Network Stacking Considered Harmful

Robert Surton
Cornell University
Ithaca, New York
burgess@cs.cornell.edu

ABSTRACT

The most important challenge facing the future Internet is not technical, but is rather the need to justify placing trust in the technical solutions. Current network models suffer from limitations that result in practical deployments being too complex to reason about. The novel channel market model, based on composing networks by sharing channels through a flat market, offers a better opportunity for reasoning. The old language is still useful, and continues to make sense in the new model. Two design principles, the haggling principle and the composition principle, provide hints for discussing and designing networks in a channel market.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks

General Terms

Reliability, Design

Keywords

Channels, network models, design principles, dependability

1. INTRODUCTION

> Dependability is the ability to deliver service that can justifiably be trusted.
>
> —Avižienis et. al. [1]

The Internet is a marvelous invention, and we have been inspired to expect more and more from it. In the future, the smart power grid will communicate power like the Internet communicates data, enabling less waste, fewer outages, and more accurate pricing. Entertainment is being radically changed by the ability to stream video on demand, wherever we go. Telephone calls, even emergency calls, are leaving the old switched circuits and traveling through the Internet. Medical records are becoming digital and federated, to enable faster, more accurate health care.

The networking research of the last several decades has provided technology for distributed computation, mobility, fault tolerance, high throughput, load balancing, privacy, and more—although the new applications bring new challenges, many of the technical ones can be solved today.

What holds us back now is dependability. Real-world implementations are too complex to be convincingly trustworthy. The interaction between different technologies combines that intractable complexity with even poorer specification. It is usually acceptable to have weak faith in web browsing, shopping, and watching movies. On the other hand, while I believe a smart power grid can be built now, I would be wary to trust it now.

Caution also makes networks slow to adapt. That is why many of the technologies of peer-to-peer computing are now being revisited under the name of cloud computing; trust is simpler within a datacenter owned by a single entity, and that enables innovation that is nearly impossible in the Internet.

I propose a new way to think about, talk about, and design networks, which I call the channel market model. I will explain the model, and revisit some old design principles and abstractions in a way that faces the challenge of dependability.

2. THE CHANNEL MARKET

> The fundamental problem of communication is that of reproducing at one point . . . a message selected at another point.
>
> —Claude Shannon [10]

The channel is the fundamental abstraction of communication [10]. A channel is any aspect of the environment that can be observed by agents and interpreted to give meaning. A network—or indeed any computation—consists of agents observing channels, and acting on the meaning they infer by manipulating further channels, which are observed by further agents, and so on.

The channel notion is independent of scale. A channel can be a fiber optic cable, a variable in a program, an entry in a distributed key–value store, or a wire between logic gates.

Every channel is an abstraction. The reason for building a TCP channel in terms of an IP subchannel in terms of an Ethernet subchannel (and on down to the realm of physics)

CF'13, May 14–16, 2013, Ischia, Italy.

Internet	OSI	
Application	Application	7
	Presentation	6
	Session	5
Transport	Transport	4
Internet	Network	3
Link	Data Link	2
	Physical	1

Figure 1: The Internet and OSI network models side by side. Both are layered, preventing networks from building up mutually. They are also fixed in scale, so protocols that build topologies using TCP or UDP subchannels are relegated to being application-layer overlays rather than true networks.

is that, for some applications, it is easier to reason about correctness in terms of the guarantees of TCP rather than the subchannels that implement it.

Thus, when designing a network to meet a new challenge, it is fruitful to think of a marketplace full of channels. Consider two examples, first a router, and then a web browser.

Routers are agents that are included in a network so that each of the other agents can make a simplifying assumption: that for each pair of them there is a channel that enables them to communicate. One common implementation is for all of the routers to flood information about their own channels, so that each of them eventually learns of all the channels available in the market. A router uses a shortest-paths algorithm on its view of the market to find paths that can be used to implement direct communication. It offers the new channels into the market, so that other agents, such as web browsers, can build from there.

Now consider how a web browser solves the problem of downloading a web resource. The desired channel will carry HTTP to a destination that is capable of serving the requested resource, and HTTP in turn depends on a reliable, byte-oriented subchannel. At the browser's request, a TCP agent can offer such a channel implemented in terms of an appropriate IP subchannel, which might originally be found in the market with the assistance of a DNS agent. After building up the appropriate abstraction, the browser can make its request.

The task of justifying trust in a network is simplified by its construction from channels found in the market. Each agent can be analyzed independently to show just one thing, namely that when the agent offers a channel with some specification, it can keep that promise.

3. NETWORK STACKING CONSIDERED HARMFUL

> ... the purpose of abstracting is *not* to be vague, but to create a new semantic level in which one can be absolutely precise.
>
> —Edsger Dijkstra [5]

As a network model, the channel market model has two competitors: the OSI model and the Internet model [2]. They both divide protocols into layers based on where in the network they are implemented. For example, in the Internet model, the link layer provides connectivity within local networks, the Internet layer connects such networks into a global address space, the transport layer adds end-to-end functionality, and everything else is at the application layer. The two models correspond closely, as shown in Figure 1, although the OSI breaks the application and link layers down into more specifics. The OSI model also numbers each layer, which leads to terminology such as 'layer 7 switch' for a network device that inspects application layer data to make decisions.

