The Design Space of Network Mobility

Pamela Zave
AT&T Laboratories—Research
Florham Park, New Jersey, USA
pamela@research.att.com

Jennifer Rexford
Princeton University
Princeton, New Jersey, USA
jrex@cs.princeton.edu

March 31, 2013

Abstract

The Internet is increasingly mobile. Because mobility is difficult to implement at Internet scale, there is a large and confusing landscape of mobility proposals which cannot easily be compared. This tutorial is based on the geomorphic view of networking, which is a uniform framework for describing many key aspects of network architecture. The geomorphic view shows that there are two distinct patterns for implementing mobility, each with its own range of design choices and costs *versus* benefits. We use these patterns to classify and explain a representative sample of mobility mechanisms, abstractly yet precisely. The patterns also serve as a basis for evaluating properties of these mechanisms such as resource costs and scalability, and for considering composition of mobility mechanisms.

1 Introduction

The Internet is increasingly mobile. Users access Internet services from mobile devices that move from one wireless access point to another, or switch between WiFi and cellular network connectivity. Ubiquitous computing relies on sensors and actuators attached to vehicles, portable objects, and animals as well as people. Services increasingly run on virtual machines that can migrate from one physical server, or even one data center, to another. Modern interactive applications increasingly demand "seamless" mobility—the ability to retain an application-layer session despite changes in the underlying locations of the endpoints. Yet the main Internet protocols were designed in an era when each machine attached to a single, fixed network location with a topology-dependent address.

In recent years, many mechanisms have emerged for supporting mobility. A recent survey [1] cites 22 Internet protocols dating from 1991 to 2009, including most prominently Mobile IPv4, Mobile IPv6, MSM-IP, HIP, MOBIKE, Cellular IP, HAWAII, ILNP, and LISP Mobile Node. In other contexts mobility is supported by:

- Ethernet LANs and VLANs, which allow an interface to retain its IP address as the host moves within the LAN;
- injecting the IP address of a mobile machine into existing routing protocols such as OSPF and BGP [2, 3];
- the General Packet Radio Service (GPRS) Tunneling Protocol (GTP), which supports mobility in most cellular networks;
- the scalable "flat" routing architectures SEATTLE [4], PortLand [5], VL2 [6], NVP [7], and Rbridges/TRILL [8, 9];
- other research proposals including TCP Migrate [10], Serval [11], and the Internet Indirection Infrastructure [12];
- application-layer protocols such as the Session Initiation Protocol (SIP) [13].

Each well-known proposal tends to spawn a family of variants, so the total number is probably in the hundreds and growing.

These various mechanisms operate in different layers, and make different assumptions about naming, routing, session protocols, the scale of the system, security, and the cooperation of remote endpoints or multiple administrative domains. Because the community lacks a common framework for describing and comparing mobility mechanisms, their relationships are poorly understood. Comparisons tend to be based on superficial characteristics rather than inherent ones. Quantitative comparison must be based on labor-intensive prototyping and measurement or simulation.

In this book chapter, we explain mobility in a new and more useful way. Our primary goal is to enable a precise discussion of mobility designs. We provide descriptions of many prominent mobility mechanisms that are sufficient to convey the richness of the design space and the important engineering tradeoffs, while stopping short of delving in to the considerable minutiae in each protocol.

First (Section 2), we present a common framework for describing mobility mechanisms and other aspects of networking, called the *geomorphic view* of networking. This framework is both comprehensive and precise, resulting in a unique description of each design that omits superficial detail while exposing subtle differences.

Second, we define mobility and present the two major patterns—*dynamic-routing mobility* and *session-location mobility*—for implementing it (Section 3). In the context of IP, an endpoint either retains its IP address when it moves, or changes its IP address and updates its correspondents, respectively. Each

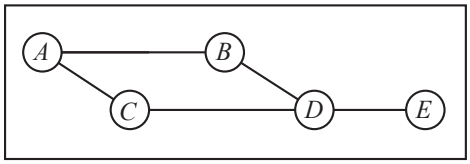

Figure 1: Members and links of a layer.

pattern has a completely different set of subsidiary design decisions and resource costs.

Third, we use the framework to describe and compare many of the most important proposals for mobility. Sections 4 and 5 compare different ways to implement dynamic-routing mobility, with a non-hierarchical name space (*e.g.,* MAC addresses in a local area network) and a hierarchical name space (*e.g.,* IP addresses in the wide area), respectively. Section 6 compares four prominent session-location mobility protocols at different stages in the IETF standardization process.

Fourth, Section 7 offers a systematic exploration of the design space of mobility, and discusses composition of mobility mechanisms. As an example, we illustrate how Mobile IPv6 is a composition of both the dynamic-routing mobility and session-location mobility design patterns.

Fifth, Section 8 briefly surveys several topics closely related to mobility, including multihoming, anycast services, site mobility, incremental deployment of mobility protocols, and security issues for mobility. The chapter ends with a brief conclusion outlining several more advanced areas of study.

2 The geomorphic view of networking

In the geomorphic view of networking, the architectural module is a *layer*. Each layer is a microcosm of networking—it has all of the basic ingredients of networking in some form. In a network architecture there are many layer instances; they appear at different levels, with different scopes, with different versions of the basic mechanisms, and for different purposes.

2.1 Components of a layer

A layer has *members*, each of which has a unique *name*. For example, Figure 1 is a snapshot of a layer with five members, each having a capital letter as a name. In general a member is a concurrent process, *i.e.,* a locus of state and control with the potential for autonomous action.

The members of a layer communicate with each other through *links*, shown by lines in Figure 1. A link is a communication channel. In general, a layer does not have a link between each pair of members.

One of the two primary functions of a layer is to enable members to send messages to each other. To do this, a layer needs *routes* indicating how one member can reach another through links and intermediate members. For example, (*A, B, D, E*) is a route from *A* to *E*. It also needs a *forwarding protocol* that runs in all members. The forwarding protocol enables members to send and receive messages. In addition, when a member receives a message on an incoming link that is not destined for itself, its forwarding protocol uses the route information to decide on which outgoing link or links it will forward the message.

A *channel* is an instance of a communication service. As mentioned above, a link is a channel. Sometimes a layer implements its own links internally. Most commonly, however, the links of a layer are implemented by other layers that this layer uses, placing the other layers lower in the *"uses" hierarchy*.

If an underlay (lower layer) is implementing a link for an overlay (higher layer), then the basic attributes of the channel must be stored in the states of both layers. In the overlay, the channel object is one of its *links*. In the underlay, the channel object is one of its *sessions*. There must be two names for the sets of channels of interest to a layer, because a typical layer both uses *links* and implements *sessions*.

The second primary function of a layer is to implement enriched communication services on top of its bare message transmission. Typical enrichments for point-to-point services include reliability, FIFO delivery, and quality-of-service guarantees. This function is carried out by a *session protocol*. A layer can implement sessions on behalf of its own members, as well as or instead of as a service to overlays.

For a link in an overlay to be implemented by a session in an underlay, both endpoint *machines* must have members in both layers, as shown in Figure 2. A *machine* is delimited by an operating system that provides fast, reliable communication between members of different layers on the machine. This fast, reliable operating-system communication is the foundation on which networked communication is built.[1]

A *registration* is a record that relates an overlay member to an underlay member on the same machine. Registrations must be stored as data in both layers. In the overlay they are called *attachments*, because they indicate how a member is attached to the network through a lower layer. In the underlay they are called *locations*, because they indicate that a member is the location of a process in a higher layer.

The session protocol creates and maintains *sessions* data in its layer, and uses *locations* data. For example, in Figure 2, *A* sent a request to *a* for a session with *E*. To create this session, *a* learned from its layer's *locations* that *E* is currently located at *e*. Messages sent from *A* to *E* through the link in the overlay

[1] Although layer members have been described as concurrent processes, they are not usually "processes" as defined by the operating system; processes in an operating system have many more properties and associations than layer members do. A virtual machine can be regarded as a *machine*, in which case communication through the hypervisor and soft switch of the physical machine is regarded as networked communication.

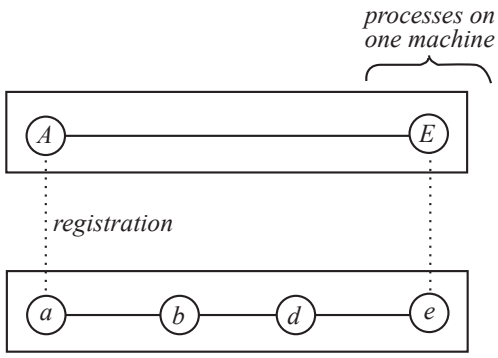

Figure 2: Implementation of a link in an overlay by a session in an underlay.

primary function	state component	maintenance algorithm
	members ←	member algorithm
	locations ←	location algorithm
session protocol →	sessions	
	attachments ←	attachment algorithm
forwarding protocol	links ←	link algorithm
	routes ←	routing algorithm

Figure 3: Major components of a layer. Arrows show which protocol or algorithm writes a state component.

travel through *a, b, d,* and *e*; the first and last steps uses operating-system communication, while the middle three steps use networked communication.

The six major components of the state of a layer are listed in Figure 3. All can be dynamic. We have seen that the session protocol creates and maintains *sessions*; the other five are created and maintained by their own maintenance algorithms.

2.2 Layers within a network architecture

The geomorphic view of networking was inspired by the work of Day [14], although we have made many changes and additions in both content and presentation. It may seem familiar and obvious because both the classic Internet architecture [15] and the OSI reference model [16] also describe network architecture as a hierarchy of layers, but in fact there are several radical differences, which the name "geomorphic" has been chosen to emphasize.

In the Internet and OSI architectures, each layer has a specialized function that is viewed as different from the function of the other layers. In both architectures there is a fixed number of layers, and the upper layers are global. In the geomorphic view the arrangement of layers is more varied and complex, as

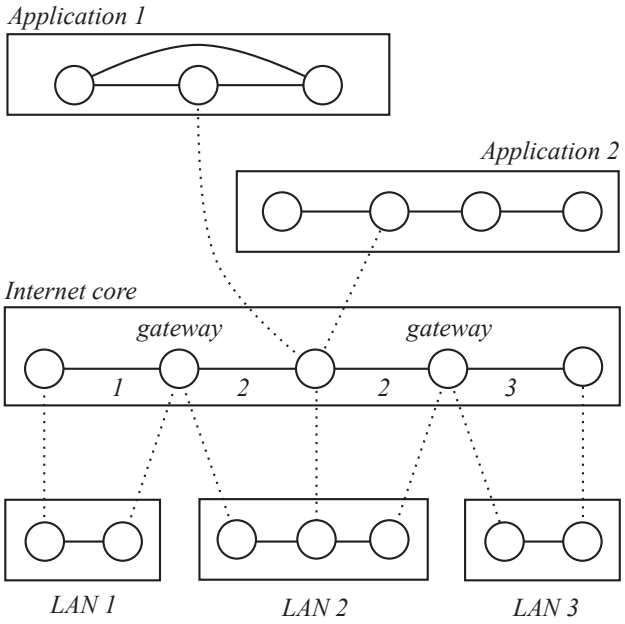

Figure 4: Geomorphic view of the classic Internet architecture. Internet links are labeled with the LAN that implements them.

it actually is in the earth's crust.

