The compositional architecture of the Internet

Pamela Zave
Princeton University
Princeton, New Jersey
pamela@pamelazave.com

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

ABSTRACT

Contrary to the "classic" Internet architecture familiar to most people, today's Internet is a composition of a wide variety of networks. The IP protocol suite offers a general-purpose network design with a widely available implementation; as such, it is re-used to design and implement networks with many different purposes. Compositional architecture explains how, despite the fact that IP has not changed significantly since 1993, the Internet has evolved to meet many new requirements and challenges since then. In this paper we introduce a new and principled model for describing Internet architecture, and give many examples of its validity. We also explain how the model can help us facilitate innovation through re-use of successful solution patterns, verification of trustworthy network services, and research on the architecture of a better Internet.

ACM Reference format:
Pamela Zave and Jennifer Rexford. 2018. The compositional architecture of the Internet. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 8 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

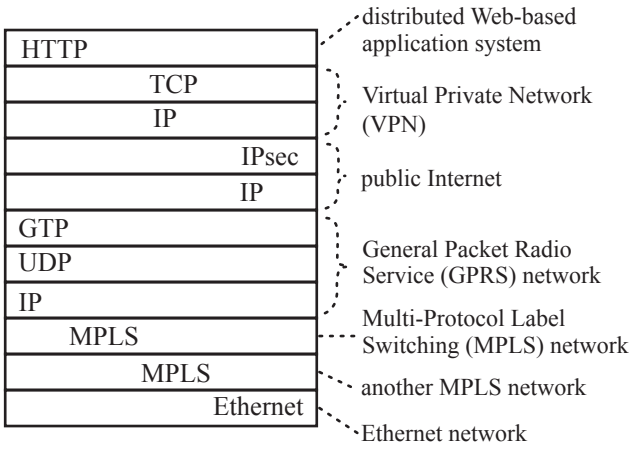

Figure 1: Headers of a typical packet in the AT&T backbone network. Headers lower in the diagram are outermost in the actual packet.

1 INTRODUCTION

In 1992, the explosive growth of the World Wide Web began. The architecture of the Internet was commonly described as having four layers above the physical media, each providing a distinct function: a "link" layer providing local packet delivery over heterogeneous physical networks, a "network" layer providing best-effort global packet delivery across autonomous networks all using the Internet Protocol (IP), a "transport" layer providing communication services such as reliable byte streams (TCP) and datagram service (UDP), and an "application" layer. In 1993 the last major change was made to this "classic" Internet architecture [11]; since then the scale and economics of the Internet have precluded further changes to IP [12].

A lot has happened in the world since 1993. The overwhelming success of the Internet has created many new uses and challenges that were not anticipated by its original architecture:

- Today, most networked devices are mobile.
- There has been an explosion of security threats.
- Most of the world's telecommunication infrastructure and entertainment distribution has moved to the Internet.
- Cloud computing was invented to help enterprises manage the massive computing resources they now need.
- The IPv4 32-bit address space has been exhausted, but IPv6 has not yet taken over the bulk of Internet traffic.
- In a deregulated, competitive world, network providers control costs by allocating resources dynamically, rather than provisioning networks with static resources for peak loads.

Here is a conundrum. The Internet is meeting these new challenges fairly well, yet neither the IP protocol suite nor the way that experts describe the Internet has changed significantly since 1993. Figure 1 shows the headers of a typical packet in the AT&T backbone [19], giving us clear evidence that the challenges have been met by mechanisms well outside the limits of the classic Internet architecture. In the classic description, the only headers between HTTP and Ethernet would be one TCP header and one IP header.

In this paper we will present a new way of describing the Internet, better attuned to the realities of networking today, and to meeting the challenges of the future. Its central idea is that the architecture of the Internet is a flexible *composition* of many networks—not just the networks acknowledged in the classic Internet architecture, but many other networks both above and below the public Internet in a hierarchy of abstraction. For example, the headers in Figure 1 indicate that the packet is being transmitted through six networks below the application system. Our model emphasizes the interfaces between composed networks, while offering an abstract view of

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

network internals, so we are not reduced to grappling with masses of unstructured detail. In addition, we will show that understanding network composition is particularly important for three reasons:

Re-use of solution patterns: In the new model, each composable network is a microcosm of networking, with the potential to have all the basic mechanisms of networking such as a namespace, routing, a forwarding protocol, session protocols, and directories. Our experience with the model shows that this perspective illuminates solution patterns for problems that occur in many different contexts, so that the patterns (and their implementations!) can be re-used. This is a key insight of Day's seminal book *Patterns in Network Architecture* [7]. By showing that interesting networking mechanisms can be found at higher levels of abstraction, the new model helps to bridge the artificial and unproductive divide between networking and distributed systems [17].

Verification of trustworthy services: Practically every issue of *CACM* contains a warning about the risks of rapidly increasing automation, because software systems are too complex for people to understand or control, and too complex to make reliable. Networks are a central part of the growth of automation, and there will be increasing pressure to define requirements on communication services and to verify that they are satisfied [14]. As we will show in this paper, the properties of trustworthy services are defined at the interfaces between networks, and are usually dependent on the interaction of multiple networks. This means they cannot be verified without a formal framework for network composition.

