A Theory of Networks: In the Beginning . . .

Pamela ZAVE
AT&T Laboratories—Research, New Jersey, USA

Abstract. The *geomorphic view of networking* is a modular theory of contemporary networking, which is currently being developed. This paper presents the basic components and ideas of this theory in the form of a genesis story—what is needed to build the simplest possible network, and why other structures must be added as the network grows bigger and more complex. The story ends with the contemporary Internet, which is vastly more complex than what the classic Internet architecture describes. The paper also summarizes work on formalizing the theory and applying it to solve real problems.

Keywords. Internet, network architecture, network protocols, network verification, Alloy, Spin

1. Introduction

The original Internet architecture was intended to empower users and encourage innovation [3], and it has succeeded beyond most people's wildest dreams.

As a result of this success, the Internet has outgrown its original architecture, and does not meet current needs in many areas. The networking community has recognized serious deficiencies concerning security, reliability, mobility, quality of service, and resource management. It is proving difficult to achieve the desired convergence of data, telephone, and broadcast networks, and difficult to balance the needs of all of the Internet's stakeholders [4,6,8,14].

The classic Internet architecture has five layers, as shown in Figure 1. Today requirements not met by this architecture are satisfied by adding many *ad hoc* intermediate layers of virtual networking, as described by Spatscheck [15]. To illustrate these *ad hoc* layers, Figure 2 shows the headers in a typical packet transmitted across the AT&T backbone.[1] The problem with these *ad hoc* layers is that each is typically designed and understood in isolation. There are many unnecessary limitations and complications. The overall network behavior when new layers are run amongst old layers is neither understood nor predictable.

In the future, the Internet will need to support an even wider variety of applications, stakeholders, resources, endpoint devices, communication functions, service guarantees, and resource policies than it does today. Consequently, from

[1] Layers and headers are related but not in one-to-one correspondence. The relationships among Figure 1, Figure 2, and the theory of networks are explained in Section 2.7.

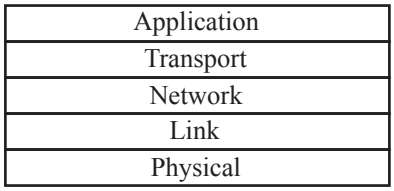

Figure 1. The classic Internet architecture.

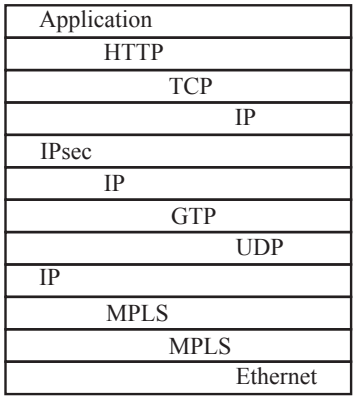

Figure 2. Headers in a packet in the AT&T backbone.

applications and middleware at the top of the Internet stack all the way down to the physical resources, there is a need for more elaborate technology, with both flexibility *and* predictability.

One approach to achieving both flexibility and predictability is *compositionality*. The basic idea of a compositional theory is well-known in formal methods. The theory defines the parts of a basic module. Through axioms and theorems, it expresses their semantics and other properties. The theory also defines how modules can be composed to make more complex assemblies. Axioms and theorems also provide the semantics and properties of module compositions. In this way, individual modules can be customized for flexibility and diversity of goals, while the compositional theory makes it possible to reason uniformly about assemblies of them.

Contemporary networking is badly in need of these benefits. In addition, although real networks are vastly more complex than any theory can be, compositional thinking would benefit the practice of networking in two important ways:

- It would help make capabilities *synergistic*, in the sense that the same user or traffic can benefit from all of them.
- It would help make capabilities *non-interfering*, in the sense that one does not break or compromise another.

In today's networks, new capabilities must be hemmed in carefully, because they can only work on particular traffic or within a particular region.

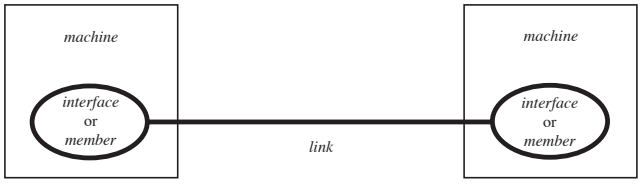

Figure 3. The simplest network.

