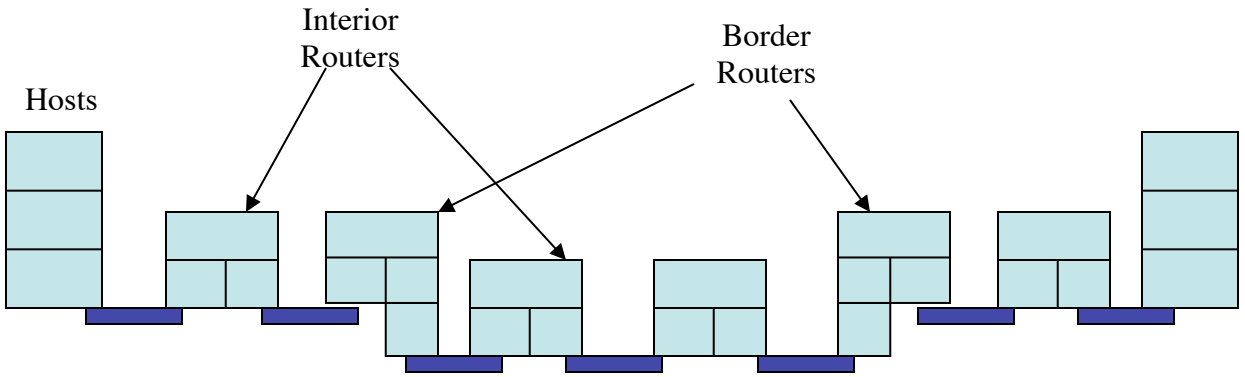
- A Global Address space is Unnecessary.
- While likely that an Application Process Name space will have greater scope than an address space, there is no architectural requirement that this be the case.
    - A global name space is not necessary either.

- 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.
- Deep packet Inspection is unnecessary.

Implications: Naming and Addressing III

- 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* and defeats the multipath properties that are necessary.
- 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.

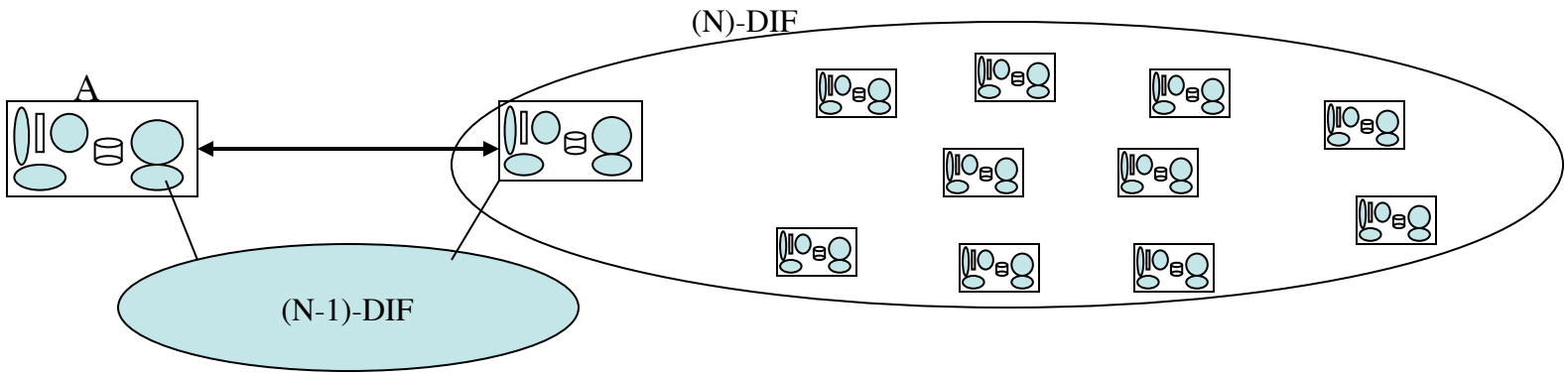
Only Three Kinds of Systems



- 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?