Evolution toward a better Internet: In response to the weaknesses of the current Internet, many researchers have investigated "future Internet architectures" based on new technology and "clean slate" approaches (e.g., [2, 20, 21, 25]). These architectures are not compatible enough to merge into one network design. Even if they were, it is debatable whether they could satisfy the demands for specialized services and localized cost/performance trade-offs that have already created so much complexity. A study of compositional principles and compositional reasoning might be the key to finding the simplest Internet architecture that can satisfy extremely diverse requirements.

In the next two sections of the paper, we begin with principles of the classic architecture, then discuss why they have become less useful and how they can be replaced. This should help clarify that we are proposing a really new and different way of talking about networks, despite the familiarity of the terms and examples. In the final section, we consider potential benefits of the new model.

2 THE USER INTERFACE TO A NETWORK

2.1 The end-to-end principle

The best-known principle of the classic Internet architecture is the end-to-end principle [5, 18], which creates a sharp divide between the network and the endpoint machines that it serves. The principle says that the functions of the network should be minimized, so that it serves everyone efficiently, and that whenever possible services should be implemented in the endpoint machines. The endpoints are easily programmable (so that anyone can add services), and the end-to-end perspective is the best perspective for functions such as reliability.

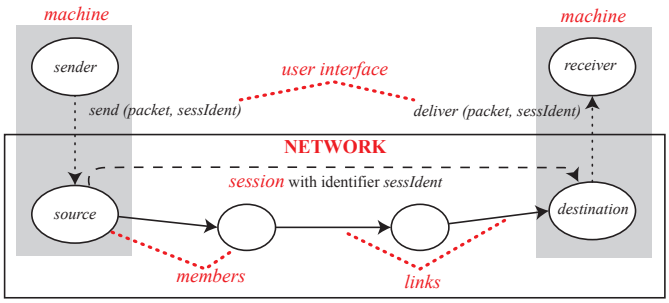

Figure 2: The main user interface to a network, with an example session.

The end-to-end principle is also expressed by the slogan "smart edge, dumb network." Another implication of the end-to-end principle is that the user interface to a network consists of the links between endpoint machines and the rest of the network.

Despite its tremendous explanatory and engineering value, the end-to-end principle does not describes the Internet as a whole. We know there are services (such as protection from denial-of-service attacks) that cannot be implemented in endpoints [4]. Today's Internet is full of *middleboxes*, which are functional elements located inside the network and inserted into end-to-end paths. On the other side of the divide, performance of the network depends to some extent on congestion control in TCP endpoints. It is no longer true that endpoint machines and networks are always owned by different parties (in clouds they are not), and no longer true that network elements such as routers are not programmable [10].

From a modeling perspective, the divide between network and endpoints is harmful for a very simple reason: *If we want to describe and verify communication (network) services, then we must include all the agents involved in providing those services.*

2.2 User interfaces are inside machines

Figure 2 illustrates the new model's approach to network services and the user interface. Each machine participating in a network must be running a *member* of that network. The network member is a software or hardware module that implements some subset of the network protocols. Members are connected by *links*, where a *link* is a communication channel that accepts packets from one member and delivers them to another member.[1] Members of the network forward packets that are not destined for them, so that a packet can reach its destination through a *path* of members.

The *users* of networks are distributed application systems—computer systems with operational modules spread across different physical machines. The modules of a distributed system need a network to communicate. The main *user interface* to a network consists of the interfaces inside machines between user modules and members of the network.

An instance or usage of network service is a *session*. A network has *packets*, which are its transmissible units of data. A session transmits a set of packets that are related from the perspective of

[1] Although the model allows one-to-many sessions and links for services such as broadcast and multicast, they are omitted for simplicity.

the user. In Figure 2, a one-way session transmits packets from an application *sender* to an application *receiver*. The session has identifier *sessIdent*. So the main user interface to the network is that the sender has action *send (packet, sessIdent)* to send a packet in the session, and the network has action *deliver (packet, sessIdent)* to deliver a packet to the receiver.

Although the user interface between two networks is always implemented inside machines, implementations vary. Many user interfaces are implemented by software in operating systems. The user interfaces to the MPLS networks in Figure 1, on the other hand, are implemented deep inside the hardware of high-speed routers.

3 THE NATURE OF A LAYER

3.1 Fixed layers with distinct functions

The classic Internet architecture prescribes five layers (including the physical media), as listed at the beginning of §1. The contemporaneous OSI reference model [13] has seven layers, with "session" and "presentation" layers between the transport and application layers. In both hierarchies each layer has a distinct function not performed by any other layer.

Fixed layers with distinct functions are no longer a realistic description of the Internet. For example, routing and forwarding are extremely important network functions; in the classic architecture the local version of these functions resides in the link layer, while the global version resides in the network layer. Yet Figure 1 also shows the presence of a GPRS (a standard for cellular data service) network and two MPLS networks, each of which has its own routing and forwarding that aggregates packets and manages resources at different levels of abstraction. Further up in Figure 1 we see three IP headers, plus evidence that three separate IP session protocols (TCP, IPsec, and UDP) apply to this packet.