The key to a compositional theory of networks was given to us by John Day, who showed that there are patterns that appear in network architectures at many levels for many different purposes [5]. Jennifer Rexford and I have been developing and extending these key insights, with the goal of producing a comprehensive, formal, and compositional theory of networks. Some of the practical uses for such a theory are described in [19].

Compared to well-known successful theories, and even successful theories of aspects of networking (for example [7]), the theory of networks has many "moving parts." On first exposure, people often have a hard time understanding why there are so many parts, and believing that these are necessary the correct choices. The theory is convincing after one has used it to describe dozens of real and varied examples, but this experience is not available to a general audience.

In hopes of answering some of the many questions that arise, Section 2 explains the basic parts and their purposes starting from the simplest possible network. This section is also a gentle introduction to the theory. Section 3 is a note about the generality of the theory.

The remaining sections provide an overview of our progress on the theory so far. Section 4 discusses formalization, while Section 5 summarizes results of the theory.

2. The genesis of networking

2.1. The simplest network

Suppose that we have two computers, and we wish them to communicate (Figure 3). There must be a physical medium through which they can send digital messages to each other, for example a wire or radio channel. In the figure, each computer is called a *machine*, and the physical medium that connects them is a *link*.

Two other ingredients are needed to make this simple network function. There must be a *link protocol*, which is a set of conventions about how digital messages are represented on the physical medium, and how they are sent and received. Each machine must have an *interface*, which is a software or hardware process. The interface communicates with the rest of the machine through the machine's operating system, sending and receiving messages on the machine's behalf. The interface follows all the rules of the link protocol in formatting, transmitting, accepting, and interpreting messages.

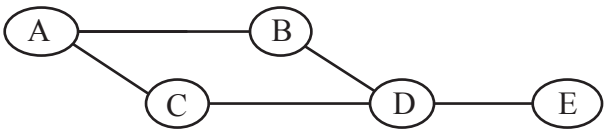

Figure 4. A network with names and forwarding. Machines are not shown.

The bold parts of Figure 3 form an elementary network. Henceforth an interface will be called a *member* of its network.

2.2. The forwarding protocol

As a network, Figure 3 is small. More realistically, we would have a network with many members representing many machines, and many links between pairs of them. All the links would use the same link protocol, which all the members would follow.

If the network is large, it may be impractical to have a link between every pair of members. More realistically, there would be enough links to allow a path of one or more links between each pair of members, as in Figure 4.

Now the problem is to deliver messages to their destinations over paths of links. The universal solution to this problem is:

- Each member has a unique *name*, which has a digital representation.
- The name of the destination of the message is added to the message, in a data structure called the *header*.
- When a message arrives at a member, and the destination of the message is not the name of that member, the member *forwards* the message toward its destination on another link.

To implement this solution, the network state must include *routes*, which tell the members where to forward messages so that they reach their destinations. In a network, *routes* is a shared, distributed data structure. It can be formalized as a relation or table with four columns of types *name, link, member, link*. If a tuple *(destName, inLink, forwarder, outLink)* is in the relation, then a message with destination *destName* received by *forwarder* on its link *inLink* must be forwarded by sending it on *outLink*. Clearly *forwarder* must be an endpoint of both *inLink* and *outLink*, and *destName* must not be the name of member *forwarder*.

More generally, forwarding can be based on other fields in the message header in addition to the destination, for example the name of the message source. A *forwarding protocol* is a set of conventions about the formats of message headers, about the representation of route information, and about how the members should behave in sending, receiving, and forwarding messages. Usually the link protocol of a network is not described separately, but is considered to be part of the forwarding protocol.

2.3. Network state is dynamic

So far the state of a network (which is distributed over its members) consists of its members, some representation of its links (including which members are

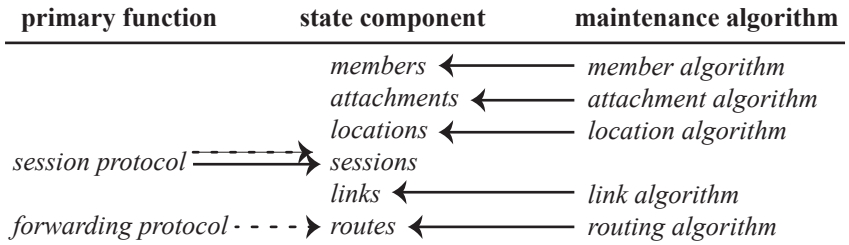

Figure 5. The components of a network. Solid arrows show which protocol or algorithm writes a state component, while dashed arrows show which state component is read by a primary function.

connected by each link), and its routes. Each of these state components must be initialized ("configured" or "provisioned") when the network starts up. The state components of a network (some of which have not been introduced yet) are listed in Figure 5.