In the geomorphic view, each layer is viewed as the same in containing all the basic functions of networking, and there can be as many layers as needed. Consequently, the network (IP) and transport (TCP/UDP) layers of the classic Internet architecture fit into one "Internet core" layer of the geomorphic view (see Figure 4). In this layer, IP is the forwarding protocol and TCP and UDP are variants of the session protocol offering variants of Internet communication service.

The *scope* of a layer is its set of potential members. Some layers have local scopes, such as the LANs in Figure 4. There can be many such layers at the same level of the hierarchy.

Some layers have global, but specialized, scopes, such as the applications in Figure 4. Each application is a layer with its own members, name space, and communication services. These layers overlap geographically, while sharing the resources of the Internet core. The overlapping and abutting shapes in Figure 4 are common to both geological diagrams and networking.

Because layers instantiated at different levels have different purposes, their functions take different forms. For one example, the best-known routing algorithms are in the Internet core, where their purpose is reachability. A higher-level middleware layer might offer security as part of its communication services. Implementing security might entail routing all messages to a particular desti-

nation through a particular filtering server, so that, in this layer, part of the purpose of routing is security. An application layer might have a link or potential link between any two members, implemented by communication services below, so that in this layer the routing algorithm is vestigial.

For another example of a basic function with different forms in different layers, low-level layers such as Ethernet local area networks (LANs) provide broadcast as a communication service. The main services provided by the Internet core are point-to-point, while an application layer might implement its own multi-party communication service.

2.3 How we use the geomorphic view

Today's Internet is host to many customized architectures running simultaneously [17, 18]. Middleware is an important part of the ecosystem, while cloud services and virtual private networks add extra layers to the classic Internet architecture. It is self-evident that fixed layer structures cannot describe these architectures adequately. The geomorphic view is intended not only to describe them, but also to generate a design space including many others not yet explored.

It might be said that the problem with mobility is not too few proposals, but too many. As mentioned in the introduction, the total number is probably in the hundreds and growing.

In this chapter, the geomorphic view will provide a descriptive framework that imposes some order on this chaotic design space. We will show that each mobility proposal has a unique description in terms of the geomorphic view, that there are two general patterns that together explain all mobility mechanisms, and that each specific proposal is a special case of one of the patterns—or possibly a composition of the two.

This approach leads to many differences from other literature on mobility. As exemplified by [19], it is common for mobility proposals to be classified according to the layer of the classic Internet architecture where they are implemented. In contrast, we emphasize that each specific proposal is an instance of a general pattern, and that the general pattern can be used at any level of a network architecture. Comparisons between ideas are less subjective, because they are based on a common framework that exposes real similarities and differences, even when obscured by incidentals of language and application.

If nothing else, we hope that the geomorphic view will help people understand how each mobility proposal fits into "the big picture." Naturally, most network mechanisms are described by exception, for example, "when router R gets a message of type M, it treats it in the following [exceptional] way...." This is purposeful and concise, but it leaves a large number of questions unanswered, such as:

- How does R know that M should be treated as an exception?
- If M is also exceptional in format or protocol, does the sending end know how to generate it, and does the receiving end know how to interpret it?

- Does this exception interact or conflict with any other mechanism?

Because the geomorphic view provides a complete (although very simple) view of what networks do, it should help people know what they do not know, ask the right questions, and fit each proposal into a larger context.

3 Two patterns for implementing mobility

In networking, the term *mobility* refers to both a problem and its solutions. As a problem, it means that a layer member is changing its attachment to lower layers of the network, in particular while using communication services. As a solution or implementation, it means maintaining the member's communication channels despite the movement.

In this section we show that there are two completely different patterns for implementing mobility. They differ in where the change of attachment appears with respect to the implementing layer, in which algorithms and protocols of the implementing layer are involved in implementing mobility, and in which parts of the shared state are altered. They also differ in their detailed design decisions, and in their cost, performance, and scalability issues.

Not only are these patterns non-overlapping, they also completely cover all implementations of mobility, in the sense that each implementation either follows one pattern or is clearly a composition of the two patterns.

3.1 Dynamic-routing mobility

Figure 5 has two stages depicting the effect of mobility on an inter-layer channel. Recall that the channel is a *link* in the state of the layer that uses it, and a *session* in the state of the layer that implements it; its *higher endpoints* are members in the user layer, while its *lower endpoints* are members in the implementing layer.

The precise site of mobility here is the lower endpoint A. In Stage 1 A is *attached* to *a1* in LAN 1. Recall that *a1* is the *location* of A, and the association between them is a *registration*. *a1* and A are connected to the rest of their layers through Links 1 and 2, respectively. Link 2 is implemented by LAN 1, which might be an Ethernet or wireless subnetwork.

Between Stage 1 and Stage 2 Link 1 stops working, possibly because the machine on which A and *a1* reside has been unplugged from an Ethernet, or has moved out of range of a wireless subnetwork. In a cascading sequence of events, Link 1 is destroyed, Link 2 is destroyed, and the registration of A at *a1* is destroyed. A is now disconnected from the rest of its layer.

Eventually the mobile machine may become plugged into another Ethernet or enter the range of another wireless subnetwork, as shown in Stage 2. In a cascading sequence of events, member *a2* (which is the mobile machine's member in the new LAN 2) connects to the rest of its layer through Link 3, A becomes attached to new location *a2*, and new Link 4 is created in the mobility layer and implemented by LAN 2. Note that A is now linked to C rather than B; this change is necessary because C is attached to LAN 2 and B is not.

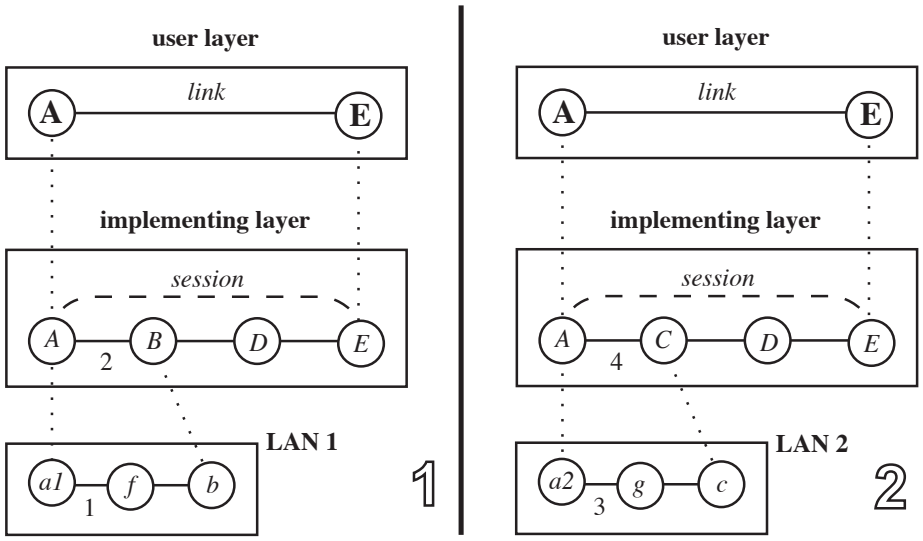

Figure 5: Two stages in an instance of dynamic-routing mobility.

Between Stages 1 and 2 there may be an interval during which A has no connection with the rest of its layer. The hard problem to be solved in Figure 5 is that even after A is again reachable by other members of its layer such as D and E, they do not know how to find it because the routes to it are obsolete. *Dynamic-routing mobility* relies on the routing algorithm of the layer, which must learn about new links, recompute routes, and disseminate new routes. After this is accomplished, D will know that it can reach A by forwarding to C.

There are three ways in which actual dynamic-routing mobility can differ from the example in Figure 5. Fortunately, none of them affect what the implementation has to do, so none of them need be discussed separately. First, the new attachment *a2* could be in the same layer as *a1*, rather than in a different layer. Because *a1* and *a2* are different locations, after the move A is probably linked to a different member of its own layer, even though the new link is implemented by the same lower layer as before.

Second, in Figure 5 the mobile member A has only one attachment and one necessary link. As shown in Figure 4, members such as gateways have multiple simultaneous attachments to different underlays. Because each such attachment is necessary for the gateway's purpose and supports its own link or links, the mobility of each attachment is a separate problem to be solved.

Third, occasionally a layer implements sessions for the benefit of its own members, rather than as a service to a higher user layer. In this case there is no **A** or **E**, and the beneficiaries of the mobility implementation are A and E.

A *router* is a member of a layer that receives and forward messages not destined for itself, whether it sends and receives messages on its own behalf or not. A *forwarding table* is a distributed copy of some of the *routes* state

component of a layer. Implementations of dynamic-routing mobility incur four kinds of resource cost:

- *storage cost* is the cost of storing routes to mobile members, in the forwarding tables of all the routers that need them;
- *update cost* is the cost of updating the stored routes as mobile members move;
- *path cost* is the cost of longer or more congested message paths due to mobility;
- *handoff latency* is the message delay caused by a move.

These costs will be discussed further in Section 3.3.

The primary issue in implementing dynamic-routing mobility (DRM) is that large layers such as the classic Internet core achieve scalability through a hierarchical name space. In the Internet core, names (IP addresses) are organized into a hierarchy based on geographical, topological, and administrative factors. A layer member is assigned a name based on its location in this hierarchy. Subtrees in the hierarchy correspond to blocks of names, and routing scales because it operates on aggregated blocks rather than individual names. Mobility violates the rules of this scheme, because a mobile member retains its name as it moves across the boundaries of the hierarchy. If implemented naively, it would require a large number of entries in the forwarding table of each IP router for individual mobile machines.

This issue is so important that the design decisions made to implement DRM are completely different in hierarchical and non-hierarchical layers. For that reason, we have divided examples of DRM into two sections (Sections 4 and 5).

3.2 Session-location mobility

Figure 6 is similar to Figure 5. One difference is that **A**'s location in the implementing layer changes from *A1* to *A2*, rather than staying the same. Another difference is that the LAN level is not shown. This is because the relevant change of attachment is now between the user layer and the implementing layer, not between the implementing layer and the LAN level, so what happens at the LAN level is irrelevant.[2]

This is a crucial difference from the perspective of the implementing layer, and requires a completely different mechanism for implementing mobility. The bulk of the work of implementing session-location mobility lies in ensuring that **A**'s correspondents know that it is now located at *A2* rather than *A1*. The distributed version of the *locations* mapping that is used for lookup must be updated. Each lower endpoint that was participating in a session with *A1* on

[2]Most often layer members *A1* and *a1* would be destroyed, and members *A2* and *a2* created. In this case there would be no mobility between the implementing level and LAN level because each of the *Aj* is attached to the same *aj* throughout its lifetime.