Conceivably there is a model with fixed layers and distinct functions that fits this packet, but the same HTTP message—if observed at different places along its end-to-end path—will be encapsulated in packets with different headers indicating different layers and different functions. So no variation on the classic Internet architecture or OSI reference model can help us understand what is going on.

3.2 Self-contained networks

A major principle of the new model is that *the layers in a composition hierarchy are self-contained networks.* Each network is a microcosm of networking, with all the basic mechanisms including a namespace, routing, a forwarding protocol, session protocols, and directories. However, because networks vary widely in their purposes, geographical spans, memberships, and levels of abstraction, these mechanisms also vary, and a mechanism may be vestigial in a particular network design where it is not needed. According to this principle the IP protocol suite is a general-purpose network design that is implemented on most networked devices. As such, it can be re-used for the design and implementation of many networks. Note that an IP network encompasses both the network and transport layers of the classic Internet architecture.

We will now give brief explanations of the major parts of a network, followed by examples. Directories will be covered in §4.2.

A network's *namespace* is the set of *names* that its members can have. Most commonly each member of a network has a unique name, although there are many exceptions.

Routing is the mechanism that determines paths and installs entries in the forwarding tables of network members, while a network's *forwarding protocol* is the mechanism in which a member uses its forwarding table and other computations to forward packets toward their destinations. It includes formats for packet headers and forwarding tables. Most commonly, the table at each member is a mapping from *headerPattern* and *inLink* to *outLink*, where *headerPattern* matches some subset of packet headers, and *inLink* and *outLink* are local identifiers for the links of that member. The mapping tells the member that on receiving a packet on incoming link *inLink* whose header matches *headerPattern*, it should forward the packet onto outgoing link *outLink*. The mapping can also tell the member, explicitly or implicitly, to drop the packet.

A *session protocol* is a set of conventions governing a specific kind of session; it always includes the behavior of the session endpoint members, and may include the behavior of other network members on the session path. It covers packet headers, packet sequence, member state, and member actions. The header format of a session protocol is a specialization of its network's forwarding format, so a header must conform to both. The new model makes particular use of the following header fields:

- the name of the *destination* endpoint;
- a *session protocol identifier*;
- a *session identifier*;
- a *user network* to identify the network being served by the session (see §4.1).

These fields are always present in headers unless they are vestigial (which means that they would be identifying elements in a set of size zero or one) or unless the information they carry is already stored in members along the path of the session.

3.3 Examples of new networks

Many campus architectures have networks called Virtual Local Area Networks (VLANs) that are not found in the classic architecture [22]. The purpose of VLANs is to maintain an important network topology that is not present in either the IP network or Local Area Networks (LANs) on campus, as shown in Figure 3. In the figure each physical machine is assigned a color and final name digit for its network members, so that it is easy to see which network members are on the same machine.

At the bottom level we see that there are physical LANs covering different areas of campus, and some high-speed physical links across campus.[2] At the top level the campus has a private IP network. User machines are divided into groups depending on whether they belong to students, administrators, departments, etc. Members of a group are identifiable by the prefixes of their IP addresses (abbreviated in the figure). Within each group each user machine is connected by a virtual link to every other group member and to one or more IP routers that serve as security gateways to the group. Members of different groups can reach each other only through IP routers, where filtering rules are installed to allow only approved

[2]The "campus IP network" at the bottom level is a tricky part of the architecture, and will be explained in §6.

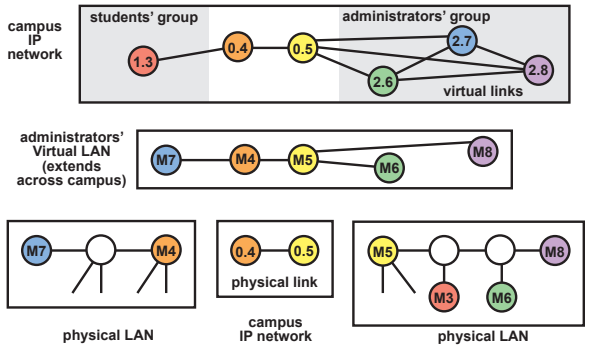

Figure 3: The architecture of a campus network. In this picture, all lines between members are bidirectional pairs of links.

communication among groups. Note that the machines with IP addresses 2.7 and 2.8 are close together in the IP topology, but far apart in the physical topology.

At the middle level of the figure there is an isolated VLAN for each group. Like the LANs, a VLAN uses the Ethernet design in which names are MAC addresses (abbreviated "M"). These VLANs do their own routing, separate from the routing in the LANs. A virtual link in the IP network must be implemented by a path in a VLAN and a link in a VLAN must be implemented by a physical path (see §4.1). As a result, a packet from 1.3 to 2.6 must go through IP router 0.5 and be screened, even though the shortest physical path between the red and green machines does not go through an IP router. The VLAN architecture has been found to simplify administration, enhance security, and improve the efficiency of campus networks [22].