Many networks are highly dynamic. This mean that all their components can change as the network runs. Members and links can change as a result of failed users and resources, or new or re-instated users and resources. Because routes are dependent on members and links, they must change as members and links change. Routes can also change as a result of changes in resource-allocation policies..

In general, there is a distributed algorithm to maintain each of these state components over time. Note that the network's information about a member can include authentication credentials and other data as well as its name. In most networks the *routing algorithm* is an important part of the network design, as it consumes many resources and has a big influence on network performance.

2.4. The session protocol

A network's forwarding protocol defines how members of the network can send messages to one another, which is the network's most basic function.

A forwarding protocol, however, has intrinsic limitations. It handles each message independently. It is not completely reliable because members and links can fail. Even if there is a healthy path between message source and destination, the routes may be out-of-date and forward messages to failed regions, where they will be lost.

To make up for these deficiencies, most networks have *sessions*. Like a link, a session is a communication channel through which members can send messages to each other. The difference is that a session is an instance of a service implemented using the network's links and forwarding protocol. The network's *session protocol* enables members to use this service. Session service in a network can include many conveniences, such as:

- Reliable delivery of messages, by means of failure detection and retransmission.
- FIFO and/or duplicate-free delivery of messages. Note that the session delimits the group of messages (those sent through the session in one direction) over which FIFO delivery is defined.

- Performance guarantees ("quality of service").
- Security guarantees (authentication of endpoints, encryption).

The implementation of the session protocol must provide these services on top of the forwarding protocol.

The identification of a session has four parts. For each endpoint there is the *name* of the member, and for each endpoint there is a *port* that distinguishes the session from all others of the same member. Ports are necessary because a pair of members may have more than one session between them simultaneously.

Figure 5 would be more symmetric if there were a session maintenance algorithm separate from the session protocol that uses sessions. In many networks, however, the session state is by far the most rapidly changing of the six state components. Thus it is conventional to speak of one protocol that sets up, tears down, and uses sessions, while also maintaining whatever state is necessary to implement the session services.

2.5. Joining networks

Now suppose that we have two networks A and B, and we wish to join them, *i.e.,* to make it possible for any machine using A to communicate with any machine using B.

Perhaps the simplest approach is to install some new links between machines/members of A and B. Unfortunately this will not work in general, because A and B may have different session and forwarding protocols—so their members do not "speak the same language." Even if A and B use the same protocols, there may be members of A with the same names as members of B, so that names in a merged network would not be unique.

Another approach would be for A and B to have some shared or dual members. This approach is very similar to the one above, and has the same deficiencies.

The only completely general solution to the problem is to create a *new* network used by the machines now serviced by A and B. In the new network, all members have unique names, and implement the same protocols. This is a sensible solution if and only if we can *implement this new network using A and B*, rather than discarding them and starting over.

The way to do this is shown in Figure 6. Each member of new network C must communicate through the operating system of its machine with a member of A or B, depending on which existing network the machine uses. In the state of network C, the member of C is *attached* to the member of A or B. In the state of A or B, the member is the *location* of the member of C. In the figure attachments/locations are shown as vertical dotted lines, and are always within the same machine, although the machine boundaries are not shown. Corresponding attachments and locations have almost the same information, but each of the two networks involved needs its own copy for its own purposes. Furthermore, each network will have its own data representation, distribution, and maintenance algorithm for it.

