"Networking is IPC": A Guiding Principle to a Better Internet

Networking is inter-process communication. —Robert Metcalfe, 1972

John Day[‡] Ibrahim Matta[†] Karim Mattar[†]
[‡]Metropolitan College [†]College of Arts & Science
Computer Science, Boston University
{day, matta, kmattar}@bu.edu

ABSTRACT

This position paper outlines a new network architecture that is based on the fundamental principle that *networking is inter-process communication* (IPC). In this model, application processes (APes) communicate via an IPC facility. The IPC processes that make up this facility provide a protocol that implements an IPC mechanism, and a protocol for managing distributed IPC (routing, security and other management tasks). Our architecture is *recursive* in that the IPC processes can themselves be APes requesting services from lower IPC facilities. We present the repeating patterns and structures in our architecture, and show how the proposed model would cope with the challenges faced by today's Internet (and that of the future).

1. INTRODUCTION

Today, the pure form of the Internet's best-effort delivery model has not been able to effectively respond to new requirements (*e.g.*, security, quality-of-service, wireless, mobility). Many individual networks in the Internet today represent commercial entities—Internet Service Providers (ISPs). An ISP may be willing to provide better than best-effort service to its customers or its peers for a price or to meet a Service Level Agreement (SLA). The lack of a structured view of how this could be accomplished has led to ad hoc solutions and so-called "layer violations" where in-network elements (*e.g.*, routers, proxies, middleboxes) deeply inspect passing datagrams so as to perform application- or transport-specific processing.

We believe that time is ripe to revisit the original Internet architecture to explicitly enable a richer internetworking model that empowers ISPs and creates an organized market structure.

Our Contribution:

We propose a new *architecture* for building networks and internets that has the following features. (We elaborate on these features in the remainder of this paper.)

1. It builds on a very basic premise, yet fresh perspective that networking is *not* a layered set of *different* functions but rather a *single* layer of distributed Inter-Process Communication (IPC) that repeats over different scopes. Each instance of this repeating IPC layer implements the *same* functions / mechanisms but policies are tuned to operate over different ranges of the performance space (*e.g.*, capacity, delay, loss).

   Remark: By IPC, we mean the general *model* of communicating processes, and not a specific implementation.

2. It is based on a comprehensive theory of networking; it does not represent another "fix", patch, or point-solution to a piece of the problem. We do not propose to simply add a new "session" layer to perform some extra functionality for bridging ISP networks. Instead we take a *clean slate* approach and begin with a comprehensive general theory of IPC where the number of IPC layers may vary at different parts of the Internet depending on the range of the resource allocation problem that must be addressed. The greater the operating range in a network, the more IPC layers it may have so as to exert more performance control. Thus configuring the appropriate number of IPC layers allows for more *predictable service* to be delivered to users.

3. This repeating structure scales indefinitely, thus it avoids current problems of growing routing tables due to the limited and hence more manageable scope of each IPC layer. It also supports features such as *multihoming* and *mobility*, with little or no cost.

   Remark: By "indefinitely" we mean that the nature of our proposed architecture does not impose any limits. There may, of course, be physical limits and other constraints.

4. An application process using this distributed IPC facility[1] only knows the name of the destination application process. It has no knowledge of addresses and there are no so-called "well-known ports." To join such a distributed IPC facility, a new member *must* be authenticated according to the policies of this particular facility. This yields a far more *secure* architecture.

5. Stacking IPC layers on top of each other allows networks to be built from smaller and more manageable layers of limited scope. This divide-and-conquer strategy gives providers more resource management options than just over-provision. Thus it provides the basis for operating subnetworks at much higher utilizations than the 30%–40% in the current Internet.

6. The distributed IPC facility that we propose here, can be configured to not only provide the fundamental services of the traditional networking lower layers but also

[1]We use the terms "IPC layer" and "Distributed IPC Facility (DIF)" interchangeably.

ReArch'08, December 9, 2008, Madrid, SPAIN

the services of application relaying (*e.g.*, mail distribution and similar services), transaction processing, and peer-to-peer. This removes the barrier created by the Transport Layer in the current Internet, opening potential *new markets* for ISPs to provide IPC services directly to their customers leveraging their expertise in resource management of lower layers.

7. Perhaps most surprising, it turns out that private networks (with private addresses) are the norm—IPC processes are identified by addresses *internal* to the distributed IPC facility—and public networks are simply a degenerate case of a private network. This lays the foundation for major competition and innovation and avoids the tyranny of the current Internet structure.