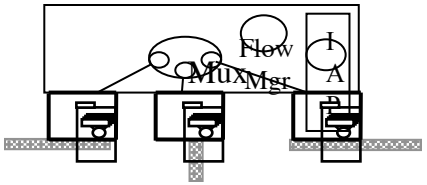
Joining or Creating a Layer



- 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

Choosing a Layer

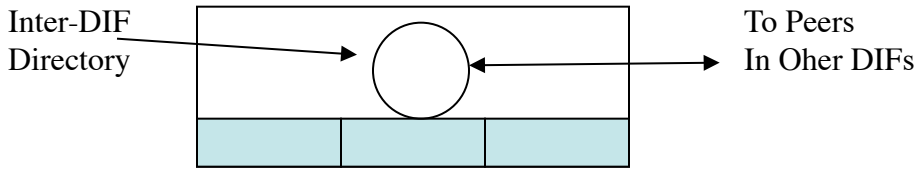
- In building the IPC Model, the first time multiple DIFs are encountered (data link layers in that case), it was found useful to have a task to determine which DIF to use.



  - 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.

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

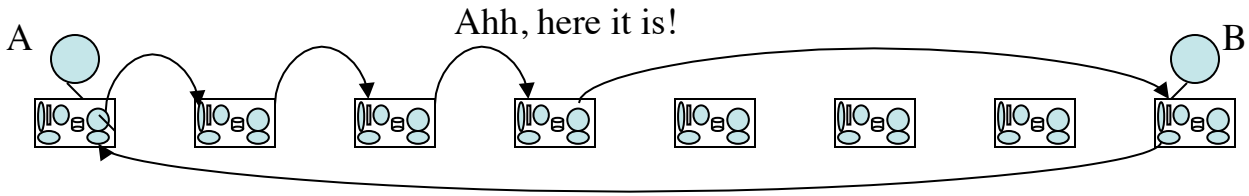


Actually one could still have distinct names spaces within a DIFs (synonyms) with its own directory database.

- But not all names need be in the Global Directory.
- A DIF could have its own parallel application name space and directory of distributed databases.
- The global name space would only rarely need to be consulted.
- This means that alternate application name spaces and directory schemes are not only possible, but useful.
- This is how networks get generated.

How Does It Work?