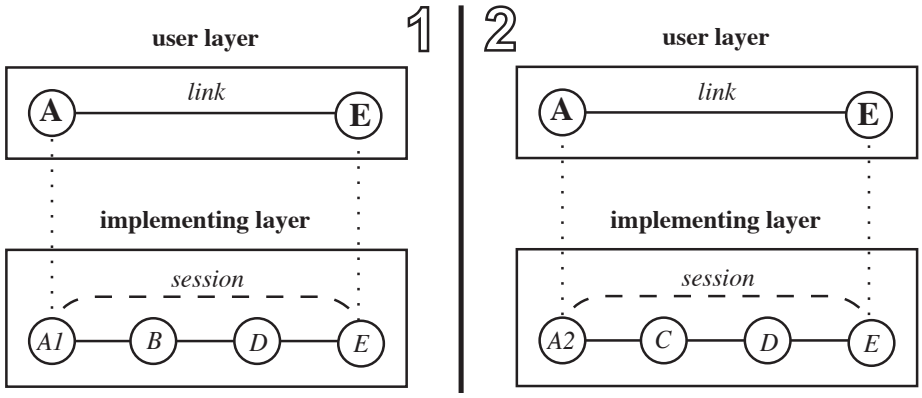

Figure 6: Two stages in an instance of session-location mobility.

behalf of **A** must be informed that it should now be corresponding with *A2* instead.

The change of registration from *A1* to *A2* should be familiar from observing what happens when a laptop containing an application layer member **A** moves to a new subnetwork of the Internet, and gets a new IP address from DHCP. From the perspective of the Internet, the laptop has died as member *A1* and become reborn as member *A2*. From the perspective of the laptop, its IP interface has changed its name from *A1* to *A2*.

It should be apparent that session-location mobility is a natural choice for implementing mobility in a hierarchical layer, because the lower endpoint of a session can change names when it moves with respect to the hierarchy. Strictly speaking some dynamic routing could be involved, because *A2* is a new member of the layer and there must be routes to it. In practice this is rarely an issue, because the name *A2* is part of some larger address block to which routes already exist.

Generally the fastest handoffs are achieved when a new lower endpoint sends updates directly to all its correspondent lower endpoints (in addition to updating the global *locations* mapping). This requires, of course, that the new lower endpoint have the correct name of the lower endpoint at the other end of each session.

Interesting behavior arises if both ends of a session move concurrently. Neither lower endpoint will know the new name of the far endpoint, so neither can send an update to the other. In this "double handoff" scenario a mobile endpoint, finding that it cannot reach a far endpoint to update it, will suspect that the far endpoint has moved also. Both endpoints must fall back on lookup from the *locations* mapping to get the new location of the far endpoint.

Like DRM, session-location mobility (SLM) has storage costs, update costs, and handoff latency. The storage costs are the costs of maintaining a scalable distributed implementation of *locations*. The update costs are the costs of

updating *locations* and current correspondents when a member moves.

Implementations of SLM vary in a number of ways (see Section 6), although no one variation is as important as the hierarchical *versus* non-hierarchical variation for DRM.

3.3 Major differences between the patterns

There are obvious structural differences between the two patterns:

- In DRM the change of attachment appears between the implementing layer and a lower layer, while in SLM the change of attachment appears between the user layer (higher) and the implementing layer (see Figures 5 and 6).
- In DRM the bulk of the work is performed by the routing algorithm, while in SLM the bulk of the work is performed by the session protocol and location algorithm (see Figure 3).

These structural differences prove that the two patterns are fundamentally different.

In attempting to understand mobility mechanisms, people are sometimes confused by the fact that *routes* (changed by DRM) and *locations* (changed by SLM) are both mappings. Furthermore, in the case that a mapping is global in its layer, it is often implemented by a shared data structure called a *directory*. The result is that directories are sometimes used in both DRM and SLM implementations.

This similarity is misleading because it does not tell us anything important about how the implementations fit into their layers or their overall network architectures. The mappings used in DRM and SLM are always fundamentally different, and can always be distinguished from one another. As mentioned in Section 3.1, *routes* is a peer-to-peer or intra-layer mapping: at each router, entries in the forwarding table map each destination name to a member, link, or path *in the same layer*. *Locations*, on the other hand, is always an inter-layer mapping, mapping names in a higher layer to names in a lower layer.

Obviously a quantitative comparison between two mobility implementations cannot be made without implementation details and a profile of the expected load. Nevertheless, it is possible to make some general comparisons between the two patterns based on their greatest strengths and weaknesses.

The greatest weakness of DRM is its storage, update, and path costs. Normally routing information is different in different places, so there is a lot of it, it is spread widely across a layer, and it is expensive to update. Attempts to economize on storage and update costs can lead to high path costs (see Section 5), as messages travel further to be routed successfully. Path costs must be weighted heavily because *every* message that travels on a channel is affected by its path cost, if any.

Locations are very different from routes because the result of a location query is usually the same no matter which member is querying (in contrast to a route,

which is different depending on where it is starting from), and because a location query is needed only at the beginning of a session and possibly after a move (in contrast to routes, which are consulted on every hop of every message). As a result, locations can be stored and updated much more cheaply than routes. For example, even a centralized directory would perform adequately in many contexts. And even if lookup of a location is slow, we do not count it as a path cost because the cost is incurred a few times for each channel rather than being built into the cost of transmitting each message on the channel.

The greatest weakness of SLM is that it must be implemented with the participation of session endpoints. This means that deployment of an SLM mechanism requires new or updated mobile devices. Interoperation with legacy endpoints calls for expensive middleboxes. Security is a serious concern because endpoint devices can initiate updates of the global layer state. Designing a protocol that correctly updates correspondents can be challenging, because it must deal with asynchronous concurrency, message loss, and out-of-order delivery. The alternative is to force all parties to refresh the locations mappings frequently, which is simpler but has higher overhead.

These concerns do not necessarily apply to DRM because a layer can, in principle, be designed so that its members are partitioned into endpoints and routers. Only the routers need be aware of or participate in an implementation of DRM.

4 Examples of dynamic-routing mobility in non-hierarchical layers

Dynamic-routing mobility is often used in LANs, which have smaller scopes and can function without hierarchical name spaces and aggregated routing.

4.1 Wired Ethernet LANs

An Ethernet LAN is a single layer. Its member processes are the Ethernet representatives of hosts (endpoints) and switches (routers), and its names are MAC addresses. It has no pre-attachment requirements or configuration for hosts, which makes it "plug and play."

The LAN offers both broadcast and point-to-point services to higher layers. In this brief section we do not consider these communication services further, so there will be no discussion of the layer's sessions or locations. Also, for simplicity, we will not extend the modeling into lower levels, so links in the Ethernet layer are primitives.

An Ethernet layer has two kinds of links. There are point-to-point links between switches, each of which is basically a wire between two machines. There are also shared media or buses. A bus delivers each message to every machine on the bus, and is used to connect a switch to a set of hosts. Either kind of link can be identified at each switch that uses it by the port on the switch's machine to which it is attached.

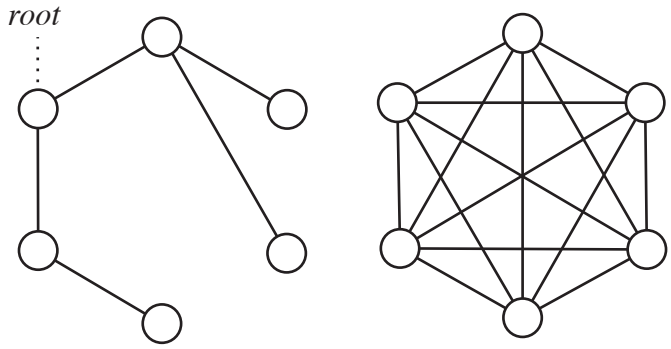

Figure 7: Inter-switch links of an Ethernet LAN layer (left) and an overlay layer (right). The Ethernet links are physical, while the overlay links are virtual.

The inter-switch links of the layer must form a bidirectional spanning tree (see Figure 7). Otherwise, when flooding is used (see below), the network could be overwhelmed by messages traveling on cycles. There are usually more physical links than needed for the spanning tree, but the extras can only be used when other links fail and the spanning tree is recomputed.

Each switch has a forwarding table containing *(MAC address, port)* pairs. The port identifies the link on which the switch should forward messages destined for the MAC address. Each switch's table is sparse and is populated lazily by a routing algorithm called "MAC learning." Upon receiving a message with a source MAC address that is not in its forwarding table, the switch adds to its table the MAC address and the link on which the message was received.

The forwarding algorithm of a switch is similarly simple. Upon receiving a message not destined for itself, the switch looks for the destination MAC address in its own forwarding table. If it finds an entry, it forwards the message on the designated link. If it does not find an entry, it "floods" by forwarding the message on every link except the one on which it was received.

These mechanisms implement dynamic-routing mobility as an aspect of normal operation rather than as a special case. When a host moves within the layer, it changes the link through which it is attached to the layer. As soon as it sends messages, new routes to it begin to propagate through the layer. Obsolete forwarding-table entries are removed when their time-to-live expires. Missing table entries are handled by flooding. Note that an entry might also be removed from a forwarding table because the table is full and space for a newer entry is needed.

4.2 Ethernet overlays

Several recent designs [4, 6, 7] avoid flooding by forming an overlay topology that interconnects all of the edge switches, as shown on the right side of Figure 7. While the inter-switch links of an Ethernet are physical and form a spanning

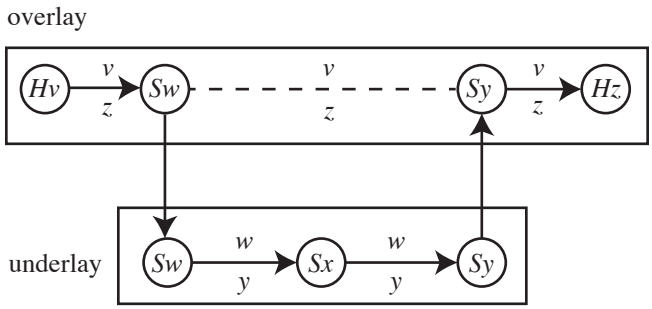

Figure 8: The path of a message through two layers and three switches.

tree, the inter-switch links of an overlay network are virtual and fully connect the switches.