A completely different kind of virtual network is often found in multi-tenant clouds, which may offer to their tenants various services such as load-balancing, packet filtering by firewalls, and application-specific performance enhancements. Such clouds have virtual networks that implement these services by inserting middleboxes into the paths of sessions. In these virtual networks, the major purpose of routing and forwarding is to direct the packets of sessions through middleboxes according to the tenant's service specification [3, 16].

The most unusual networks in this paper are Named Data Networks (NDN) [25]. In NDN each piece of data has a unique name. For purposes of the networking functions of routing and forwarding, a data server has the name of every piece of data available from it; a server can have many names, and a name can be assigned to many servers. The routing protocol uses advertising and other conventional techniques so that a request for data is usually forwarded to the nearest server with the requested data. In NDN, a session consists of a single request and its response, and there is no source name in the request packet. (A source name would be useless for returning the response to the requestor, because users don't have names and server names are not unique.) Rather, the session protocol leaves traces of the session in every member that forwards the request, so that the response can follow the path in reverse.

NDN is a "future Internet architecture" as mentioned in §1. In current NDN deployments, wherever NDN links must traverse non-NDN nodes, they are implemented by being layered on top of the public Internet. NDN networks are particularly interesting because their design shows how the session protocol, routing, and forwarding of a network can be highly specialized and tightly integrated.

4 COMPOSITION BY LAYERING

4.1 Layering of self-contained networks

The most important operator for composition of networks is *layering*, which is simply what happens when one network uses the services of another network, in exactly the sense of §2.2. More specifically, a link in a user network is implemented by a session in a used network.

A *usage hierarchy* is a directed acyclic graph whose nodes are networks and whose edges represent composition by link implementation. A *level* in this graph is a set of networks that all have the same graph distance from some reference point. This definition will be refined further in §5 and §6.

For example, Figure 3 is derived from a usage hierarchy, with the levels of the graph being represented by vertical placement. The bidirectional link between 2.7 and 2.8 in the campus IP network is implemented by a bidirectional session in the administrators' VLAN that follows the path shown between M7 and M8. The link between M7 and M4 in the VLAN is implemented by a session in the left physical LAN following the spanning-tree path between M7 and M4. Note that machines have distinct members in VLANs and LANs, even though those networks happen to use the same Ethernet design and the same namespace.

Consider the right LAN in Figure 3. It links machines in both the students' and administrators' groups, so it must implement links in at least two VLANs. When the destination of a session in this LAN receives a packet, which member of which VLAN should it deliver the packet to? LAN packets in this architecture have a user-network identifier called a "VLAN tag," which tells the destination which user network is being served by the session.

The shift from a principle of fixed layers to a principle of many self-contained networks encourages a shift in thinking and terminology—from different concepts and terminology for each layer to concepts and terminology that emphasize the similarities among layered networks. Most importantly of all, users of networks—*distributed application systems* in §2.2—can be *networks* themselves, and the distinction between the two concepts weakens.

If the service provided by a session protocol has a specification, then the specified properties of a session are also the guaranteed properties of a link that the session implements. For example, the best-known service of IP networks is implemented by the session protocol TCP. A user of TCP sends a stream of bytes, and this byte stream must be received by the user at the other end of the session with no bytes missing or duplicated, and all in the same order in which they were sent.

In the case of TCP the work needed to satisfy this specification is performed by the protocol implementation in the network members at the endpoint machines. IP/TCP packet headers have a *session identifier* (the four-tuple with both names and both ports) and a *user network* (the destination port or "well-known port"). The network

has a maximum transmission unit limiting the size of IP packets. So the TCP implementation at the source accepts a byte substream, disassembles it into IP packets, encapsulates each packet in the TCP/IP header, and sends it through the network. When the TCP implementation at the destination receives packets, it decapsulates them by removing the TCP/IP header, requests retransmissions of missing substreams, assembles a complete substream in byte order, and delivers it to the receiver.

4.2 Names and directories

Classic descriptions of the Internet associate "domain names" with the application layer, IP "addresses" with the network layer, and MAC addresses with the link layer. In the new model every network simply has a namespace, and network members have *names* in the namespaces of their networks. In the literature of networking, names in various networks are also referred to as "service names," "identifiers," and "locations."

In every instance of layering composition, a network A uses a network B. Some members of A must be running on the same machines as members of B, and interfacing with them to get network services. If B must set up sessions dynamically to serve A, then there must be a *directory* mapping names in A to the names of the members of B on the same machines. For example, a Web request is sent from a client to a server having a domain name in the Web namespace. For an IP network to implement this communication, it must discover the network name (IP address) of the server, which will be the destination of the TCP session carrying the request. DNS is the directory providing this information.[3]

The new model does not constrain internal implementation details of networks. For example, although most networks store member-specific forwarding tables in individual members, in SEATTLE there is a single (although distributed) forwarding table used by all members [15]. And although many networks have centralized directories, in Ethernets the directory information obtained from the Address Resolution Protocol is cached in individual members. Thus forwarding state and directory state cannot always be distinguished by the way they are implemented. But they can always be distinguished by what they are mapping: forwarding state maps destination names to members/names in the same network, while directory state maps names from one network to names in another network.