To create a physical connection between A and B, at least one machine must belong to both subnetworks. On this "gateway" machine there are members of

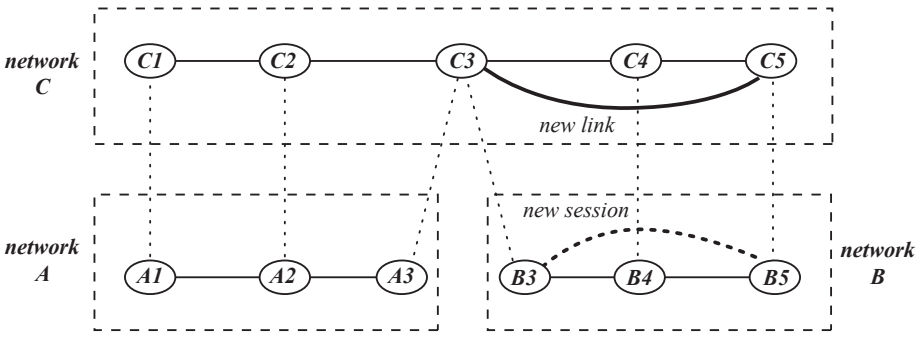

Figure 6. A network C implemented using networks A and B.

all three networks; for example the member *C3* of C is attached to members of both A and B.

Functionally the relationship among these networks is quite simple. Each *link* of C is implemented by a *session* of A or B, depending on which subnetwork its endpoints are attached to. To see this in more detail, let us consider one way to add a new link to C between *C3* and *C5*. As both *C3* and *C5* have attachments to network B, this link will be implemented by a new session between *B3* and *B5*.

If *C3* initiates formation of the new link, it might send to *B3* (through the operating system of its machine) a request to form a new link to *C5*. *B3* looks up *C5* in the *locations* state component of its network, and finds that it is located at *B5* in network B. An exchange of messages between *B3* and *B5* sets up the session. A message from *C3* to *C5* travels as follows:

1. *C3* sends the message to *B3* through the operating system of their machine.
2. *B3* encapsulates the message from *C3* to *C5* by enclosing it in a message with source *B3* and destination *B5*. This message is sent through the new session to *B5*.
3. At *B5* the received message is decapsulated by stripping off the B-level names, and delivered to *C5* through the operating system of their machine.

A message from *C5* to *C3* travels the reverse path. Messages between *C3* and *C5* create the dynamic link and use it to transmit data for C in both directions.

These details illustrate that a link in a network can be virtual (as the links in C are) as well as physical (as the links in A and B are).

The new session in B is not intended primarily to benefit the members of B. Rather, its primary purpose is to implement a *communication service* that is used by C. So one network can export and implement a communication service that is used by other networks, provided that the communicating members of a user network are *attached* to members of the implementing network. Apart from the necessary programming interfaces for attachment and services, each network is independent and adheres to all previous descriptions and constraints.

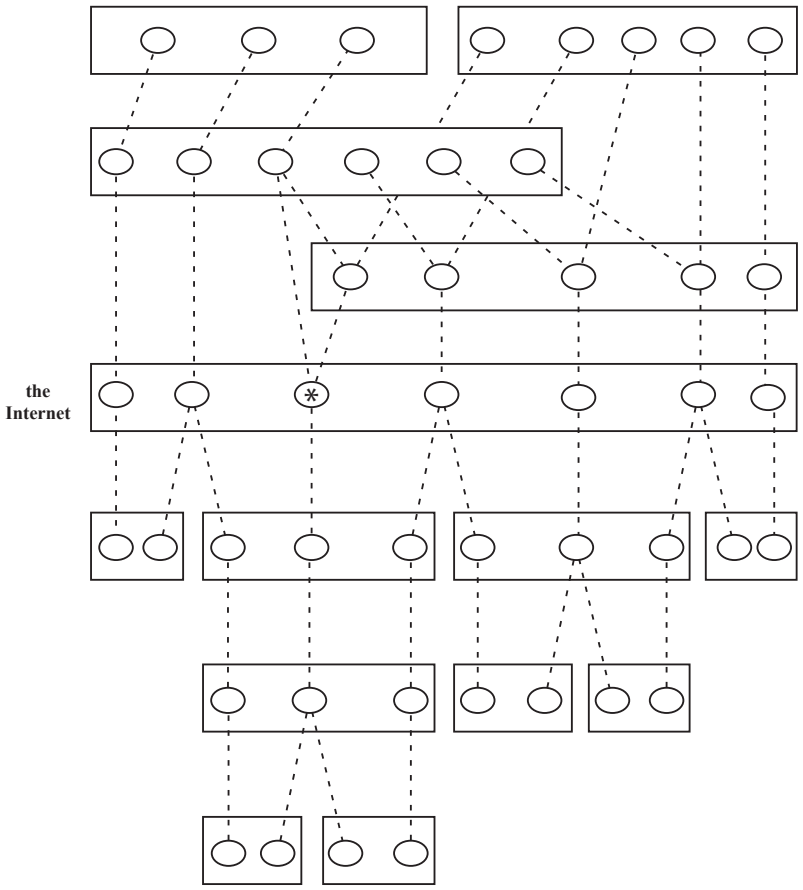

Figure 7. The relationship among networks world-wide. A network with attachments in another network uses that network directly.