The virtual links are communication services implemented by a second, lower layer. For example, Figure 8 shows the path of a message from host *Hv* to host *Hz* (the lower-case letters stand for their MAC addresses). On each hop, the path is labeled with the source name above and the destination name below. The virtual hop between switches *Sw* and *Sy* in the overlay layer is implemented in the underlay, where the message is encapsulated in a message with source w and destination y.

How are the virtual links in the overlay implemented by the underlay? The members of the underlay layer are the switches only, not the hosts. Each switch's name is the MAC address of its machine, just as in the overlay, so there is no need for a *locations* state component to map one name to another. The members of the underlay are stable and stationary. Routing is static except for failures, and the forwarding tables are fully populated. Because there is no flooding, there is no need to restrict the links to a spanning tree, and all of the physical links between switches can be fully utilized. The underlay can run an efficient routing protocol, such as a link-state protocol, to compute a shortest path from one edge switch to another.

Routing in the overlay is unusual compared to routing in general, because every edge switch is directly linked to every other edge switch. This means that an inter-switch route to a host can be identified simply by the MAC address of the host's edge switch, and is exactly the same no matter which switch needs the route! Thus inter-switch routing is a mapping that is *global* within the layer.

As with Ethernet LANs, each switch has a routing table that is populated lazily (*e.g.*, through MAC learning). The difference lies in what happens when a switch needs a route to an unknown destination. Rather than flooding, it looks the route up in a global routing directory.

When a host moves, the directory is updated with the new route to the host. The exact update mechanism differs from one overlay design to another,

depending on whether mobility is planned (*e.g.,* virtual-machine migration in a data center) or unplanned (*e.g.,* a mobile device moving within a campus). In a data center, a central controller that triggers virtual-machine migration can also update the directory with the new route [6, 7]. If the directory cannot be informed in advance that a host is moving, the new local switch can learn that a new device has connected and subsequently update the directory [4].

The routing directory in an overlay is an important part of its design, and may have many features to make both queries and updates fast and efficient. Different designs have different directory structures. VL2 [6] and NVP [7] run a centralized directory on a collection of server machines. In these designs, if the ingress switch *Sw* does not know the route for host *Hz*, *Sw* queries a directory server to learn the route *Sy*. In contrast, SEATTLE [4] implements the directory as a "one-hop Distributed Hash Table" [20] running directly on the switches. If the ingress switch *Sw* does not know the route for host *Hz*, *Sw* computes the hash of the *Hz*'s MAC address and forwards the message over a single overlay link to the switch responsible for this hashed value. This switch, in turn, forwards the message to *Hz*'s local switch *Sy* and informs switch *Sw* of the route for *Hz* so that future messages flow directly from *Sw* to *Sy*.

To improve the speed of mobile handoff, the ingress switch *Sw* can receive an update when a host moves to a new location. To perform these updates, the directory could maintain information about all ingress switches that recently queried the directory for a route to *Hz*. However, this can require the directory to maintain a large amount of state. Instead, when a host moves, the directory can update the mobile host's old local switch. Upon receiving a message for the mobile host, the old local switch can both forward the message to the mobile host and send an immediate notification about the new route to the sending switch [4]. This reactive invalidation of stale routes obviates the need for the directory to maintain information about which ingress switches sent queries for *Hz*, while still ensuring rapid invalidation of stale routes.

In addition to SEATTLE, VL2, and NVP, several other designs adopt certain aspects of the overlay solution. The early work on Rbridges [8], and the resulting TRILL [9] standard at the IETF, also forms an Ethernet overlay with shortest-path routing in the underlay. However, instead of having an explicit directory service, TRILL relies on flooding to reach hosts with unknown routes. Rather than flooding on all normal overlay links, TRILL floods on a special multicast link in the overlay. This special link is implemented in the underlay by a multicast tree formed on the underlay topology.

Like VL2 and NVP, the PortLand [5] design has a set of directory servers that allow ingress switches to learn the route to a destination host. Instead of encapsulating a message, PortLand assigns each edge switch a block of host "pseudo-MAC addresses" and rewrites the host MAC addresses at the edge. To enable the use of hierarchical pseudo-MAC addresses, PortLand is restricted to the tree topologies common in data-center networks. Table 1 summarizes the structural characteristics of all five designs.

Protocol	Routing Directory	Encapsulation
SEATTLE	one-hop DHT on the switches	simple encapsulation
VL2	directory servers	simple encapsulation
NVP	directory servers	simple encapsulation
Rbridges/TRILL	none, flooding on multicast tree	simple encapsulation
PortLand	directory servers	none, MAC rewriting

Table 1: Ethernet overlay designs for dynamic-routing mobility.

4.3 Comparative resource costs

Concerning storage costs, both Ethernet LANs and overlay designs incur the costs of the forwarding tables in switches. These costs are kept moderate by the fact that the tables are sparsely populated. Because there is no aggregation of names or table entries, the costs of densely populated tables would be too great. In addition to the forwarding tables, the overlay designs incur a storage cost for the routing directory, which maintains global state for the layer.

Concerning update costs, both approaches incur negligible costs for populating forwarding tables lazily though MAC learning. The biggest update cost is the cost of Ethernet flooding. The cost of flooding, in terms of bandwidth, grows quadratically with the size of the network—which makes it a potential scalability problem. Whether it becomes an actual problem or or not depends on its frequency, which depends on both the frequency of moves and the number of correspondents that a mobile host tends to have. SEATTLE, VL2, NVP, and PortLand have no flooding cost, though they do have the additional cost of updating the directory.

Mobility in the overlay designs incurs no path cost. The path cost of Ethernet mobility is significant, because the spanning tree (which is necessitated by flooding) forces paths to be longer and forces some physical links to go unused.

We can measure handoff latency from the instant when the mobile host re-attaches to the network and informs its local switch (before that time no mobility mechanism can take effect). The following scenarios assume that a correspondent switch is sending a steady stream of messages to a mobile host. They describe the elapse of time before messages sent by the correspondent switch (CS) are forwarded to the mobile host at its new attachment.

The Ethernet scenario:

1. The time-to-live of the CS's route to the mobile host expires, if it has not already.
2. CS receives the next message from the correspondent host and floods it.
3. After a round trip to the mobile host, CS learns the new route.

After Step 3, messages sent by CS are forwarded to the mobile host at its new attachment.

In the directory-based overlay solutions (*i.e.,* SEATTLE, VL2, NVP, and PortLand):

1. The directory receives an update of the mobile host's new local switch.
2. The directory informs the mobile host's old local switch of the new route.
3. The next message arrives at the mobile host's old local switch, and is forwarded on the new route.
4. The mobile host's old switch also informs the CS of the new route.

At Step 3 and after, messages sent by CS are forwarded to the mobile host at its new attachment. If Step 1 of the Ethernet scenario takes time, then the handoff latency of the overlay designs will be smaller than the Ethernet's.

In addition to resource costs, security and privacy are ever-present concerns. In Section 3.3 we noted that DRM usually has minimal security problems because only routers participate in routing. Ethernet flooding is an exception to this rule because it allows hosts to play a role in routing. Malicious hosts can force flooding by filling up the network's forwarding tables. (This would be accomplished by sending many messages from spoofed source MAC addresses.) Severe flooding can cause denial of service. Also, the malicious hosts will receive all the flooded packets, which may contain private information that they wish to see.

5 Examples of dynamic-routing mobility in hierarchical layers

Both of the designs in this section are intended for mobility within the Internet core. For this reason, both must grapple with the problem of a hierarchical name space as explained in Section 3.1. To reduce overhead, both solutions significantly limit the number of routers in the Internet that must store and update state concerning how to reach each mobile node.

5.1 Mobile IPv4

Mobile IPv4 [21, 22] drastically reduces storage and update costs by reducing the number of routers that must have a current route to a particular mobile host to one or two. Also, because each router is responsible for only a limited number of mobile hosts, no router is over-burdened by mobility.

Figure 9 shows the path of a message from correspondent host C to mobile host M in an Internet core layer with Mobile IPv4. Router *HA* is the *home agent* of M, and is supposed to have a route to it at all times. Router *FA* is the *foreign agent* of M, meaning that it is local to the subnetwork where M is now attached, and currently knows a route to M through the subnetwork.

The IP address M is in an aggregated routing block such that all messages destined for the block are routed to *HA*. Thus this router need only be the home

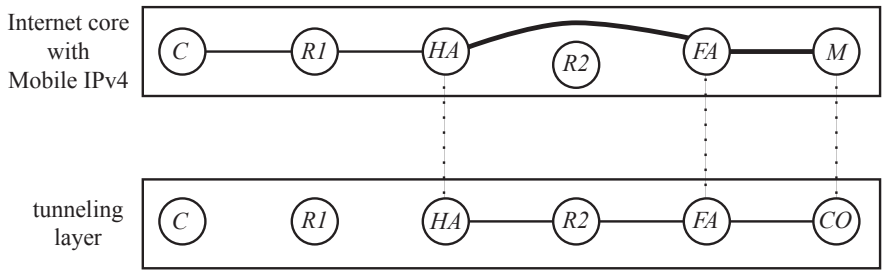

Figure 9: The path of a message to mobile host *M* with Mobile IPv4. Special links are drawn with heavier lines. Only the links employed in the path are shown.

agent for mobile hosts with IP addresses in its block. The message from *C* arrives at *HA* by means of normal IP routing through router *R1*. The subnetwork of *HA* is *M*'s home subnetwork, so when *M* is at home *HA* has a local route to it.

When *M* is not at home and becomes attached to the subnetwork of *FA*, it gets a local "care-of" IP address *CO* in that subnetwork. *M* informs *FA*, which informs *HA* that it is the current foreign agent of *M*. To forward messages to *M*, however, *HA* cannot merely forward them toward *FA*. It they were sent out on normal IP links, normal IP routing would send them back to *HA*! Messages to *M* from *HA* and *FA* must be forwarded on special links that are separate from normal IP links.

As shown in Figure 9, the special links in the Internet core are implemented by a tunneling layer below the core layer. The home agents, foreign agents, and mobile hosts of the Internet core are all registered at members of the tunneling layer. Home agents and foreign agents have the same names in both layers, while *M* is attached to member *CO* in the tunneling layer. To forward a message for *M* on its special link, *HA* in the core layer encapsulates the message in another message destined for *FA*, and passes the message to member *HA* in the tunneling layer.

Although the tunneling layer resembles the core layer (see below), its state differs from that of the core layer in several important respects:

- *Routes:* In the core layer, at *HA* messages for *M* are forwarded to *FA* on a special link, at *FA* messages for *M* are forwarded to *M* on a special link, and everywhere else messages for *M* are forwarded to *HA* on a normal link. In the tunneling layer *M* does not exist.

- *Attachments:* Some members of the core layer are attached to members of the tunneling layer.