2. BACK TO BASICS: NETWORKING IS DISTRIBUTED IPC AND ONLY IPC

We all became familiar with the "layered" reference model of ISO OSI as well as the layered TCP/IP architecture. In these models, a layer is said to provide a "service" to the layer immediately above it. For example, the transport layer provides "virtual" end-to-end channels to the application layer, and the internetworking layer provides the transport layer with "physical" packet delivery across individual networks making up the Internet.

What's wrong with this layered model? As Robert Metcalfe's quote in the paper's subtitle indicates, we have always known that IPC was the core of the problem, but we somehow missed what it could tell us. Both the transport and internetworking tasks *together* constitute an IPC service to application processes. Let us call this an *Internet-wide IPC service*. Now, to implement such a service over individual ISP networks, we contend that one needs a similar *ISP-wide IPC service* over each ISP network. In other words, we need to repeat such an IPC service over different regions/scopes. Of course, an ISP, in turn, may manage its own network (perhaps large-scale and/or with a significant all-wireless component) by implementing IPC layers of narrower scope over a number of its own components.

We note that we are aware that "recursion" has been recently promoted in network architectures, but to the best of our knowledge, this has been limited to tentative proposals of repeated functions of *existing* layers, and how one may either reduce duplication or create a "meta"-function (*e.g.*, error and flow control) that could be re-used in many layers, *e.g.* Touch *et al.* (2006) [10]. Independently, we have pursued a general theory to identify patterns in network architecture [2] (1996). This proposal is based on this different direction [3]:

Application processes communicate via a distributed IPC facility. The processes that make up this facility provide a protocol that implements an IPC mechanism, and a protocol for managing distributed IPC (routing, security and other management tasks).

We need to view what repeats, as an IPC service, which combines transport (flow-based quality-of-service), routing (multiplexing/relaying), and other management functions. This enables each ISP (at any level, small or large) to sell its IPC-based services to others, thus promoting competition and an organized market-driven Internet.

3. PROPOSED IPC-BASED NETWORK ARCHITECTURE

3.1 Elements of a Two-System Scenario

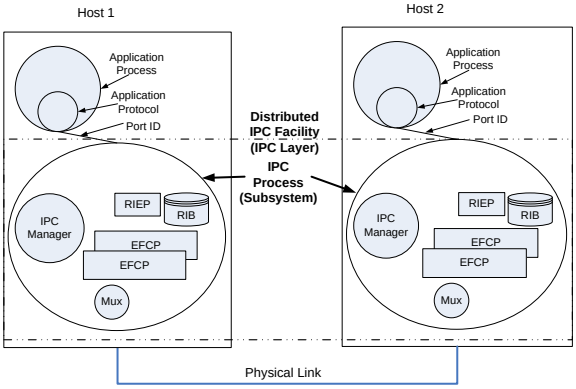

Figure 1: One layer of IPC consisting of hosts with user applications and IPC subsystems.

Figure 1 shows the elements of an IPC facility required for communication between two application processes in two hosts that are directly connected by a physical link. The *application protocol* part of the application processes establish communication using an IPC interface. This IPC interface allows the source application process to *name* the destination application process and specify desired properties for the communication. Application names should be *location independent*, and unlike existing IPC interfaces (notably the sockets interface), applications never see addresses. The job of the IPC facility is to:

- locate the destination application process using its name,
- if found[2], establish the communication channel and allocate resources required to meet the desired properties[3],
- return unique port IDs to the application processes to use to send/receive data over the allocated channel, and to release the channel when done.

Remark: Unlike existing IPC interfaces, it is not necessary to overload port IDs with application-name semantics. Here a port ID is simply a local, *dynamically* assigned, identifier that identifies one end of a channel/connection at the layer boundary.

To accomplish its job, the IPC facility needs mechanisms to support the following functions:

- an *IPC manager* to manage the various functions (discussed below) needed to establish and maintain connections,
- a *Resource Information Exchange Protocol* (RIEP) to populate a Resource Information Base (RIB) with application names, addresses, and performance capabilities, used by various DIF coordination tasks, such as routing, connection management, *etc.*,
- an *Error and Flow Control Protocol* (EFCP) to support requested channel properties during data transfer,
- a *multiplexing task* to efficiently use (schedule) the underlying IPC facility (communication medium) that is shared among several connections.