2.6. The "uses" hierarchy

The Internet consists of a very large number of networks arranged in a "uses" hierarchy. The exact nature of the "uses" relation is explained in Section 2.5 and Figure 6.

Figure 7 is a large-scale, extremely simplified view of the shape of the world-wide "uses" hierarchy. In interpreting this figure, it is important to realize that each network spans some area—from house-sized to continent-sized—of the (approximately) two-dimensional surface of the earth. So the largely one-dimensional picture of each network does not do justice to its span and ability to connect machines.

In Figure 7, the largest network is the Internet. It has the capability to connect any two machines in the world. The Internet Protocol (IP) is its forwarding protocol, and TCP is its session protocol.

The Internet uses (from a top-down perspective) or connects (from a bottom-up perspective) thousands of networks with fewer members and smaller physical

spans. Ownership and administrative authority are important attributes of each of these lower-level networks. In larger autonomous domains, there can be multiple levels of subnetworks, because each level implements (in its routing algorithm) a different kind of resource management. A member at a higher level has one or more attachments to lower-level networks, depending on whether or not it is a gateway.

Although it is less well understood, there is also a hierarchy of virtual networks that use the Internet and are therefore above it in Figure 7. These networks may have restricted membership for security, as VPNs do. They may implement special-purpose communication services, as middleware does. Or they may connect members that are programs (clients and servers) of a specific application, communicating in application-specific ways.

Like the networks below the Internet, these networks above the Internet have fewer members. They have fewer members because they are functionally specialized, however, rather than smaller in physical span.

Note in the figure that the attachment structure above the Internet is usually the opposite of the attachment structure below it. Often members from multiple networks are attached to the same lower-level member. This is because functionally specialized cliques are sharing the resources of general-purpose communication networks. Note especially the Internet member marked with an asterisk. On its machine there are members of all four networks above, and all four of these higher-level members get their network service from the marked Internet member.

In this context, it is worth mentioning the most widely used network above the Internet, which is the World-Wide Web. Its members are Web clients and servers. Its session protocol is HTTP. It has a dynamic link, implemented by IP, for each session, and therefore has no routing or forwarding.

The name space of the Web network has two name types: IP addresses and domain names. Clients are named with IP addresses, and servers have domain names.

In a network, the *attachments* state indicates which members are attached to which network below. There is no need for explicit attachments in the Web network, because all members are assumed to be attached to the Internet network, by which the Web's links are implemented.

In a network, the *locations* state indicates which members of which higher-level networks are located at (attached to) which members of the network. For Internet locations of Web members, there are two cases. Web clients with IP addresses in the Web network have the *same* name in the Internet, so there is no need for explicit state. Web servers with domain names are located at IP addresses in the Internet; the *locations* state for them is implemented by the Domain Name System (DNS). DNS is a globally distributed directory in which any member of the Internet can look up the Internet location of a domain name.

2.7. Network architecture

Although the modules in Figure 7 are called networks, they are also often referred to as *layers* because the necessary software in each machine is organized as a hierarchy of layers. (This hierarchy of layers within network software is also known as a "protocol stack.")

Adopting the terminology of *layers*, it may seem that there is nothing new here—both the classic Internet architecture [3] and the OSI reference model [10] also describe network architecture as a hierarchy of layers. In fact, our approach is radically different from these earlier models.

In both the Internet and OSI architectures, there is a fixed number of layers. Each layer has a specialized function that is indispensable and different from the functions of the other layers. Referring to Figure 1, the link layer provides communication links by means of "local" networks with varying physical spans. The Internet network layer does routing and forwarding as defined by IP. The Internet transport layer provides sessions as defined by TCP (and occasionally other session protocols).

In our theory of networks, each network or layer is a microcosm of networking containing *all* its basic functions and state components, as in Figure 5. Because networks are self-similar, any number of them can be composed (as suggested by Figure 7).