4.3 Service properties and compositional reasoning

A network offers to its users one or more communication *services*, each specified as a set of *properties*, and some associated with the use of specific session protocols. Some properties are defined on individual sessions, while others are defined on aggregates of sessions. In general, the properties fall into four categories:

- *Reachability* properties specify which receivers a member can send packets to.
- *Performance* properties specify quantities such as maximum latency, minimum bandwidth, maximum packet loss rate, and faults tolerated.
- *Behavioral* properties are more service-specific. In addition to TCP guarantees, they include synchronization, load-balancing among user endpoints, and the requirement that a session must persist despite physical mobility of one or both endpoint machines.
- *Security* properties are diverse. For example, access control is the negation of reachability. Denial-of-service protection supports availability. Security properties on individual sessions include endpoint authentication, data confidentiality, data integrity, and privacy.

In addition to providing specified services, network designers and operators are also concerned with efficient *resource allocation*, so that the services are provided at minimal cost.

Basic reasoning about composition by layering is easy to explain. There should be a one-to-one mapping between implemented links and implementing sessions. The packet load on the link, possibly fragmented into smaller packets, becomes the packet load on the implementing session. The guaranteed properties of the session become the assumed properties of the implemented link.

Although such rigor is not always needed, it should be possible to reason that a network satisfies its service specifications, and that its use of resources is close to optimal. Network designers have been very successful at this, at least with respect to performance properties. They have learned to abstract the effects of used and using networks, and have developed effective optimization algorithms and tools for self-contained networks.

Reachability, behavioral, and security properties are not so well understood. In §5 and §6 we will discuss examples in which the new model captures the structures and relationships needed for reasoning compositionally about these properties.

5 BRIDGING AND SECURITY

5.1 Bridging

In our model a network has a single administrative authority, which is responsible for providing the network's services with their specified properties. *Bridging* is a composition operator in which sessions or a service are implemented by a set of networks chained end-to-end. With bridging, the two endpoints of a session can be members of different networks. The public Internet consists of a large number of autonomous IP networks, composed by bridging.

There are several variations on bridging, depending on how much structure the bridged networks share. In the simplest case two bridged networks have identical designs and protocols, names of all network members are unique across both networks, and members of both networks have access to the routing and directories of the other. In this simple case, the networks can be bridged by shared links, and little changes except that the reach of both networks is extended. This is how public IP networks are bridged.

In other cases, bridged networks are less similar. They may have different or overlapping name spaces. They may have unshared routing, unshared directories, or other barriers. In these cases a member of one network can still reach a member of a bridged network, but only with the addition of *compound sessions*. A *compound session* is simply a session in which there is at least one middlebox acting as a *joinbox*. The joinbox serves as a destination for one simple session and a source for another simple session, and

[3] In cases where DNS maps a domain name to the server nearest the client, the domain name does not uniquely identify a server.

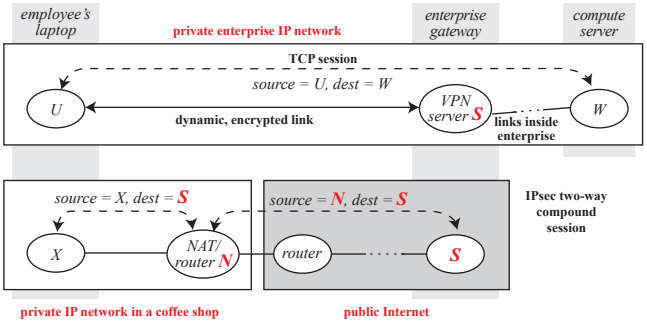

Figure 4: VPN architecture. Public names are in boldface red, while private names are not. Light-gray boxes show attachments of members within the same machine.

maintains state that associates the two simple sessions so it can forward packets from one to the other.[4] If two bridged networks have incompatible session protocols, then a joinbox, acting as a protocol converter, must be the shared element between them.

We will now introduce a simple, familiar example which will illustrate bridging, trust, and service verification. Figure 4 shows two private networks communicating through the public Internet, although their relationships to the public Internet are not symmetric. In this example an employee's laptop using the private IP network in a coffee shop is connected to the public Internet through bridging. At a higher level, using Virtual Private Network (VPN) technology layered on top of the previous networks, the laptop joins the employer's private enterprise network, and accesses a compute server within it. We will look at the bridging first.

It has been a long time since there has been enough room in the IPv4 32-bit name space to give every networked machine a unique name. Outright exhaustion of the name space was delayed by the fact that most private networks re-use the same set of private IP addresses. The cost of this strategy is that private IP addresses are ambiguous except in their local context, and a machine with a private address cannot be reached from outside its local network except with a compound session.