- *Locations:* The core layer has no *locations* state, at least not related to Mobile IPv4. Although the tunneling layer need not maintain explicit *locations* state for mobile routers because they have the same names in

both layers, it must maintain explicit *locations* state for mobile hosts from the core layer. This state, which supplies the current local IP address of a mobile host, is stored in the foreign agent to which it is relevant.

In Mobile IPv4, mobile hosts such as M send messages to their correspondents such as C though normal IP links. This often creates problems because IP address M is not part of the normal routing block of the subnetwork at FA. If there is ingress filtering for security in or near this subnet, messages with a source address of M will be thrown away. In Section 7 we shall see how Mobile IPv6 eliminates this problem.

Overall this is an interesting architecture because the Internet core layer and the tunneling layer are mostly identical, and share the same implementation. Home agents, foreign agents, and mobile hosts are all aware of the differences between the layers and aware of their dual membership and dual roles. The shared implementation works because none of the other members of the layers need to be self-aware in that way. They always behave the same, without knowing that sometimes their actions contribute to the core layer, while other times their actions contribute to the tunneling layer.

By distinguishing clearly between the two layers, we make it possible to check the correctness of the software for each. It also becomes possible to make further distinctions if advantageous. For instance, implementation of a link between HA and $R2$ can be shared by both layers, but it might make sense to distinguish links in the two layers for reasons of performance or accounting.

5.2 MSM-IP

MSM-IP [23] is a proposal for using IP multicast to implement mobility. A mobile host gets an IP address M in the distinguished multicast block. When the mobile host attaches to a new subnetwork using local IP address L, L joins the multicast group for M, and the previous local address used by M resigns from the group.

With IP multicast there is a distinguished set of multicast routers, which are globally distributed and are responsible for routing messages destined for a multicast address to all members of the address's current multicast group. These routers exchange information and forward messages to each other through special links, exactly as the routers participating in Mobile IP do. The special links are implemented by a tunneling layer, exactly as the special links in Mobile IP are.

With MSM-IP, every subnetwork that supports either mobile hosts or their correspondents must have a multicast router. Messages to mobile hosts (or true multicast groups) are recognized by their distinguished addresses and sent to their local multicast router, where they enter the special multicast routing system.

5.3 Comparative resource costs

The costs of dynamic-routing mobility depend greatly on the number of routers that must have a current route to each mobile host. More routers incur more storage and update costs. Storage and update costs are much greater for MSM-IP than for Mobile IPv4, because an entire network of multicast routers must be updated on each move.

Using fewer routers, on the other hand, incurs more path cost. With MSM-IP path cost is minimal, because a message travels from the multicast router in the sender's subnetwork, along an optimal path through the distributed multicast routers, to the multicast router in the receiver's subnetwork. With Mobile IPv4 path cost can be high, because each message to a mobile host must pass through the home agent, regardless of where the sender is and where the mobile host is. This problem of path cost or "triangular routing" is the reason why the designers of Mobile IPv4 decided to send messages from mobile hosts through normal IP links. They incur no path cost, but they do run afoul of security filtering.[3]

In Section 3.3 we said that dynamic-routing mobility does not in principle require participation of the endpoints. Mobile hosts in Mobile IPv4 and MSM-IP do not have this advantage. The reason that they must have special behavior is that both designs use special routing mechanisms, separate from normal IP routing, to find mobile hosts. Because the routing mechanism is special, it is not necessary to update every IP router when a mobile host moves. But also, because the routing mechanism is special, mobile hosts must also behave differently to interact with it in the correct way.

6 Examples of session-location mobility

In this section we compare four IETF standards for session-location mobility: the Host Identity Protocol (HIP) [24, 25], the Identifier-Locator Network Protocol (ILNP) [26], the Locator/Identifier Separation Protocol (LISP) Mobile Node [27], and the "route optimization" mechanism of Mobile IPv6 [28, 29]. Section 7.3 will explain how route optimization fits into Mobile IPv6 overall.

These standards have many similarities, as they all provide mobility by splitting the Internet core layer (shown in Figure 4) into two layers. These two layers are shown in Figure 10, and also correspond to the two layers in Figure 6. Mobility supports the persistence of an inter-layer channel that is a link in the upper layer. This is called the *identifier layer* because each Internet host has a persistent name in it called its *identifier*. The channel is implemented by a session with SLM mobility in the lower layer. In the lower layer, called the *locator layer*, each IP interface of each host has a *locator* which is an IP address.[4]

[3] Messages from MSM-IP mobile hosts do not have problems with security filtering because multicast IP addresses are recognizable as belonging to a special category.

[4] Note that the members of the identifier layer are *hosts*, while the members of the locator layer are *interfaces*. This distinction can safely be ignored in this section, but it is important for multihoming as discussed in Section 8.1.

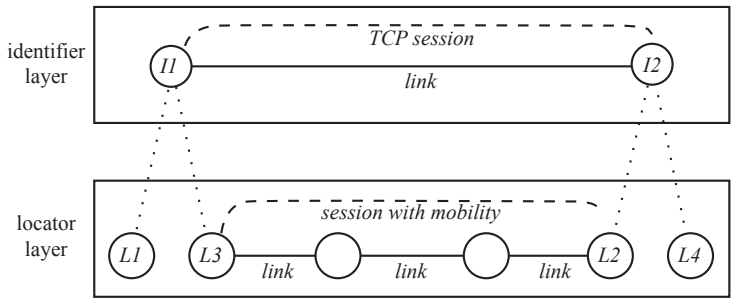

Figure 10: The Internet core layer splits into two layers for session-location mobility. In the identifier layer, session protocols such as TCP run largely unmodified. In the locator layer, the session protocol implements mobility.

Protocol	Identifier	Locator
HIP	(hash of) public key	IPv4 or IPv6 address
ILNP	64-bit IPv6 suffix	IPv6 address
LISP Mobile Node	IPv4/IPv6 address (called EID)	IPv4/IPv6 address (called RLoc)
Mobile IPv6	IPv6 address	IPv6 address

Table 2: Comparison of SLM standards on the basis of names.

Figure 10 is a geomorphic approximation of the real implementations of these standards, in which the split between layers is implicit and incomplete. In the geomorphic view, two separate session protocols are employed. In the identifier layer, a largely unmodified TCP implementation provides the usual TCP service as if identifiers were IP addresses. (UDP and other service protocols operate here as well.) In the locator layer, the only purpose of the session protocol is to implement SLM.

6.1 Names

Table 2 compares the four standards on their choices of names. They differ most on identifiers, which must be globally unique and persistent, but have no other necessary constraints.

HIP places a great emphasis on building in security, so the identifier of a host is the host's public cryptographic key. With the use of keys as identifiers, messages can have self-authenticating source information. Self-authentication provides security within the SLM locator update protocol (see Section 6.3), while the guaranteed presence of a public key makes it easy to protect the channel data with encryption. Identifiers can also be hashes of public keys, which allows for shorter identifiers without sacrificing self-authentication.

ILNP can only be used in the context of IPv6. Its identifiers are 64 bits long; a host usually chooses a unique identifier for itself by taking the 48-bit

MAC address of one of its hardware interfaces and using a standard algorithm to extend it to 64 bits.

The significance of 64 bits is that in IPv6 routing based on hierarchical names and aggregation, the longest possible prefix is 64 bits. This means that at its finest-grained, IPv6 routing examines the first 64 bits of an IPv6 address and points to a subnetwork. The IPv6 address still has a 64-bit suffix to map to an IPv6 interface attached to the subnetwork. In ILNP, identifiers are carried in the 64-bit suffixes of IPv6 addresses. In other words, an ILNP locator is derived from an ILNP identifier by prefixing 64 bits that indicate a subnetwork where the identified host can be found.

This scheme is very efficient in its use of address bits. It places constraints on the locator layer that are certainly satisfiable, but cannot be taken for granted, and may not be achievable in legacy situations. First, when some interface of a host attaches to a subnetwork, it must be able to choose its own IP suffix (local address) within the subnetwork, and it must be free to choose its own identifier as that suffix. Second, consider what happens when a host with two MAC addresses *m1* and *m2* chooses *m1* as the basis of its ILNP identifier. It may then attach to a subnetwork through the interface associated with *m2*. Despite the use of *m2* to access the subnetwork, the subnetwork must allow it to choose an IP suffix based on *m1*.

LISP Mobile Node and Mobile IPv6 are less interesting. In both cases, identifiers are normal routable IP addresses.

Naming choices have the biggest effect on the deployment opportunities of a design. Mobile IPv6 and ILNP require the deployment of IPv6. HIP and ILNP require more changes to TCP because their identifiers are not normal IP addresses. Deployment of new protocols is usually incremental, which means that updated hosts and subnetworks must interoperate with legacy hosts and subnetworks. Often a legacy components cannot know whether a name it receives is an identifier or a locator. This gives a big advantage to LISP Mobile Node and Mobile IPv6, which use the same name space for both. Interoperation is a complex subject, and will not be considered further here.

6.2 Directories

An implementation of SLM requires a globally accessible implementation of *locations* in the locator layer, mapping identifiers to locators. Table 3 compares the four standards on their choices of location directory or other mapping implementation.

LISP Mobile Node inherits its directory mechanism from LISP [30], which is an IETF standard designed for a different purpose (multihoming of large-scale enterprise subnetworks), and not originally intended for the support of mobility. The directory mechanism is a special-purpose distributed subsystem of directory servers. While this requires a substantial initial investment, it does give the deployer maximum freedom. For example, different deployments could use almost any name space as the set of identifiers.

Protocol	Location Directory
HIP	DNS
ILNP	DNS
LISP Mobile Node	LISP subsystem
Mobile IPv6	home agent for each host has its locator

Table 3: Comparison of SLM standards on the basis of directories.

Both HIP and ILNP make use of the Domain Name System (DNS) as a scalable, highly available directory subsystem. In both cases it is necessary for a mobile host to have a domain name, which serves as the lookup key for *both* the host's identifier and locator, in two separate lookups. This is necessary because a domain name is hierarchical in structure, while the identifiers used in HIP and ILNP are not. DNS relies on the hierarchical structure of domain names both for scalability of lookups and to manage the distributed administration of DNS servers. Although an entity wishing to communicate with a mobile host would often know it only by a domain name anyway, the need for an extraneous third name that is neither an identifier nor a locator violates the principle of separation of concerns.

For both HIP and ILNP it is necessary to add new record types to those stored by DNS servers, because the value being looked up is not always an IP address. Finally, the DNS server with the authoritative copy of a locator must send it out with a time-to-live of zero. Otherwise other DNS servers will cache the information, impeding responsiveness to changes of location.

The route optimization (SLM) mechanism of Mobile IPv6 is an adjunct to the Mobile IPv6 DRM implementation (see Sections 5.1 and 7.3). Because the DRM implementation uses home agents, the SLM implementation uses them also. The current locator of an identifier can always be obtained from its home agent. Mobile IPv6 may be less reliable than other designs because a home agents is a single point of failure with respect to its mobile hosts. Home agents do not necessarily have the built-in redundancy and high availability that the directories of the other designs have.