[2]if the destination application is found but is not available, the IPC facility could start it.
[3]Resources could be allocated in many different ways, including best-effort, differentiated, or guaranteed services.

How is this Distributed IPC Facility (IPC layer) different from the traditional definition of a layer?

- First, the proposed IPC layer does not perform a single function or a small subset of pre-determined functions, but a coordinated set of functions to achieve the *desired* IPC service.
- Second, the proposed IPC layer naturally separates various concerns, including operation over different timescales (*e.g.*, short-term data transfer and multiplexing vs. long-term connection management and access control issues).

3.2 Elements of a Multi-System Scenario

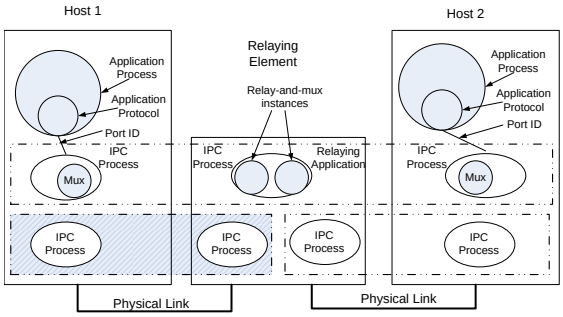

Figure 2: Two IPC layers consisting of hosts with user applications and IPC subsystems.

Figure 2 shows the elements of an IPC facility in the general case when dedicated systems (such as routers) are used to scale communication among a large number of hosts. In this figure, two hosts are connected via a router. There are two levels of IPC layers: two bottom IPC layers (DIFs), one tailored to each physical link, and one higher-level IPC layer that provides application processes with end-to-end communication through the router. The higher-level IPC processes communicate using the services provided by the bottom IPC layers.

In general, any system that has multiple interfaces would have a separate IPC process (with its own multiplexing task) for each interface—each one of these IPC processes would implement *policies* that are appropriate to its associated interface. Furthermore, to manage the multiple IPC processes, a multi-interface system would have a higher-level IPC process that performs not only multiplexing but also a relaying function. Thus, to manage the IPC channel through a multi-interface system (*e.g.*, a router), a relaying-and-multiplexing task would be required to forward messages toward their destination.

How is this two-layer IPC model different from the traditional transport-IP-interface model?

- First, the relaying applications in dedicated systems need to be *named* so an IPC process can specify the name of the next system/node along the path toward the destination application process. Given the name of the next multi-interface (multi-homed) node, the underlying (lower) IPC layer could then choose one path, out of the multiple paths leading to the various interfaces. Notice that the difficulty of the current Internet architecture to support multihoming is resolved by simply keeping the distinction between IPC facilities clear.
- Second, the relaying processes in the dedicated systems, as well as the multiplexing processes in end-hosts, are all members of the same distributed IPC facility. This IPC facility integrates *both* transport and routing tasks, along with other management tasks. Thus an IPC facility is capable of fully supporting flow-based performance requirements if so desired[4]. The current Internet architecture suffers from the complete separation of transport (flow and error control) tasks and multiplexing/routing tasks into separate layers, thus misses opportunities for sharing conditions the two sets of tasks are observing.
- Third, an interesting consequence of our model is that since the communicating elements are application processes, they have application names. Hence, we see that application names are external names, while addresses (of multiplexing and relaying IPC processes) are *internal* identifiers used by the members of the DIF to facilitate coordination among themselves. Hence there is no reason for an address to ever be visible outside its DIF.
- Fourth, as noted earlier, to become a member of a distributed IPC facility, an IPC process in our proposed architecture needs to explicitly enroll, *i.e.*, authenticated and assigned an address. The current state of affairs employs ad hoc procedures for enrollment that do not naturally fit within the current Internet architecture.

We discuss all the features of our proposed architecture in more detail in Section 6.

4. REPEATING IPC LAYERS

To summarize our proposed IPC model so far, we are fundamentally concerned with application processes communicating via a distributed IPC facility. Since the IPC layers repeat, the IPC processes within an IPC facility are in turn the application processes requesting service from the IPC layer below. In other words, the IPC facility is itself composed of application processes in different systems, whose coordinated behavior creates a distributed application for allocating IPC resources. The IPC layers repeat until the IPC facility is tailored to the physical medium.