In Figure 4, the joinbox for the compound session is the coffee shop's IP router, which incorporates the functionality of Network Address Translation (NAT). The bidirectional compound session is initiated from the private address X, to public address **S**. Upon receiving the session-initiating packet, the NAT/router alters it before forwarding, thus making an outgoing session with its own public address **N** as the source. When **S** accepts this session and sends packets in the reverse direction, it uses reachable **N** as the destination rather than unreachable X. In this figure the dark-gray box represents the public Internet as one network, ignoring the fact that it is really a bridging of many networks. Bridging is shown explicitly by the link and session across a network boundary. In the usage hierarchy, the enterprise network uses both lower-level networks.

[4] A joinbox must change at least one of the source or destination in the session header; it may or may not be a "proxy," which is a session-protocol endpoint. For example, the NAT in Figure 4 is a joinbox and not a proxy.

At the higher level of Figure 4, the enterprise network is also a private IP network, with private addresses U and W, and public address **S**. The laptop joins the enterprise network by creating a dynamic link to the VPN server. The link is implemented by the IPsec session, so that packets are transmitted in encrypted and authenticated form. The VPN server authenticates the laptop, which has secret credentials issued by the enterprise, and gives it temporary address U within the enterprise network. At this point the laptop can initiate a session with compute server W, using TCP as the session protocol in the higher-level IP network.

5.2 Verification of trustworthy services

To prove security properties, some entities must have responsibilities and be trusted to fulfill them. Normally the entity that is trusted is a machine because the whole machine has a single owner,[5] but trusted to do what, and by whom? A machine can have members of multiple networks, and in each network its member can play a different role.

In networks bridged together in and with the public Internet, as on the lower level of Figure 4, a network's administrative authority owns routers (and other infrastructure machines) and trusts them to behave as specified. Because the administrative authority does not trust the user members (endpoints), the behavior of the routers and other infrastructure machines should be sufficient to provide the specified services in cooperation with well-behaved endpoints, and to protect the network from ill-behaved endpoints. Beyond the technical sources of trust, economic relationships provide incentives for administrative authorities to ensure that networks satisfy their service specifications [6].

In Figure 4 the employee's laptop and enterprise gateway have network members that are not trusted by their Internet providers, but are trusted by the enterprise. The VPN server does not allow the laptop's member U to join the enterprise network until it shows that it is trustworthy by sending secret credentials.

This VPN architecture enforces two security properties:

- Only packets originating at members of the enterprise network should be allowed to reach W.
- All enterprise data being transmitted outside the walls of the enterprise should have confidentiality and integrity, meaning that no external agent can read or alter the data.

The second property is guaranteed by the IPsec implementation of dynamic links outside enterprise walls. To prove the first property, it is necessary to establish that only packets transmitted on links in the enterprise network (which is not bridged to other networks) are forwarded to W. The easiest way to prove this is to rely on the fact that dynamic links of the enterprise network are associated with specific lower-level sessions. Then it is only necessary to check—no matter what packets the public Internet delivers to its member **S**—that the member drops all received packets unless they belong to sessions implementing dynamic links.

The VPN example is especially simple because the security mechanisms at both levels are implemented on the same machine. The same verification pattern works for more complex security mechanisms, however. The common structures are a secure network

[5] These terms must be refined slightly to apply to clouds, in which a machine hosts virtual machines.

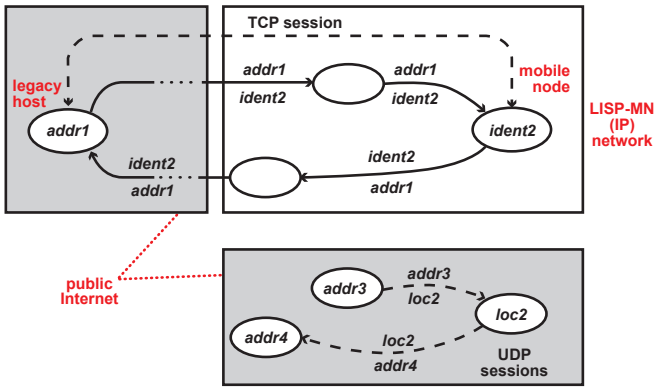

Figure 5: The interoperation of LISP-MN with the public Internet. Each link (solid line), session (dashed line), or path of links and forwarders (solid line broken with dots) is labeled above with the source of the packets traveling on it, and below with their destinations.

layered on top of the public Internet, and a packet-filtering mechanism that prevents harm (including denial-of-service attacks) at the level of the public Internet [1]. The secure overlay carries only approved packets, as enforced by its ingress members. The packet filters are on different machines, and need only have enough knowledge to reject packets not belonging to sessions implementing links of the overlay.

These examples barely scratch the surface of network security. Nevertheless, a broad survey of security mechanisms [24] has shown that the compositional model is important for understanding all aspects of security, and for working toward a comprehensive proof framework. The model is especially valuable for discovering how security interacts with other aspects of network architecture such as session protocols, routing, virtualization, and middleboxes.

6 THE USAGE GRAPH

One of the most interesting aspects of composition is that sometimes the "usage hierarchy" is a convenient fiction, because composition creates a *usage graph* with cycles. It is still useful to think in terms of usage hierarchies, provided that we remember they are approximate abstractions with localized exceptions.