6.3 Locator update protocols

An implementation of SLM must have a protocol through which mobile nodes update the directory and their correspondents after a move. The protocol must have security to prevent updates from unauthorized hosts.

It would take far too much space to report on how each standard meets these requirements. Also, many standards provide a menu of implementation alternatives, some of them better-documented than others. In lieu of this detail, we will merely touch on a few of the design issues for SLM protocols.

Even without the problem of double handoff (as introduced in Section 3.2), an update protocol can suffer from lost or re-ordered messages. If a correspondent node or directory receives two different update messages from a mobile

Protocol	Encapsulation
HIP	encapsulation with IPSec Encapsulating Security Payload
ILNP	none, identifier is extracted from locator
LISP Mobile Node	simple encapsulation
Mobile IPv6 (RFC 3775)	simple encapsulation
Mobile IPv6 (RFC 6275)	Home Address destination option and Type 2 Header

Table 4: Comparison of SLM standards on the basis of encapsulation.

host in the wrong order, it could retain an obsolete locator for the mobile node. If a correspondent node or directory determines from sequence numbers that an update message has been lost, it might wait forever for a retransmission that will not come because the mobile node is somewhere else and will not receive the retransmission request. These bugs have been discovered in real SLM protocols [31]. In general, the two techniques to rely on are (1) version numbers rather than sequence numbers, and (2) some form of protocol verification to insure against otherwise-almost-inevitable mistakes.

If an endpoint loses track of the session's other endpoint because of double handoff, loss of update messages, or a protocol bug, it can always get the current locator by making a new lookup in the directory. In general, an SLM protocol can be made more robust by having mobile nodes report their locators to the directory frequently, and having correspondent nodes refresh their cached locators from the directory frequently. This robustness comes at the cost of increased overhead in the form of message traffic.

HIP uses a different method to solve the problem of double handoff. When a mobile host moves, its old locator is adopted by a "rendezvous server" that keeps track of its new locator. The rendezvous server intercepts control messages destined for the old locator, and forwards them to the new locator. Even when both endpoints of a session move at the same time, their update messages will reach each other through rendezvous servers. As always, data messages travel directly between hosts.

6.4 Encapsulation

Like Table 1, Table 4 compares the standards on the basis of how they encapsulate overlay messages as they travel through an underlay. Note that there are two versions of the Mobile IPv6 standard which differ in this respect (the newer [29] supersedes the older [28], so this comparison is of academic interest only).

Consider a message being sent from left to right in Figure 10, at a time when identifier *I1* has locator *L3* and identifier *I2* has locator *L2*. In the simplest implementation, the message consists of a message with source *I1* and destination *I2*, encapsulated in a message with source *L3* and destination *L2*. There are other possibilities, however, motivated by the desire to conserve space in message headers. This is a serious concern in IPv6, where each of the four address fields is 128 bits long.

LISP Mobile Node and the original version of Mobile IPv6 use simple encapsulation as above. HIP does also, with the additional proviso that the message body containing the identifiers is protected with IPsec. In ILNP each identifier is a suffix of its current locator, so it need not be sent separately.

In the revised Mobile IPv6 standard, there is an optimization apparently based on the observation that, most of the time, only one of the endpoints of a session will be mobile. For a stationary node, the identifier and locator are always the same (see Section 7.3), and need not be sent twice. So a message from the mobile node needs a source identifier and not a destination identifier, and a message to a mobile node needs a destination identifier and not a source identifier.

Now let us assume that *I1* is mobile and *I2* is stationary. For messages from *I2* to *I1*, a destination identifier is needed. The revised Mobile IPv6 standard uses a special Type 2 header. This is a kind of "source routing" header, allowing the source to provide a list of destination addresses through which the message must be routed. The messages have destination list (*L3; I1*), where the second hop from *L3* to *I1* is internal to the mobile host. For messages from *I1* to *I2*, a source identifier is needed. The extra source address *I1* is inserted using a special "Home Address destination option." This option is an extension to IPv6 allowing an extra field in a message header. In this way, the extra identifiers can be added to messages only when needed.

One might speculate that such a complex optimization would cause trouble in the form of further, cascading complexities, and this is indeed the case. There are elaborate rules in [29] for processing messages so that IPsec works correctly: each message must be processed partially with *I1* in the ordinary source or destination field, and partially with *I1* moved to the Type 2 header or Home Address destination option field. Note that this is an interaction that has been explicitly recognized and accommodated in the standard. No one knows how many problematic interactions with other protocols, caused by this optimization, will be discovered if Mobile IPv6 comes into widespread use.

The SLM implementation of Mobile IPv6 is constrained by the need to compose with the DRM implementation of Mobile IPv6 (see Section 7.3). If there were not so many constraints, the size of headers could be reduced without violating the principle of separation of concerns. For example, Figure 11 revises Figure 10 in an obvious way as suggested by the geomorphic view. The session protocol in the locator layer now performs both TCP and SLM functions.

In the identifier layer, the link between *I1* and *I2* is uniquely identified at one end by *I1, p1*, where *p1* is a port number, and uniquely identified at the other end by *I2, p2*. Note that the identifier layer is more like an application layer than an IP layer in that it has no forwarding—only direct links between pairs of communicating endpoints. Consequently, once the link has been set up, there is absolutely no need to transmit *I1* and *I2* in data messages. Each endpoint simply uses the port number to pass messages unambiguously across the layer boundary. Figure 11 has many similarities with TCP Migrate [10] and Serval [11, 31], which are other proposals for SLM.

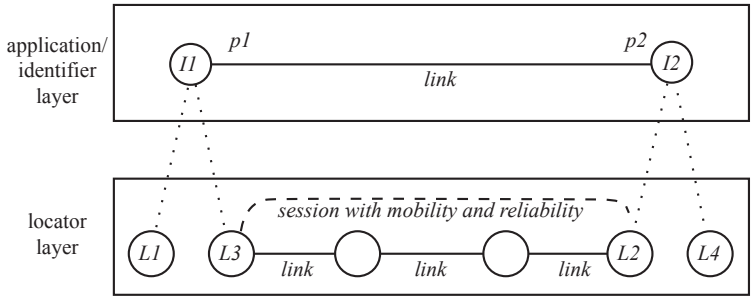

Figure 11: A more efficient version of Figure 10.

7 Composition of the patterns

7.1 Structural modeling

The geomorphic view of networking is rigorously defined and can be formalized. We have formalized various aspects of the geomorphic view in Alloy, which is the modeling language of the Alloy Analyzer [32], and in Promela, which is the modeling language of the Spin model checker [33]. These are *structural models*.

Networking researchers and practitioners are accustomed to *analytical models*, which are also formal, but quantitative rather than structural. Once can assign numbers to some symbols in an analytical model, give the numbers and model to a suitable evaluator, and receive numbers for other symbols in the model.

Structural models are similar, except that evaluation is logical rather than quantitative. We assume that some formulas in a model are true, give the assumptions and the model to a suitable evaluator such as the Alloy Analyzer or Spin, and receive information about the truth of other formulas. We either learn that a formula is true, or get a counterexample showing why it is not true.

Sections 3 through 6 should have made clear why the term we use to describe these models is *structural*. We use them to describe hardware and software structures within networks, and to compare mechanisms based on where their structures are similar and different. We also use them to show how structural decisions constrain other decisions and affect important properties such as scalability and interoperability. In the next subsection, we will mention some additional knowledge gained with the help of structural modeling.

7.2 Generating the design space of mobility

An instance of mobility is an isolated episode in which one layer member changes its attachment from one underlay member to another. One of our goals is to give network architects the freedom to handle any instance of mobility with any mobility pattern at any level of the layer hierarchy. This should enhance efficiency

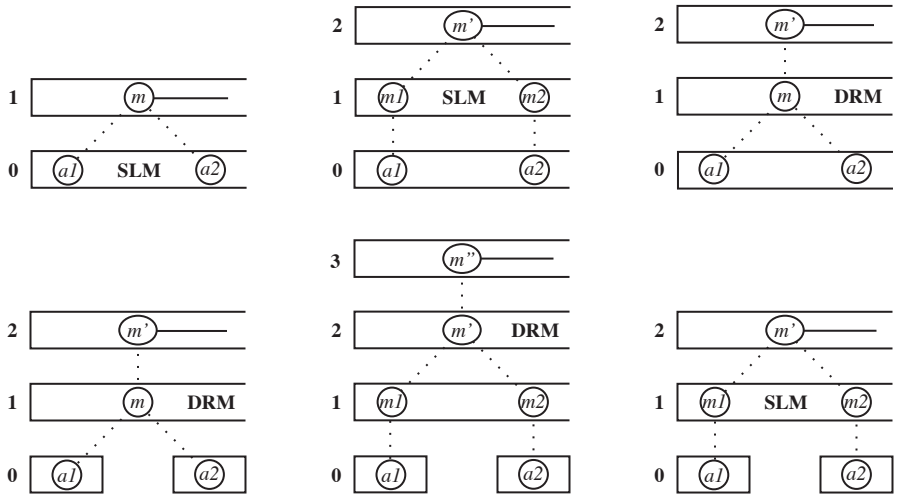

Figure 12: Generating the design space.

and scalability by allowing solutions that are finely tuned to the characteristics of the problem they are solving.

The first step was to identify the two possible implementation patterns and to provide sufficiently abstract versions of them (Section 3). The next step, taken in this section, is to show that any instance of mobility can be implemented with either pattern at almost any level of the layer hierarchy.

In the left column of Figure 12, top half, we see a fundamental instance of mobility in which the old and new locations are in the same layer at Level 0. As notated, the channel at Level 1 can be preserved by session-location mobility (SLM) at Level 0. In the left column, bottom half, we see a fundamental instance of mobility in which the old and new locations are in different layers at Level 0. As notated, a channel at Level 2 can be preserved by dynamic routing mobility (DRM) at Level 1.

The middle column of the figure shows the effects of a "lifting" transformation in which each mobility implementation is moved up a level in the hierarchy. The purpose is to show that mobility can be implemented in many different places, if the current architecture allows it or the designer has control of the content and design of relevant layers. In each case member *m* at Level 1 is replaced by two members *m1* and *m2*. Neither *m1* nor *m2* is mobile, as each has a stationary registration in Level 0 throughout its lifetime. Now member *m'* at Level 2 is mobile. As shown in the figure (top), a channel in Level 2 with *m'* as its higher endpoint can be preserved by SLM at Level 1. Or, as shown at the bottom, a channel in Level 3 with *m"* as its higher endpoint and *m'* as its lower endpoint can be preserved by DRM at Level 2.