Each IPC process consists of three distinct sets of tasks dealing with IPC aspects at different timescales. These task sets are loosely coupled through an information base and per-flow state:

- an IPC **Data Transfer**, which supports multiplexing (scheduling) and relaying (forwarding), and per-flow data transfer,
- an IPC **Transfer Control**, which implements EFCP and controls the per-flow data transfer parameters,
- an IPC **Management**, which implements RIEP to query and update a Resource Information Base for routing, security, resource management, address assignment, and so on.

An IPC layer has a rank (its position relative to other layers) and a scope (the collection of IPC processes that make up the IPC facility). As rank decreases, scope tends to decrease and hence the IPC facility could be more tightly managed as more control can be exerted over its resources. We have all observed that specific policies are only effective over a limited range (*e.g.*, closed-loop control is more effective/stable for shorter feedback loops). As the Internet has grown we have tried to accommodate an increasingly wider range of the resource allocation problem with one set of policies in one layer (notably the Internet Layer). In our

[4]By the term "flow" we mean an aggregated stream of messages that have been multiplexed at the edges of the IPC facility.

proposed architecture, the IPC layers (DIFs) can be applied to different ranges of the problem, effectively by a divide-and-conquer approach.

Figure 3 shows a repeated instantiation of IPC layers, where bottom IPC layers are tailored to the physical media, including wireless links. Higher IPC layers are built on top of lower layers, for example a higher-level DIF is configured over (and tailored to) the bottom wireless DIFs. This figure illustrates how an underlying channel between two hosts could be more effectively managed by repeating / adding an extra IPC layer whose range is relatively narrower and thus policies appropriate to that range's characteristics (*e.g.*, capacity, delay, loss) could be associated with the various multiplexing, relaying, error control, flow control, and management mechanisms.

In terms of today's roles of network elements, systems directly connected to the end-hosts in Figure 3 act as "border" ("access") routers, whereas the middle system acts as an "interior" router. It is important to note that we are not necessarily advocating more "layers" than what we have today, but we are viewing layers as IPC facilities which network elements explicitly become members of.

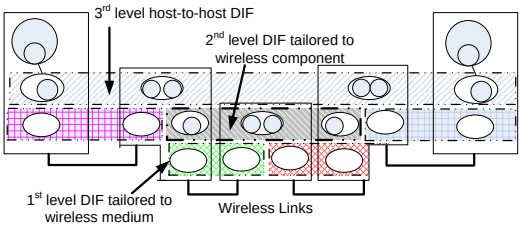

Figure 3: Layers of IPC consisting of hosts with user applications and IPC subsystems. More IPC levels exert more control over part of the host-to-host connection.

To summarize, ***our proposed IPC layers are not so much isolating different functions, like existing architectures, as they are supporting different ranges of the resource-allocation problem.***

5. OPERATION OF A DISTRIBUTED IPC FACILITY (DIF)

5.1 Creating a New DIF

To create a new distributed IPC facility at rank N, (N)-DIF, a higher-level (network management) application could create an initial IPC process and connect this process to one or more (N-1)-DIFs. This initial IPC process could then be directed to initiate enrollment with other IPC processes or simply wait for other IPC processes to join it (as described next).

Remark: Note that a global *name space* is needed to name applications / services with wider scope, so as to create a new / wider-scoped higher-level DIF. There is no need for a global *address space*.

5.2 Adding a New Member to a DIF

For a new IPC process, x, to join an existing (N)-DIF, x has to be connected to the (N)-DIF by an underlying (N-1)-DIF. Furthermore, like any other application, x has to know the name of the (N)-DIF or the name of some member of it, say y, but *not* the address. x attempts to establish a connection to y. Once this connection is established, y authenticates x. If the authentication is successful, y assigns x an (N)-address, and x becomes a member of the (N)-DIF.

Remark: This is not creating a connection-oriented architecture. This connection is purely for purposes of enrollment. It has no effect on the nature of forwarding decisions. It is no more connection-oriented than having a cable between two routers makes the router connection-oriented.

5.3 Transferring Data within a DIF

As described earlier, a distributed IPC facility provides an application process with an interface to establish a connection to a destination application process. Unlike the current Internet architecture, which looks up a name in DNS and returns the result to the requester, here, once an address has been found, the request continues to the identified IPC process to ensure that the application is really there and that the requester has access to it. This is analogous to what IPC in a single system does and here it has many additional benefits as well, such as access control, handling an application that has moved, imposing resource constraints, *etc.*

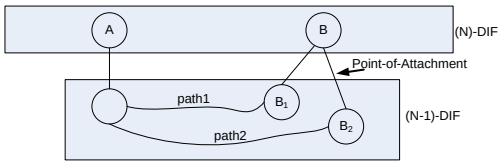

Figure 4: Two-step routing process.