Establishing Communication

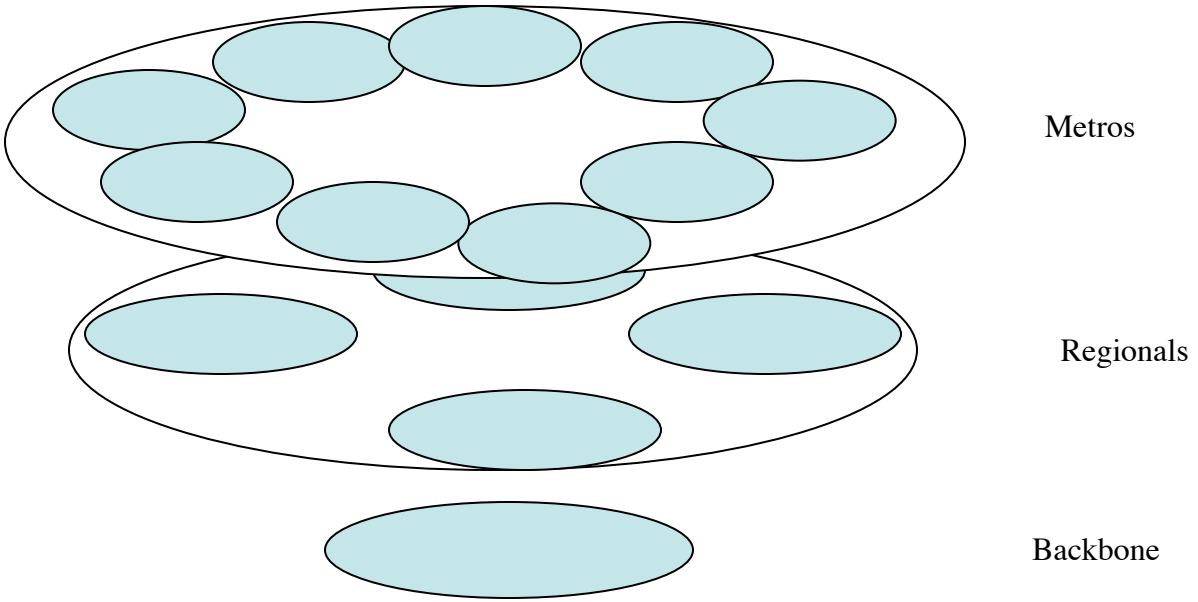


- Simple: do what IPC tells us to do.
  - A asks IPC to allocate comm resources to B
  - Determine that B is not local to A use search rules to find B
  - Keep looking until we find an entry for it.
  - Then go see if it is really there and whether we have access.
  - Then tell A the result.
- This has 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

How Does It Work?

The Internet and ISPs

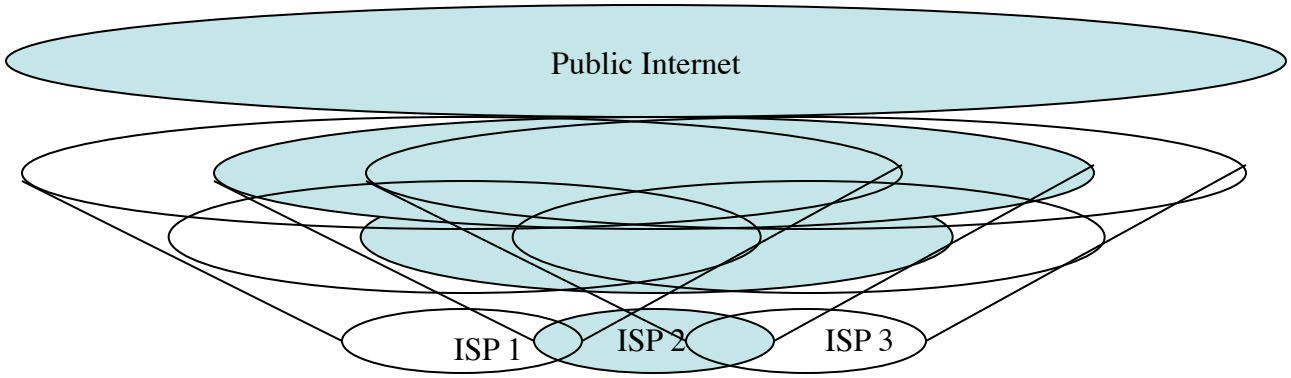
- ISPs have as many layers as they need to best manage their resources.



How Does It Work?

The Internet and ISPs

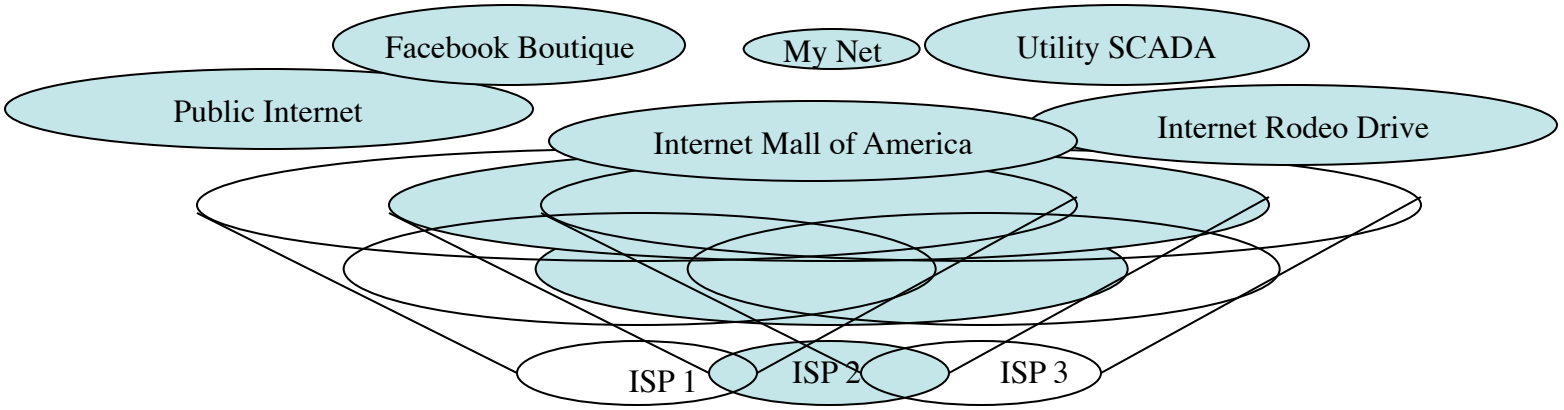
- The Internet floats on top of ISPs, a “e-mall.”
  - One in the seedy part of town, but an “e-mall”
  - Not the only emall and not one you always have to be connected to.