Mobility is a network service that preserves reachability to a network member, and may even preserve the member's ongoing sessions, even though the member's machine is moving. One kind of mobility is provided by LISP Mobile Node [8, 9] (for a survey of all kinds of mobility, see [23]). With LISP-MN, a machine has a network member with a persistent IP address called an "identifier." In a lower-level IP network, the machine has a member with a temporary, location-dependent IP address called a "location." As a new and lightweight way to provide mobility, LISP-MN must interoperate with the public Internet. Figure 5 shows how. As in Figure 4, the public Internet is depicted as if it were one network.

At the top level of this figure, the public Internet is bridged with a LISP-MN network, which is a specialized IP network. The LISP-MN network owns a range of IP addresses, from which identifiers are drawn. Because of the bridging, a legacy host with IP address *addr1* has been able to initiate a TCP session with a mobile node whose identifier is *ident2*.

The shared elements for bridging are the unlabeled middleboxes. In both networks these middleboxes resemble IP routers, in that they forward packets and do not behave as session endpoints. The middleboxes advertise the mobile range of IP addresses into the public Internet, which means that each packet destined for an address in this range will be forwarded to one of them. The LISP-MN network has a directory mapping identifiers of mobile nodes to their current locations. When such a middlebox receives its first packet for *ident2* (or first in a long time), it gets *ident2*'s location *loc2* from the directory, creates a dynamic link to *ident2*, and forwards the packet on it. Subsequent packets to *ident2* use the same link.

The LISP-MN network is layered on top of the public Internet, so that dynamic LISP links are implemented by public UDP sessions. On the same machines as the three members of the LISP-MN network there are members of the public Internet with IP addresses *addr3, addr4,* and *loc2*, and these are the endpoints of the UDP sessions. When a mobile node changes its location, it notifies all the middleboxes with which it has dynamic links, and also updates the directory. The UDP sessions will move to the new location, but the LISP-MN links will remain.

Like Figures 3 and 4, Figure 5 uses vertical position to imply a usage graph. In this usage graph, the LISP-MN network is both bridged with the public Internet (at the same level) and layered on it. To avoid drawing the cycle, we depict the public Internet in two places. This graph shows a common pattern for interoperation of special-purpose IP networks with the public Internet.

Figure 3 is another example of a usage graph with a cycle in it. As in Figure 5, rather than drawing a cycle, we have put a network—here the campus IP network—in the figure twice. At the bottom level of the figure, the only physical connection between LANs is the campus IP network. The link shown is exactly the same as the link between 0.4 and 0.5 at the top level of the figure. When an IP packet is sent from source 2.7 to destination 2.8, it is encapsulated in a VLAN header with source M7 and destination M8. When that packet is traversing the VLAN link between M4 and M5, it is further encapsulated in an IP header with source 0.4 and destination 0.5. This packet format is called the "VXLAN" format.

Special-purpose virtual links in IP networks are often called "tunnels." Our model provides a structured view of tunnels, clarifying the roles of network members at the upper and lower levels of tunnel endpoints, the state that each network member requires, and the fields that must be present in packet headers. This uniformity across levels can explain confusing designs and make them analyzable. For example, even though a network can use itself in a usage graph, a network link must never use itself.

7 POTENTIAL BENEFITS OF THE COMPOSITIONAL MODEL

Since 1993, the Internet has evolved by means of new networks and new compositions. The Internet today is a vast collection of networks composed in a rich variety of ways by layering and bridging, including being composed with themselves. Networks are easy to

add locally (campus networks, cloud computing) or at high levels of the approximate usage hierarchy (mobility, distributed systems). They are slower to disseminate when both global and low in the hierarchy (IPv6).

This evolution, while necessary to keep up with increased demand, new technology, and many new requirements, has created tremendous complexity. First and foremost, our compositional model describes the current complex Internet as precisely as the classic Internet architecture described the Internet of 1993. Because it is inherently modular, it also has the potential to organize, explain, and simplify as well as to describe.

Based on our experience applying the model to all kinds of networks and aspects of networking, there are two primary reasons for adding a new network to the global Internet architecture:

- The network provides a specialized service or unusual cost/performance trade-off through mechanisms that are not compatible with the general-purpose classic Internet design.
- There is a need for two different instances of a network structure with two different purposes. As in LISP-MN, member names might be either permanent identifiers or temporary locations. For another example, the topology of a network might be dictated by security partitions (VLANs) or by paths through required middleboxes, as well as by physical connectivity.

Layered networks hide information, which can make problem diagnosis very difficult [19]. On the other hand, separation of concerns into different networks is a way of taming complexity. This is especially obvious when networks are being added for the second reason, and distinct topologies (for example) are maintained by distinct networks. Also, very often, it is more efficient to compose two networks than to intertwine distinct structures in the same network. This is illustrated well by [16], which shows that the conflation of a middlebox topology and a physical topology would cause a combinatorial explosion of router state.

The most immediate potential benefits of the new model are based on its capacity to explain the complexity that is already present and must be dealt with. The model can be formalized through analytic tools and reasoning technology, in support of robustness and verification of trustworthy services. We also believe that the model should be used in graduate-level teaching, to cover a wider variety of networks in a shorter period of time, and to encourage recognition of patterns and principles.