Routing within an (N)-DIF is done over a graph of IPC processes that are members of this IPC facility. A routing path toward a destination (N)-IPC process is specified by the internal addresses of the (N)-IPC processes along the path.[5] To facilitate routing, we would want to route over a topology that is perhaps more stable than the, typically time-varying, graph of IPC processes. To that end, we believe internal addresses should be *topological (location-dependent)*, rather than today's provider-based / graph-based addresses.

Figure 4 illustrates the routing process from an (N)-IPC process A to its next-hop (neighbor) B along the path to the destination using the services provided by the underlying (N-1)-DIF. When the (N-1)-DIF relays the message to B, the (N-1)-DIF maps the process name of B to an (N-1)-IPC process name (or (N-1)-address), B_1 or B_2, that corresponds to the specific path within the underlying (N-1)-DIF.

Remark: Unlike the current Internet architecture, the IPC architecture is *relative*. The (N-1)-address is internal to the (N-1)-DIF and is considered a "point-of-attachment" (PoA) address for the (N)-DIF.

6. FEATURES OF OUR ARCHITECTURE

Contrary to other proposals where many aspects require specific mechanisms to accomplish specific capabilities, in this proposal many capabilities are accommodated without specific mechanisms but as a consequence of the structure. As one would expect in a complete architecture, so-called middleboxes are unnecessary, and so-called "layer violations" do not exist.

6.1 Security

In our proposed model, applications never see addresses, which are *private* to the underlying IPC facility. Thus, the

[5]How paths are chosen is a matter of policy.

IPC facility is less impervious to attacks from outside the facility. This is contrary to the vulnerability of the current Internet infrastructure to attacks by hosts, because its IP addresses are made public.

Of course management applications are the exception in our architecture, as they themselves authenticate and assign those "internal" addresses to other IPC processes when they join the distributed IPC facility (and could revoke these addresses if malicious behavior is detected). Moreover, different authentication policies could be employed within each facility, thus providing a range of security levels from public (as in the current Internet) to private.

In addition, the kludge of firewalls to create security domains is avoided in our architecture. In our model, *firewalling is a natural function of a border router, where there is a repetition of the DIF structure.*

6.2 Manageability

As noted earlier and illustrated in Figure 3, our IPC model would support better control over resources by repeating the IPC layer over smaller scopes/regions. Nowadays, ad hoc proxy-based techniques have been deployed to improve performance over wireless or long-fat pipes. In our model, *proxying is a natural function of a border router, where there is a repetition of the DIF structure.*

To overcome the crippling effect of the traditional layered model that isolates transport functions from multiplexing, relaying and routing functions, so-called "cross-layer" approaches have been proposed. Our model integrates all IPC tasks into one layer and communication between IPC processes enables service negotiation. We are already finding that commonality across layers has a simplifying effect on manageability.

6.3 Multihoming

Multihoming (*i.e.*, having more than one connection to the network) has been challenging to support within the current Internet architecture because the Internet has an incomplete naming and addressing architecture (see Saltzer's work [7])—the IP address names the interface rather than the node itself (the IPC process in our model).

To address this naming/addressing problem, many ad hoc techniques have been proposed. For example, SCTP [9] supports the ability to change the IP address (of a host interface) without disrupting the transport connection. However, there is no easy way for SCTP to know that a host interface has failed so it could initiate a switch to another interface, nor it is its job to do so as this requires SCTP to do, at a minimum, degenerate routing. This is again because of the strict layered model that isolates transport from routing.

In our IPC model, the naming of IPC processes names nodes by necessity. Routing in the (N)-DIF is then naturally done in terms of (N)-addresses. Interface names are (N-1)-IPC process names. As pointed out, this relation is relative. An (N)-IPC process may have access to more than one (N-1)-DIF (interface), and more than one (N-1)-DIF may be directly connected to the same adjacent (N)-IPC process. In other words, there may be more than one path to the next hop. Multihoming is solved by simply recognizing that building the forwarding table is a two-step process of picking the next hop and then selecting the path to the next hop (cf. Figure 4).