How Does It Work?

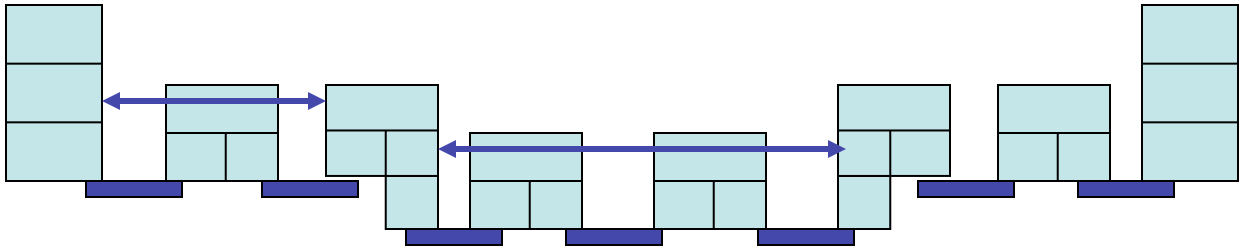
The Internet and ISPs

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



How Does It Work?

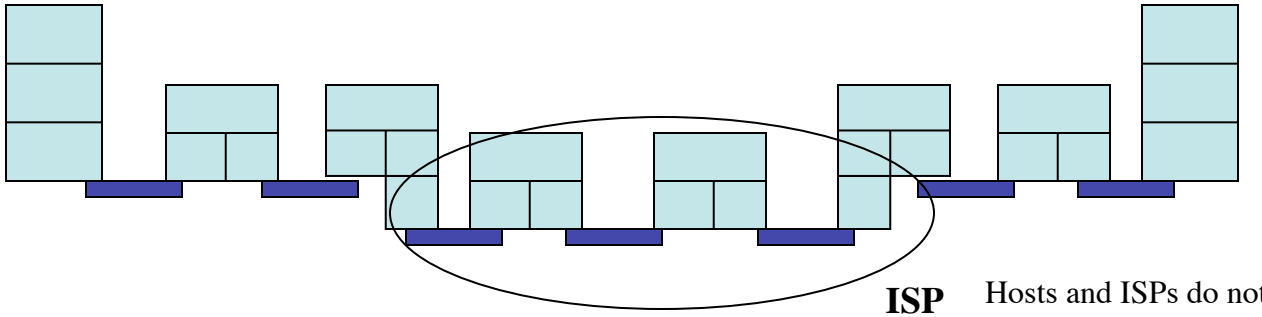
"Congestion Control"



- 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?