Networks instantiated at different levels, with different scopes (sets of potential members) or with different physical spans, have different purposes. As a consequence, their functions and state components take different forms. For one example, we are most familiar with routing algorithms in the Internet core, where their purpose is global 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 destination through a particular filtering server, so that, in this layer, part of the purpose of routing is security. An application layer might create a new link between two members whenever those members need to communicate, implemented by communication services below. In this layer the routing algorithm is vestigial, as no member ever forwards.

We now consider the real packet in Figure 2. How do these network architectures describe it? It simply does not fit the classic Internet architecture, which tells us that the headers should be, from bottom to top:

1. An Ethernet header, allowing a physical Ethernet to implement a link.
2. An IP header, for forwarding in the network layer.
3. A TCP header, creating a session in the transport layer.
4. A header for the application protocol.

In our theory of networks, this packet makes sense. From bottom to top:

1. There is an Ethernet providing the physical link on which this packet has just traveled. Its software uses an Ethernet header.
2. Above the Ethernet, there are MPLS networks at two different levels. MPLS networks create and use long-distance virtual links. Routing in these networks manages high-volume, high-speed traffic. The presence of networks at two extra levels indicates the sophistication of resource management needed to operate a large network.
3. The next three headers (IP/UDP/GTP) belong to the AT&T mobility or cellular network. IP is used as the forwarding protocol, and UDP plus GTP makes a session protocol that provides mobility and quality of service for cellphone users.

4. We can see from these headers that the cellphone is being used for a data service rather than voice. The next two headers belong to a network providing secure communication service. Its forwarding protocol is IP (again), and its session protocol is IPsec.
5. The next two headers (IP again and TCP) belong to a network providing forwarding and reliable FIFO communication service.
6. The HTTP header is the most mysterious: HTTP is usually thought of as an application protocol, but it is not playing that role here. Because of private networks and firewalls, it can be difficult to create a TCP session between two machines, one on the public Internet and one connected to the Internet through a private network. This header indicates the presence of another network in which the links are implemented by TCP at a lower level, and HTTP is a session protocol capable of forming a session that spans a public/private boundary. HTTP has been comandeered for this odd purpose for the simple reason that firewalls are more tolerant of HTTP traffic than other kinds.
7. Finally there is a distributed application system whose members communicate for their application-oriented purposes. Its links are dynamic and connect whichever application members need to communicate, being implemented by the HTTP network. Viewing this distributed system as an application-oriented network, there is no need for forwarding or a session protocol, so most network components are vestigial.

This example shows how even Figure 7 is over-simplified in implying that there is a single uniquely identifiable Internet layer. The broadest network through which this packet travels might be 4 or 5, depending on design information we cannot guess from the packet alone.

We have named this view of network architecture the *geomorphic view of networking*, because its complex arrangements of layers resemble the complex arrangements of layers in the earth's crust. The name distinguishes it from network architectures having a fixed number of layers.

3. A note on generality

Section 2 has presented an informal descriptive framework for networking. This framework has been validated by applying it successfully, over five years, to a large number of examples at all levels of networking.

Some aspects of the framework have been simplified just for presentation in this paper. For example, it was stated that every member of a network has a unique name. In actuality a member can have no name or many names. One name can map to more than one member. These variations are all expressible in the formalization of the theory (next section).

We have also discovered two additional features that will need to be added. It must be possible to partition a network into *domains*, mirroring regions of administrative authority and trust. A domain within a network will have names and routes separable from those of other domains within the network. Also, it must be possible to have *compound sessions* consisting of concatenations of *simple*

sessions, which are the same as sessions in Section 2. Compound sessions are related to domains: if naming is different in different domains, a compound session (which can have different source and destination names in each simple session) may be needed to traverse domain boundaries.

Except for these relatively minor additions, the descriptive framework of Section 2 appears to be completely general with respect to contemporary networking.

4. Formalization of the theory

To get the benefits of a real theory of networks, it is necessary to formalize the descriptive framework. Our target must be a family of inter-related models, as no single model could possibly be comprehensive and yet analyzable. Building a family of increasingly detailed and useful formal models will be a long-term project.

Most of the models so far are written in Alloy. Alloy and the Alloy Analyzer are presented in Jackson's book [11] and an on-line tutorial [1]. These models are simplified in two enormous ways:

- They describe only static states or "snapshots." There is no time and no state change over time.
- They describe the state of each network as if it were centralized. There is no formal representation of distribution or replication.