The *late binding* from the (N)-address to the (N-1)-address (or (N)-PoA address), and the *narrower scope* of the underlying (N-1)-DIF make mechanisms for re-routing through a different path toward a multi-homed node much more scalable.

6.4 Mobility

With the expected proliferation of a number of competing wireless access subnets (*e.g.*, urban WiFi, cellular WiMAX), the future Internet should be able to intrinsically handle mobility. Mobility has been a challenge for the current Internet architecture because again, it only names interfaces and not the nodes themselves. For example, in the Mobile-IP solution [1], the IP address of the mobile is treated as a "special" case by the home and foreign routers which themselves constitute two single points of failure.

We note that *mobility is simply dynamic multihoming*, whether to different base stations, switches, or different subnets. In a sense, mobility is dynamic multihoming with controlled "link failures", *i.e.*, as a wireless signal becomes weak, the link "fails".

Figure 5 illustrates the mobility of a host M from right to left. The host belongs to multiple DIFs of different rank, *i.e.*, the host has multiple node addresses. As a mobile host moves, it joins new DIFs and drops its participation in old ones. For example, by moving from one subnet to another, a node at the (N)-layer changes its PoA or its (N-1)-address, but its (N)-address may not change if the movement is local. In general, PoA addresses change more frequently in DIFs of lower rank (smaller scope). As the figure shows, a movement of host M within the scope of the right (N-1)-DIF results in a *local* routing update at the (N-1)-process A since M's (N-2)-address changes. Further movement to the left (N-1)-DIF results in a routing update at the higher (N)-process B since M's (N-1)-address changes.

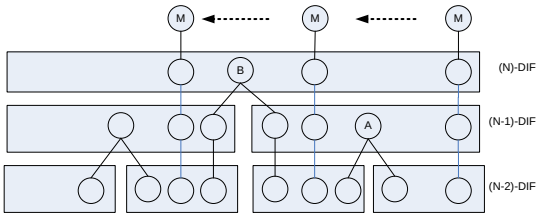

Figure 5: As a mobile host moves, it joins new DIFs and drops its participation in old ones.

As noted earlier, our IPC model would effectively support multihoming, thus it would also support mobility. This mobility support requires the cooperation of *nearby* ISPs (DIFs) to re-route messages to the mobile node, but occurs naturally as a consequence of routing updates within a *narrow scope.*

6.5 Scalability

Given the repeating nature of our IPC model, where each IPC facility has its own *private* internal addresses / identifiers, and with management policies that constrain the membership size of each IPC facility, we expect much better address scalability compared to that of the current Internet. The kludge of Network Address Translation boxes (NATs) is avoided in our architecture. In our model, *NATing is a natural function of a border router, where there is a repetition of the DIF structure.* None of the problems NATs cause in the Internet exist in our model, even though private addresses are the norm, because there is a complete addressing architecture.

In addition, as discussed above, the narrower scope of topology changes and late binding from node name (address) to PoA address (interface) make our architecture much more scalable in terms of routing overhead.

6.6 A Competitive Marketplace

The layered best-effort architecture adopted by the Internet relegates ISPs to a commodity business. The Transport Layer effectively seals ISPs off in the lower layers with IP providing a best-effort service. The current view of IP makes over-provisioning the only effective response to maintaining service and makes differentiation nearly impossible, leaving ISPs squeezed between the application/content (host) providers and the equipment vendors with little room to maneuver. This does not contribute to a healthy market.

Our proposed IPC model promotes a healthy marketplace by getting ISPs into the business of IPC services. While the distributed IPC layers can be used by ISPs to better manage resource allocation in their networks, the same functions appear in what are now called application relaying (*e.g.*, email), transaction handling (*e.g.*, checkpointing and two-phase commit), peer-to-peer, *etc.* This allows ISPs to expand into what has traditionally been a purely host service, leveraging their knowledge of resource allocation in the layers below.

6.7 Adoptability

An IPC facility in our model is merely a private network. In this model, the current Internet is simply a specific private layer with very weak requirements for joining it. Consider the Internet as an example of a vast e-mall. Other e-malls are possible with other characteristics such as tighter security for joining, perhaps specialized to certain market segments. There is no public network or address space that one must belong to, no single network one must always be attached to. A user could be part of any network by choice. In one's network, an IPC service provider is in control of everything including the addresses. Networks have considerable flexibility in *how* they provide their services. This would encourage alliances among groups of providers with complementary interests to provide advanced services in competition with groups of other providers.