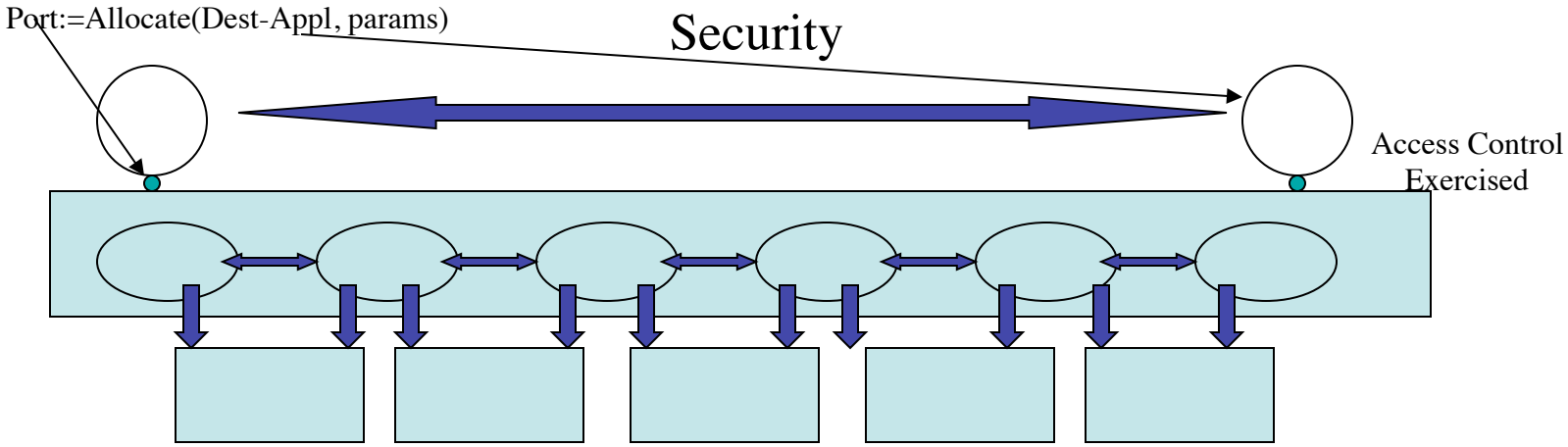
Security



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?

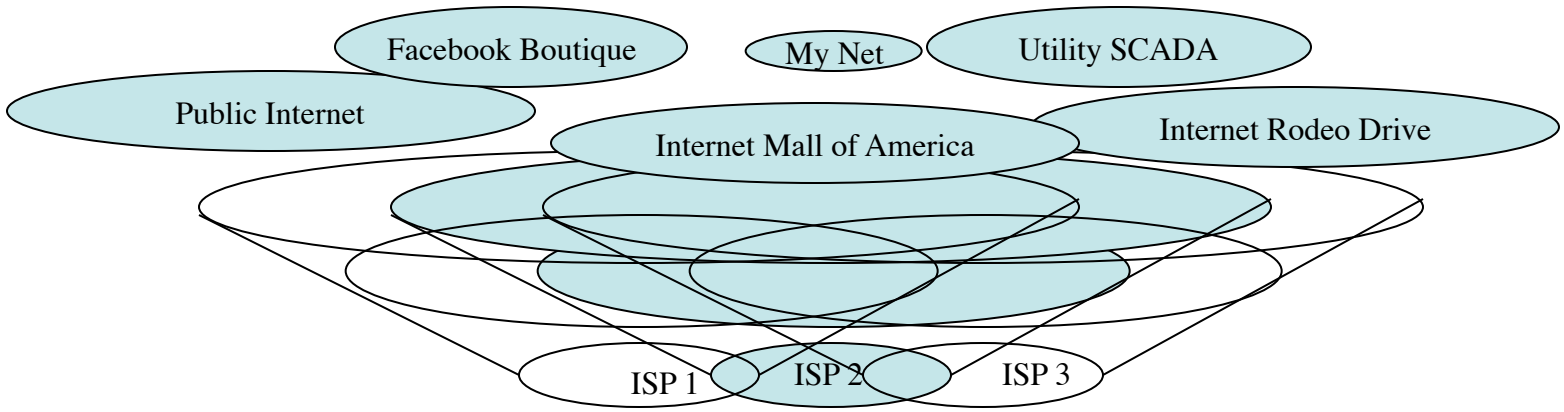


- The DIF is a securable container. DIF is secured not each component separately.
- Application only knows Destination Application name and its local port.
- The layer ensures that Source has access to the Destination
  - Application must ensure Destination is who it purports to be.
- All members of 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