Although these simplications might seem to remove everything of interest, networking is such virgin territory for formal modeling that the models are very revealing. Two of these models, one focusing on routing and one focusing on sessions, can be found in the reference materials for the Marktoberdorf lectures [17].

For one project an Alloy model was extended with time and with operations that change the state. This model was used to prove a theorem about mobility, which is a phenomenon in which multiple networks reconfigure themselves to maintain connectivity to a moving device [21].

For some studies the temporal aspect of modeling is paramount. For this research we use the model-checker Spin and its modeling language Promela [9]. For example, some of our work is concerned with extending current session protocols to perform new functions [22].

Both Alloy and Spin perform fully automated analysis of formal models (for a detailed comparison of the two, see [18]). More specifically, they verify properties within a bounded universe, by means of exhaustive exploration of the universe. Fully automated analysis is absolutely necessary for development of a theory of networks, because the modeling work is highly exploratory. There is a continual interplay among formal description of network behavior, formal description of network properties, and verification of properties. The best outcome balances all three, rather than emphasizing one at the expense of the others. When the theory is more settled and mature, it will make sense to create real proofs of important theorems.

As stated above, a full and useful theory of networks is a long-term project. It will require models with great expressiveness with respect to states, models

with great expressiveness with respect to temporal behavior, and some way to build bridges between the two. It will require simulation, automated analysis, and theorem proving. It will also require ways to modularize and organize a large family of models, some of which present overlapping views, and some of which are specializations or refinements of others. Modeling and verification tools are improving rapidly, and it is to be hoped that their progress matches the growing use of formal methods in networking.

5. Applications of the theory

This section is a brief summary of our ongoing research that uses, enhances, and validates the theory of networks.

5.1. Mobility

Mobility is a communication service that preserves the communication channels of a mobile device (machine) as it changes its attachments to various networks. Mobility is enormously important today, but it was not important when the classic Internet architecture was designed, and the classic Internet architecture makes mobility difficult to achieve.

Although there are hundreds of protocols and proposals for implementing mobility, we have discovered that there are exactly two patterns of which all these proposals are instances. This was not known until the application of the geomorphic view made it clear to us.

The patterns are explained in a chapter of a recently published SIGCOMM electronic textbook [20]. The chapter also surveys all the major approaches to mobility, and shows how each is an instance of one of the patterns. For example, four proposals in the standardization process of the Internet Engineering Task Force are similar implementations of the same pattern, and knowledge of the pattern makes it possible to compare their subtle details.

These patterns expand into a large potential design space for mobility. As either or both patterns can be implemented in any layer, it is important to show that these implementations can compose with synergy and without interference. A recent paper shows how a compositionality property can be formalized—which is often the hardest part—and proved (for a small universe) with the help of Alloy analysis [21].

5.2. Middleboxes

Middleboxes are members of a network placed on the communication path between two endpoints (also members). Middleboxes perform a wide variety of functions, usually related to security or optimization of performance. Middleboxes are not part of the classic Internet architecture, and they have been controversial ever since the Internet became widely used.

Today the need for middleboxes is accepted, and the challenge is to introduce them into communication paths in efficient and scalable ways. This is usually done by means of routing, but we are investigating the possibility of using a session

protocol instead. In this work, the geomorphic view is a guide to how design new mechanisms for generality and compositionality [22].

5.3. Network verification

Network verification is a new and rapidly growing area of research. Network verification is based on general models of routing that are state snapshots, just like the nontemporal Alloy models in Section 4. These general models are instantiated with data from real networks. Decision procedures then check for specific instances of desirable general properties such as reachability, security blocking, and routing through middleboxes [2,12,16].

Most work on network verification is like the classic Internet architecture in the sense that it assumes a central IP layer in which all the action occurs. To the extent that there are non-classic functions such as middlebox insertion, they must be encoded or embedded in ordinary IP routing.