7. RELATED WORK

The addressing architecture in our proposed IPC model is inspired by Jerry Saltzer's work on *Naming and Binding of Objects* (1978) [6]. Although, in this work, Saltzer does not consider computer networks, he noted the important concepts of program name, its logical (virtual memory) address, and its physical (memory) address. John Shoch in 1978 [8] noted three distinct concepts in computer networks: location-independent names ("what we seek"); location-dependent names ("where it is"); and routes ("how to get there"). In 1982, Saltzer [7] revisited his naming and addressing concepts for computer networks. Saltzer noted four levels, not three: services and users, nodes, network attachments, and paths. So he recognized that the same node may change its attachment to the network, and the node address should be independent of that. Saltzer identifies mappings from service name to node name, from node name to PoA, then from PoA to path. At that time, Saltzer lumped the last two mappings into one routing step and did not clearly distinguish between node name (address) and its point-of-attachment, understandably as it was rare to have multiple paths (links) to the next-hop node along the route. In our model, *routing is a two-step process*: a route is first determined as a sequence of node addresses, then to reach the next-hop (*i.e.*, next node address) along the route, one has to choose a path, possibly out of many possible paths (points of attachment), toward the next-hop node. The PoA address is in turn a node address in the underlying IPC layer. Although some recent work on the so-called location/identifier split [4] has attempted to address this long-standing naming/addressing problem, we believe the new model we are proposing naturally and cleanly solves this problem by adopting (and expanding) Saltzer's approach and generalizing it in the context of the *repeating* IPC model.

Though our IPC model may invoke some resemblance to "middleware", our model goes well beyond by *recursing* the functionalities of all lower layers of networking. Policies determine how much of these functionalities is activated at each level. The repetitive nature of our IPC model may also invoke some resemblance to the practice of tunneling. Our model is significantly different in that what we encapsulate is an IPC layer within another, where the inner IPC is totally invisible in terms of routing, management, *etc.*

8. ONGOING AND FUTURE WORK

We are currently specifying the complete operation of a distributed IPC layer, along with the various mechanisms and interfaces that are needed to support a variety of policies over different scopes. By separating mechanisms from policies, as well as separating concerns over different timescales (packet-level vs. connection-level vs. management), we can enable users to specify (and the community to contribute) IPC *policies declaratively* within our IPC framework, as we have recently done in [5] for transport policies.

We believe we have worked out enough design and specification details that a fast implementation could be developed over a lightweight process substrate with minimal or no context-switching overhead, data copying, *etc.* We also plan to develop performance and economic models to evaluate our proposed architecture and compare it to today's architecture as well as newly proposed ones.

9. ACKNOWLEDGMENT

This work has been partially supported by National Science Foundation awards: CISE/CCF #0820138, CISE/CSR #0720604, CISE/CNS #0524477, CNS/ITR #0205294, and CISE/EIA RI #0202067.

10. REFERENCES

[1] Ed. C. Perkins. IP Mobility Support for IPv4. Internet RFC 3344, August 2002.
[2] J. Day. Patterns in Network Architecture I, II, III. Presentation Slides, SC6 in Seoul Korea, NIST, BBN, November 1996.
[3] J. Day. *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2008.
[4] D. Farinacci, V. Fuller, D. Oran, and D. Meyer. Locator/ID Separation Protocol (LISP). Internet Draft, November 2007.
[5] K. Mattar, I. Matta, J. Day, V. Ishakian, and G. Gursun. Declarative Transport: No more transport protocols to design, only policies to specify. Technical Report BUCS-TR-2008-014, CS Dept, Boston U., July 12 2008.
[6] J. Saltzer. Naming and Binding of Objects. In R. Bayer, editor, *Operating Systems, Lecture notes in Computer Science*, volume 60. Springer-Verlag, New York, 1978.
[7] J. Saltzer. On the Naming and Binding of Network Destinations. In *International Symposium on Local Computer Networks*, pages 311–317, April 1982.
[8] J. Shoch. Inter-Network Naming, Addressing, and Routing. In *IEEE Conference on Computer Communication Networks*, pages 72–79, Washington DC, 1978.
[9] R. Stewart and C. Metz. SCTP: New Transport Protocol for TCP/IP. *IEEE Internet Computing*, 05(6):64–69, 2001.
[10] J. Touch, Y-S. Wang, and V. Pingali. A Recursive Network Architecture. Technical report, USC/ISI, October 2006.