The right column of the figure shows where one implementation pattern can be replaced by the other. To replace SLM by DRM (top right), it is necessary

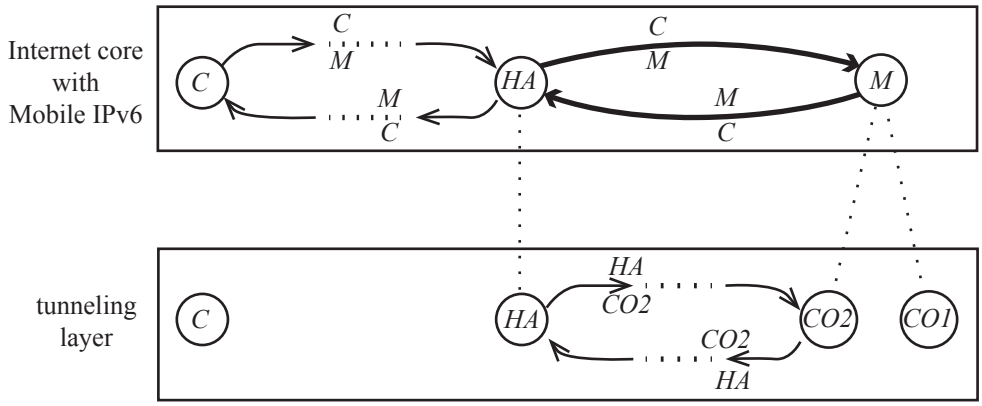

Figure 13: The paths of messages to and from mobile host M with the dynamic-routing mobility mechanism of Mobile IPv6. Special links are drawn with heavier lines. Only the links employed in the path are shown.

to lift the channel up one level. To replace DRM by SLM (bottom right), the channel can stay at the same level, but the mobility must be lifted up a level.

The final step of this exercise is to show that implementations of the two patterns can be freely composed, in the same layer or different layers of the same hierarchy. This result is verified with the Alloy Analyzer in [34], and establishes that network architects can exploit the entire design space of mobility without worrying about interactions or incompatibilities. The most important limitation is that, to benefit from proven compositionality, implementations must maintain the minimal separation of concerns inherent in the geomorphic view.

7.3 Composition in Mobile IPv6

As we saw in Section 5, Mobile IPv4 is an instance of DRM. Mobile IPv6 [28, 29] uses a similar DRM mechanism, and also composes it with the SLM mechanism described in Section 6. We first consider the DRM mechanism.

Figure 13 is the Mobile IPv6 version of Figure 9. Note that Mobile IPv6 has no foreign agents, as their functions are performed by the mobile hosts themselves. In the figure, both the old attachment of M at *CO1* and its new attachment at *CO2* are shown. Normal routers are not shown, being replaced by ellipses in the paths consisting of normal links.

Figure 13 also differs from Figure 9 in showing the source and destination addresses of the messages on every link (source on top, destination below). Thus a message in the core layer with source C and destination M is forwarded on a special link from *HA* to *M*. The implementation of this special link in the tunneling layer encapsulates the message in another message with source *HA* and destination *CO2*, and sends it through normal IP links and routers.

Figure 13 also differs from Figure 9 in showing the paths of return messages

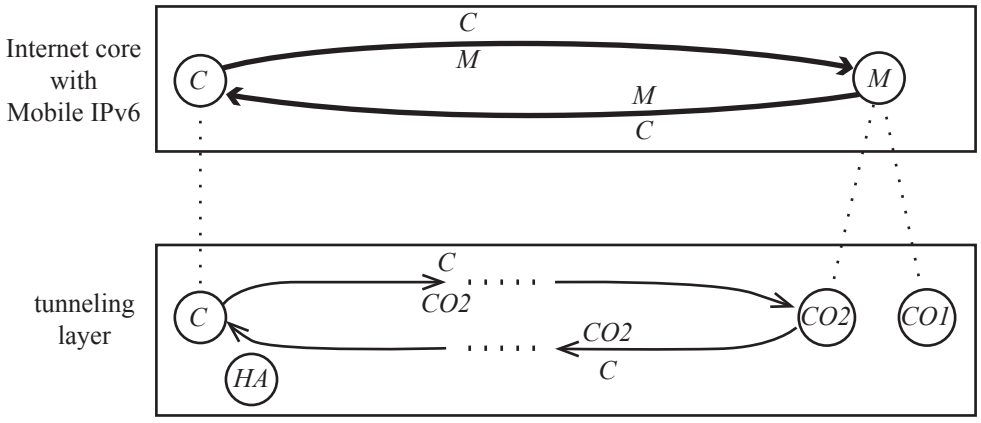

Figure 14: The paths of messages to and from mobile host M with the session-location mobility mechanism of Mobile IPv6. Special links are drawn with heavier lines.

from M to C. In contrast to MIPv4, return messages from M travel on a special link as far as HA. At HA they enter the realm of normal links and routers. There is no problem with ingress filtering because M belongs to the address block of HA's subnetwork, so M is a normal source address at that location.

The big problem with Mobile IP is the path cost of routing every message through a mobile host's home agent. Path cost is even worse in Mobile IPv6 than in Mobile IPv4, because it is incurred by messages *from* a mobile host as well as messages *to* it. To reduce this problem, Mobile IPv6 standardizes a version of SLM called "route optimization," as already presented in Section 6. SLM is used only after the session between C and M is established, and only if both endpoints have the protocol capability.

Figure 14 shows the SLM mechanism of Mobile IPv6 in the same context as its DRM mechanism in Figure 13. For simplicity, this figure uses simple encapsulation as in [28]. After the endpoints have exchanged messages to set up SLM, they send messages on special links implemented in the tunneling layer. Note that these are *different* special links than those used by DRM (Figure 13). The DRM special links involve HA and will change when M moves. The SLM special links do not involve HA, and will not change from the perspective of the core layer when M moves. With SLM, the only role played by HA in the tunneling layer is to store the directory entry for M. It is not needed originally because when SLM begins the two endpoints are already connected and know each other's locations in the tunneling layer, but it may be needed in case of double handoff.

Figure 14 also shows the SLM mechanism of Mobile IPv6 in the same context as Figure 10. The "Internet core with Mobile IPv6" layer in Figure 14 is the same as the identifier layer in Figure 10. The tunneling layer in Figure 14 is the

same as the locator layer in Figure 10.

So far our discussion of the Internet core/identifier layer has been limited to its links. We have seen that *C* and *M* may have a choice of links over which to send their messages. Because the network path associated with each link is different, its performance may be different. It is the job of TCP in the Internet core/identifier layer to smooth over any difficulties caused by diverse paths, for example by ensuring that messages are delivered to an application layer in FIFO order.

The design space generated in Section 7.2 is intended primarily for compositions in which different instances of mobility are managed by different mechanisms or at different levels of the architecture. Composition in Mobile IPv6 is a little different because the exact same instance of mobility—pictured in the figures as *M*'s change of attachment from *CO1* to *CO2*—is being handled simultaneously by both forms of mobility. The DRM implementation is in the core/identifier layer, while the SLM implementation is in the tunneling/locator layer. DRM is the default mechanism, because SLM can only be used if both endpoints are SLM-enabled.

8 Design considerations related to mobility

Mobility mechanisms lie close to the heart of networking, so they are related to many other communication services and aspects of networking. In this section, we touch briefly on several other topics closely related to mobility.

8.1 Multihoming

Increasingly, mobile devices connect to the Internet via multiple interfaces (*e.g.,* a laptop with WiFi and wired Ethernet interfaces, a smartphone with WiFi and cellular interfaces, or a virtual machine running on a physical server with multiple wired Ethernet interfaces). Since these interfaces usually connect to different administrative domains (*e.g.,* a campus WiFi network and a commercial cellular provider), they must have different IP addresses in different address blocks. There is no name for the Internet host itself, and no way to route to the host itself rather than a specific one of its interfaces.

Sequential multihoming is the use by a host of multiple interfaces, one after the other, during the lifetime of a channel. A mobility mechanism can be used to implement sequential multihoming; in principle either of the mobility patterns can be used to provide it.

If we look at the specific real mobility protocols in Sections 4 through 6, however, we see that the DRM protocols in Sections 4 and 5 work with a single IP address per host, while the SLM protocols in 6 work with multiple IP addresses per host. This is an artifact of where these protocols sit in the IP stack rather than an inherent property of the implementation patterns, but it does mean that SLM implementations are currently a better match for multihoming than DRM implementations.

Simultaneous multihoming allows a host to use its multiple interfaces to contribute bandwidth to the same channel simultaneously. Simultaneous multihoming is closely related to sequential multihoming and therefore to mobility, but not strictly the same because the multiple interfaces of a host might not be allowed to *change* during the lifetime of a channel. Of course, what we really want is simultaneous *and* sequential multihoming, which is a little more complex than sequential multihoming because it requires protocol extensions so that a layer member can have and use more than one attachment at a time. All of HIP, ILNP, LISP Mobile Node, and Serval already have session protocols capable of simultaneous multihoming.

Although we can easily adapt a mobility solution to provide multihoming, it is more difficult to go the other way. Multipath TCP [35] and the Stream Control Transmission Protocol (SCTP) [36] are both proposals for multihoming, which can be used for either increased bandwidth or failure recovery. A communication channel formed with these protocols may have multiple IP addresses at each endpoint, but there is no higher-level host identifier that can survive the loss of network connectivity through those addresses. Without such an identifier, there is no way to preserve the channel until network connectivity is restored through other IP addresses.

8.2 Anycast

Increasingly, the Internet is a platform for users to access services hosted on multiple servers in multiple locations. The appropriate network abstraction for their requirements is *anycast*, in which a service has an *anycast name* that corresponds to a group of servers offering the service. A request for a communication channel to the service can result in a channel to any member of the group of servers.

For simple query-response services like DNS, all server replicas can share a single locator (*i.e.,* an IP address), and rely on IP routing to direct client requests to one of the server replicas. However, IP routing does not guarantee that multiple messages sent to the same IP address would reach the same server replica. In today's Internet, a domain name for a geo-replicated service can map to multiple IP addresses, one for each server replica. This supports communication services with multiple message exchanges between client and server. It is different from mobility, however, because the higher-level domain name has nothing to do with the channel after the initial DNS lookup.

Alternatively, anycast could be combined with SLM, as it is in Serval [11, 31]. A Serval identifier is an anycast name, and is registered as located at a dynamic group of servers in the locator layer. When a request for a channel to an identifier is handled by the Serval session protocol, the protocol selects the locator of some member of the group. In contrast to the previous paragraph, the identifier remains the name of the channel's higher endpoint throughout its lifetime. The SLM session protocol can thus maintain the channel through both mobility events and changes to the membership of the server group.