In formalizing the geomorphic view, we take a complementary approach to network verification, in which the modularity of layering is paramount. This means that routing in each layer can by analyzed separately, checking for the properties required by the purposes of that layer. It also means that we can define and analyze cross-layer properties. So far we have formalized cross-layer properties in two general categories:

- A session or link is *effective* if messages can actually be transmitted through it. For a higher-level link or session to be effective, it must be properly implemented at lower levels by effective sessions and links. Failures and mobility can compromise effectiveness, and it is important to know that failure-recovery and mobility mechanisms work correctly to restore effectiveness.
- One of the most important practical aspects of routing is load balancing, which is a kind of resource allocation. However, it has been observed that many different load-balancing algorithms operate in networks, at different levels, and that they run without knowledge of each other [15]. In such situations, algorithms at different levels can interfere with each other. *Load independence* is a property saying that if two routes or implementations appear to be independent in a layer (where the resource-allocation decisions are being made), they are actually independent at lower layers (which is the implicit assumption). Without load independence, loads that appear to be balanced over alternate resources may actually be allocated to a single bottleneck resource at a lower level.

Formalization of the model is different from current work on network verification in another way as well. Rather than emphasizing decision procedures on specific data, we are currently emphasizing theorems that relate desirable properties to each other and to optional properties of network architectures.

6. Conclusion

To an outsider, the field of networking presents an enormous barrier to entry. Networking appears to be all details and no principles [13], and the number of details

to master grows rapidly indeed. Even when mechanisms are well-documented, their purposes are not.

From an outsider's perspective, the benefit of the geomorphic view of networking is that it is a lens through which networking makes sense. By decomposing complexity into networks or layers, and looking for the common components and patterns within each layer, it becomes possible to understand the purposes for structures, and to ask the right questions about them. This has been my experience in learning about networking, and I hope that others can enjoy the same benefit.

Acknowledgments

Section 2 was conceived on a long walk around Marktoberdorf with Michael Jackson, who asked all the right questions. Alexander Pretschner encouraged me to believe that this is a good way to teach networking.

References

[1] Tutorial for Alloy Analyzer 4.0. `http://alloy.mit.edu/alloy/tutorials/online`.

[2] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker. Netkat: Semantic foundations for networks. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, January 2014.

[3] D. D. Clark. The design philosophy of the DARPA Internet protocols. In *Proceedings of SIGCOMM*. ACM, August 1988.

[4] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: Defining tomorrow's Internet. *IEEE/ACM Transactions on Networking*, 13(3):462–475, June 2005.

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

[6] A. Feldmann. Internet clean-slate design: What and why? *ACM SIGCOMM Computer Communication Review*, 37(3):59–64, July 2007.

[7] T. G. Griffin and J. L. Sobrinho. Metarouting. In *Proceedings of SIGCOMM*. ACM, August 2005.

[8] M. Handley. Why the Internet only just works. *BT Technology Journal*, 24(3):119–129, July 2006.

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

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

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

[12] P. Kazemian, G. Varghese, and N. McKeown. Header space analysis: Static checking for networks. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, 2012.

[13] J. Rexford. The networking philosopher's problem. *Computer Communication Review*, 41(3):5–10, July 2011.

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

[15] O. Spatscheck. Layers of success. *IEEE Internet Computing*, 17(1):3–6, 2013.

[16] G. Xie, J. Zhan, D. A. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford. On static reachability analysis of IP networks. In *Proceedings of IEEE Infocom*. IEEE, March 2005.

[17] P. Zave. Lectures and models for Marktoberdorf. `www2.research.att.com/~pamela/mark.html`.

[18] P. Zave. A practical comparison of Alloy and Spin. *Formal Aspects of Computing*, 2014. The final publication is available at Springer via `http://dx.doi.org/10.1007/s00165-014-0302-2`.

[19] P. Zave and J. Rexford. The geomorphic view of networking: A network model and its uses. In *Proceedings of the 7th Middleware for Next Generation Internet Computing Workshop*. ACM Digital Library, 2012.

[20] P. Zave and J. Rexford. The design space of network mobility. In O. Bonaventure and H. Haddadi, editors, *Recent Advances in Networking*. ACM SIGCOMM, 2013.

[21] P. Zave and J. Rexford. Compositional network mobility. In E. Cohen and A. Rybalchenko, editors, *Proceedings of the 5th Working Conference on Verified Software: Theories, Tools, and Experiments*, pages 68–87. Springer LNCS 8164, 2014.

[22] P. Zave and J. Rexford. Stuck in the middle for you: Middlebox-aware session protocols. Technical report, AT&T Laboratories—Research, June 2014.