Next, the model has the potential to improve current design and development of software-defined networks. Re-usable patterns would both increase the availability of different points in a trade-off space, and make each easier to deploy by means of re-usable or generated software. Optimizations should become easier to apply, because the model can help us reason that they are safe.

Finally, a compositional model may help us to find a simpler future Internet architecture that truly meets forseeable requirements and might even adapt to unforseeable ones. Perhaps, with study of compositional principles and compositional reasoning, we can discover optimal uses of composition, in configurations that exploit its benefits and ameliorate its disadvantages. This could be the basis of network architectures that offer both flexibility and manageability. Pushing Internet evolution in this direction would be a truly worthy goal.

ACKNOWLEDGMENTS

We are grateful to the reviewers, whose comments have greatly improved the presentation.

REFERENCES

[1] David G. Andersen. 2003. Mayday: Distributed filtering for Internet services. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*.
[2] David G. Anderson, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. 2008. Accountable Internet Protocol (AIP). In *Proceedings of ACM SIGCOMM*.
[3] Theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. 2011. CloudNaaS: A cloud networking platform for enterprise applications. In *2nd ACM Symposium on Cloud Computing*.
[4] Marjory S. Blumenthal and David D. Clark. 2001. Rethinking the design of the Internet: The end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology* 1, 1 (August 2001), 70–109.
[5] David D. Clark. 1988. The design philosophy of the DARPA Internet protocols. In *Proceedings of SIGCOMM*. ACM.
[6] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. 2005. Tussle in cyberspace: Defining tomorrow's Internet. *IEEE/ACM Transactions on Networking* 13, 3 (June 2005), 462–475.
[7] John Day. 2008. *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall.
[8] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. 2013. The Locator/ID Separation Protocol (LISP). IETF Request for Comments 6830. (January 2013).
[9] D. Farinacci, D. Lewis, D. Meyer, and C. White. 2017. LISP Mobile Node. IETF Network Working Group Internet Draft draft-ietf-lisp-mn-01. (October 2017).
[10] Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2014. The road to SDN: An intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review* 44, 2 (2014), 87–98.
[11] V. Fuller, T. Li, J. Yu, and K. Varadhan. 1993. Classless inter-domain routing (CIDR): An address assignment and aggregation strategy. IETF Network Working Group Request for Comments 1519. (1993).
[12] Mark Handley. 2006. Why the Internet only just works. *BT Technology Journal* 24, 3 (July 2006), 119–129.
[13] ITU. 1994. Information Technology—Open Systems Interconnection—Basic Reference Model: The basic model. ITU-T Recommendation X.200. (1994).
[14] Marten Karsten, S. Keshav, and Sanjiva Prasad. 2006. An axiomatic basis for communication. In *Proceedings of HotNets-V*. ACM.
[15] C. Kim, M. Caesar, and J. Rexford. 2011. SEATTLE: A scalable Ethernet architecture for large enterprises. *ACM Transactions on Computer Systems* 29, 1.
[16] Zafar Ayyub Qazi, Cheng-Chun Tu, Louis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. 2013. SIMPLE-fying middlebox policy enforcement using SDN. In *ACM SIGCOMM*.
[17] Timothy Roscoe. 2006. The end of Internet architecture. In *Proceedings of the 5th Workshop on Hot Topics in Networks*.
[18] J. Saltzer, D. Reed, and D. D. Clark. 1984. End-to-end arguments in system design. *ACM Transactions on Computer Systems* 2, 4 (November 1984), 277–288.
[19] Oliver Spatscheck. 2013. Layers of success. *IEEE Internet Computing* 17, 1 (2013), 3–6.
[20] Arun Venkataramani, James F. Kurose, Dipankar Raychaudhuri, Kiran Nagaraja, Suman Banerjee, and Z. Morley Mao. 2014. MobilityFirst: A mobility-centric and trustworthy Internet architecture. *ACM SIGCOMM Computer Communication Review* 44, 3 (July 2014), 74–80.
[21] Yuefeng Wang, Ibrahim Matta, Flavio Esposito, and John Day. 2014. Introducing ProtoRINA: A prototype for programming recursive-networking policies. *ACM SIGCOMM Computer Communications Review* 44, 3 (July 2014).
[22] Minlan Yu, Jennifer Rexford, Xin Sun, Sanjay Rao, and Nick Feamster. 2011. A survey of virtual LAN usage in campus networks. *IEEE Communications* 49, 7 (July 2011), 98–103.
[23] Pamela Zave and Jennifer Rexford. 2013. The design space of network mobility. In *Recent Advances in Networking*, Olivier Bonaventure and Hamed Haddadi (Eds.). ACM SIGCOMM.
[24] Pamela Zave and Jennifer Rexford. 2018. Network Security. https://www.cs.princeton.edu/courses/archive/fall18/cos561/papers/Security18.pdf. (2018).
[25] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, and Van Jacobson. 2014. Named Data Networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (July 2014), 66–73.