8.3 Subnetwork mobility

Mobility proposals typically focus on the movement of a single mobile endpoint, like a mobile device or a virtual machine. However, in some scenarios a subnetwork serving multiple endpoints can move from one location to another. For example, a fast-moving bus, train, or plane may carry a LAN that provides network connectivity to a large collection of passengers.

Dynamic-routing mobility in a hierarchical layer naturally handles subnetwork mobility by updating the routes used to reach the entire aggregated block of names. For example, Boeing had an early in-flight WiFi service that provided seamless mobility for airline passengers by associating each international flight with an IP address block, and announcing the block into the global routing system at different locations as the plane moved [2]. However, this solution required all interdomain routers in the Internet to store and update fine-grained routing information, leading to high overhead.

In our recent work [34], we have shown that subnetwork mobility is merely the mobility of the gateway's attachment to the larger network, and is implemented with the same two patterns as mobility of endpoints. We identified several applications of the design patterns that seem promising for handling combinations of subnetwork mobility and endpoint mobility. These solutions have the property that the mechanism for subnetwork mobility (a bus moves its access point from one roadside LAN to another) is completely independent of the mechanism for endpoint mobility (a user with a laptop gets on and off the bus). Nevertheless, it is not yet clear which solutions for subnetwork mobility would be most viable in practice.

8.4 Incremental deployment and interoperation

Deploying new protocols that span administrative domains is always challenging, since the Internet is a federated infrastructure and cannot easily have a "flag day" on which everyone upgrades to new protocols. Most real deployments are DRM mechanisms that operate within a single administrative domain (*e.g.,* cellular networks, Ethernet LANs, or data-center networks), or require support only from the mobile endpoint and a small number of routers (*e.g.,* Mobile IPv4).

It is not surprising that most real mobility implementations use DRM, because SLM entails many more deployment hurdles. There can be a new set of identifiers, a new global directory service, changes to both endpoints, and even changes to the service interface that are visible to applications. The early SLM protocol TCP Migrate [10] is probably the most deployable, requiring only changes to the operating system at the participating endpoints, but even so it has not had significant deployment.

Nevertheless, as noted in the introduction, the pressure for better network mobility support is mounting. Ubiquitous computing may be a particularly powerful motivator, because an enormous number of sensors and actuators will require network access. This could accelerate the adoption of IPv6, enabling

many other changes in its wake.

For incremental deployment, an SLM-enabled host can interoperate with a legacy host through a proxy or other middlebox. This raises many new questions concerning how a middlebox is introduced into the path between the hosts, and on the scalability of stateful middleboxes. These new questions must be added to the perennial list of old interoperability questions, such as how to traverse NAT boxes. Identifying effective ways to deploy these protocols incrementally remains an active area of research and standards work.

8.5 Security

All mobility solutions raise important questions about security. In DRM, who is authorized to announce routing changes for an address or address block? In SLM, who is authorized to update the directory service and a mobile endpoint's correspondents? Answering these questions successfully requires unforgeable notions of identity, and secure protocols for sending update messages to routers, directory servers, and endpoints. Can accidental misconfigurations or malicious attacks overload the routers, directory servers, or endpoints? Preventing denial-of-service attacks requires effective ways to limit the work performed before recognizing that messages are unauthorized.

As mentioned in Section 3.3, SLM protocols face greater security challenges, because arbitrary endpoints can initiate updates to global layer state. SLM protocols that use DNS as the directory service can use a secure update protocol for updating DNS records [37]. Some protocols, like HIP and Serval, embed an endpoint's public key (or a hash of the key) in the identifier, as a way to bootstrap secure communication and secure updates to both the directory servers and correspondents. Still, security is a rich and important topic warranting a much deeper treatment, especially since new protocols can easily introduce unforeseen vulnerabilities and new threats.

9 Conclusion

In this chapter we have presented an abstract framework for describing, understanding, and comparing approaches to network mobility. As illustrations, we have covered several mobility protocols in some detail. We believe the geomorphic model provides a clear and precise way to understand the considerable similarities between different mobility proposals, allowing discussions to focus on their meaningful distinctions rather than artificial differences in terminology.

We have compared mobility proposals on both qualitative (deployment constraints, security) and quantitative (resource costs, latency) criteria. The basis for making comparisons has been completely *structural*, in the sense of *structural modeling* as defined in Section 7.1. This is important because structural comparisons are vastly easier to obtain than comparisons based on simulation, and should always be the first step in any evaluation project.

In the interest of brevity, our structural discussions of quantitative criteria have been high-level and mainly suggestive. A true understanding of metrics such as storage cost, update cost, and path cost requires a more detailed investigation of the proposals, including supporting technologies such as routing protocols and directory services. Scalability depends on how these costs grow as the size of a network grows within the expected range.

Equally important, different mobility mechanisms can be composed. Even today it would not be surprising to see dynamic-routing mobility used within an administrative domain, while session-location mobility is used simultaneously across administrative boundaries. The ultimate goal would be to compose performance models along with the mechanisms they are modeling, so that the performance of a composed solution could be derived from the performance of its components.

In addition, most existing mobility protocols operate at fairly low levels in a network architecture, specifically the link, network, and transport levels of the classic Internet stack. Yet the essence of mobility is simply a dynamic binding of more abstract names to more concrete names. As such, mobility can be implemented in middleware as a service to even higher-level application layers. We believe that this is a fruitful avenue for further exploration, particularly because it might be easier to optimize narrowly-targeted implementations of mobility.

More generally, we believe that the geomorphic model embodies important design principles, such as separation of concerns and the value of design patterns and code reuse. While widely known to software engineers, these concepts are not as familiar to students of networking. We believe that a deeper understanding of these concepts would be valuable for networking researchers and practitioners alike, far beyond the treatment of any one subject like network mobility.

References

[1] Z. Zhu, R. Wakikawa, and L. Zhang, "A survey of mobility support in the Internet," IETF Request for Comments 6301, July 2011.

[2] B. Abarbanel, "Implementing global network mobility using BGP," NANOG Presentation, http://www.nanog.org/meetings/nanog31/abstracts.php?pt=NTk1Jm5hbm9nMzE=&nm=nanog31, May 2004.

[3] V. Jalaparti, M. Caesar, S. Lee, J. Pang, and K. van der Merwe, "SMOG: A cloud platform for seamless wide area migration of networked games," in *Proceedings of ACM/IEEE NetGames*, November 2012.

[4] C. Kim, M. Caesar, and J. Rexford, "Floodless in SEATTLE: A scalable Ethernet architecture for large enterprises," in *Proceedings of ACM SIGCOMM*, 2008.

[5] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *Proceedings of ACM SIGCOMM*, August 2009.

[6] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *Proceedings of ACM SIGCOMM*, 2009.

[7] Nicira, "It's time to virtualize the network," http://nicira.com/en/network-virtualization-platform, 2012.

[8] R. Perlman, "Rbridges: Transparent routing," in *Proceedings of IEEE INFOCOM*, 2004.

[9] J. Touch and R. Perlman, "Transparent Interconnection of Lots of Links (TRILL): Problem and applicability statement," IETF Request For Comments 5556, May 2009.

[10] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proceedings of MOBICOM*, 2000.

[11] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Ko, J. Rexford, and M. J. Freedman, "Serval: An end-host stack for service-centric networking," in *Proceedings of the 9th Symposium on Networked Systems Design and Implementation*, April 2012.

[12] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proceedings of ACM SIGCOMM*. ACM, August 2002.

[13] H. Schulzrinne and E. Wedlund, "Application-layer mobility using SIP," *Mobile Computing and Communications Review*, vol. 4, no. 3, pp. 47–57, July 2000.

[14] J. Day, *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2008.

[15] D. D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proceedings of SIGCOMM*, August 1988.

[16] ITU, "Information Technology—Open Systems Interconnection—Basic Reference Model: The basic model," ITU-T Recommendation X.200, 1994.

[17] T. Roscoe, "The end of Internet architecture," in *Proceedings of the 5th Workshop on Hot Topics in Networks*, 2006.

[18] O. Spatscheck, "Layers of success," *IEEE Internet Computing*, vol. 17, no. 1, pp. 3–6, 2013.

[19] I. F. Akyildiz, J. Xie, and S. Mohanty, "A survey of mobility management in next-generation all-IP-based wireless systems," *IEEE Wireless Communications*, vol. 11, no. 4, pp. 16–28, August 2004.

[20] A. Gupta, B. Liskov, and R. Rodrigues, "One hop lookups for peer-to-peer overlays," in *HotOS*, Lihue, HI, May 2003.

[21] C. E. Perkins, "Mobile IP," *IEEE Communications*, May 1997.

[22] ——, "IP mobility support for IPv4," IETF Network Working Group Request for Comments 3344, 2002.

[23] J. Mysore and V. Bharghavan, "A new multicasting-based architecture for Internet host mobility," in *Proceedings of the 3rd Annual ACM/IEEE International conference on Mobile Computing and Networking*, 1997.

[24] R. Moskovitz and P. Nikander, "Host identity protocol HIP architecture," IETF Network Working Group Request for Comments 4423, 2006.

[25] P. Nikander, A. Gurtov, and T. R. Henderson, "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks," *IEEE Communications Surveys and Tutorials*, vol. 12, no. 2, pp. 186–204, April 2010.

[26] R. J. Atkinson and S. N. Bhatti, "ILNP architectural description," This document appears as a sequence of IETF Internet Drafts beginning in 2012, 2013.

[27] D. Farinacci, D. Lewis, D. Meyer, and C. White, "LISP mobile node," IETF Internet Draft draft-meyer-lisp-mn-08, work in progress, October 2012.

[28] D. Johnson, C. Perkins, and J. Arkko, "Mobility support in IPv6," IETF Request for Comments 3775, June 2004.

[29] C. Perkins, D. Johnson, and J. Arkko, "Mobility support in IPv6," IETF Request for Comments 6275, July 2011.

[30] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "The locator/ID separation protocol (LISP)," IETF Request for Comments 6830, January 2013.

[31] M. Arye, E. Nordstrom, R. Kiefer, J. Rexford, and M. J. Freedman, "A formally-verified migration protocol for mobile, multi-homed hosts," in *Proceedings of the International Conference on Network Protocols*, October/November 2012.

[32] D. Jackson, *Software Abstractions: Logic, Language, and Analysis*. MIT Press, 2006, 2012.

[33] G. J. Holzmann, *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2004.

[34] P. Zave and J. Rexford, "Compositional network mobility," in *Proceedings of the 5th Working Conference on Verified Software: Theories, Tools, and Experiments*. Springer-Verlag LNCS to appear, May 2013.

[35] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," IETF Request For Comments 6824, January 2013.

[36] R. Stewart, "Stream Control Transmission Protocol," IETF Network Working Group Request for Comments 4960, September 2007.

[37] D. Eastlake, "Secure domain name system dynamic update," IETF Network Working Group Request for Comments 2137, April 1997.