There are two things wrong with the layered models. First, layering puts unnecessary limits on how agents can compose their functionality. Layering has, in fact, already been considered harmful by the IETF [3], in defense of the less strict approach taken by their Internet model compared to the OSI model.

Second, and worse, the models are wrong—real-world protocols have dealt with the limits of layering by sidestepping it, so that although one of the advantages of layering should be that each layer can evolve separately, in practice they are very tightly coupled. For example, the TCP standard defines the checksum for each segment to include header fields from the IP version 4 packet that carries it. A separate standard specifies how to compute the checksum for version 6. Using TCP over any other subchannel is undefined.

If the old models were just wrong, it would be easy enough to throw them away; the real problem is that they are very useful. Having language for the concepts of layer 2 and layer 3 switches is so useful that when protocols like MPLS started to blur the lines between those layers, the language expanded to call them layer 2.5.

Thus, to ease the transition away from layering, I will revisit some of the most useful concepts in the current language, showing that they still make sense—generally more sense—in the channel market model.

3.1 Control Plane

Control channels are the storefronts of a channel market. A control channel carries and describes other channels. Although it can be convenient to think of a channel market as a pool of resources to be dipped from and poured into, in fact there are only agents and channels. For example, a routing agent offers a path channel to an IP agent, which creates multiplexed channels from it and offers one to a TCP agent, and so forth. Or, an agent sends a domain name to a DNS agent, which responds by offering an appropriate IP channel. Control channels are the metachannels for enabling agents to create and offer the resources that come to be seen, all together, as the channel market.

3.2 Paths

A path channel makes a symbol observable to its destination through a sequence of subchannels, which in this context might be called link channels, with the cooperation of agents forwarding it from hop to hop. A path channel is usually offered by an agent participating in a routing algorithm, such as BGP or OSPF. If the agent discovers that the path is no longer connected, it can implement the same path channel transparently using a different sequence of subchannels; thus, path channels are the abstraction that hides routing from agents that do not need to know.

3.3 Addresses and Ports

When an agent presents a subchannel as multiple, separate channels, the offered channels can be called multiplexed channels. Usually, a multiplexed channel is implemented over a subchannel by simply adding a unique label that can be used by the destination to distinguish the different multiplexed channels based on observations of the subchannel. Based on the terminology of the abstraction being built, such labels can be called addresses, protocols, or ports. For example, IP provides 2^{64} channels between agents, each of which it multiplexes into 2^8 protocols, such as UDP or TCP. UDP and TCP, in turn, multiplex each of their subchannels into 2^{32} flows labeled by ports.

3.4 Transport Layer

A transport channel adapts a single subchannel to present a different abstraction. For example, TCP adapts an unreliable, message-oriented subchannel into a reliable, byte-oriented channel. Ethernet adapts a constant-rate, symbol-oriented subchannel into a frame-oriented channel. TLS adds authenticity and privacy.

A transport channel can also be useful when it offers the same abstraction as its subchannel, but hides its implementation so that the subchannel can be replaced.

The end-to-end principle [9] is one of the most successful principles already used in network design; restated for a channel market, it is the principle that whenever two agents require some property to hold for a channel between them, and that property can or must be provided by adapting it with a transport channel, then the same property should not be redundantly added to lower subchannels (unless doing so provides a convincing performance advantage). Thus, transport channels are one of the most common and important concepts in channel market design.

3.5 Side Channels

A side channel provides information about another, otherwise independent channel. For example, a sniffer mimics the symbols sent on another channel. The statistics gathered by firewalls and network devices are crucial side channels for maintaining networks. Side channels carry checksums, timestamps, and acknowledgments. As control channels are metachannel channels, side channels are metadata channels.

3.6 Broadcast Channels

A broadcast channel replicates a symbol to all of its subchannels, usually with the purpose of delivering it to agents that would otherwise not be able to observe a common channel. The same behavior can also be used when the subchannels all have the same destination, enabling the channel to tolerate faulty subchannels.

3.7 Anycast Channels

An anycast channel replicates each symbol to one of its subchannels, generally with the intent that each destination will respond equivalently. The same behavior can also be used when the subchannels all have the same destination, enabling the anycast channel to exploit their combined capacity.

3.8 Application Layer

An application channel communicates directly with the environment. The environment might be a different abstraction in different channel markets, but it is always the case that the application channels are the ones not used as subchannels of any other channel in the market.

4. DESIGN PRINCIPLES

> There probably isn't a 'best' way to build the system, or even any major part of it; much more important is to avoid choosing a terrible way, and to have a clear division of responsibilities among the parts.
>
> —Butler Lampson [7]

A design principle is a heuristic that helps a person to avoid obvious poor choices, or to keep a model fresh in mind. For example, the end-to-end principle [9] and the separation of policy from mechanism [4] are two of the most important defenses against bad networks. I propose two principles to uphold the channel market model; they apply to the shoppers and to the sellers, respectively.

4.1 The Haggling Principle

Never accept more or less from the market than the network needs. Choose channels based on their specifications, with the primary goal of making the network simple to implement and easy to reason about, and the secondary goal of building on as few assumptions as possible. If the right channels don't exist, demand that some agent offers them, even if you have to implement it yourself separately.

One of the biggest current failures with respect to the haggling principle is in the concept of the 'overlay network'. All networks are overlays. In practice, the term 'overlay' usually implies that the network's subchannels are provided by UDP or TCP, which is an acceptance of too specific a specification. One of the most important features of the channel market is that it is flat, and all that matters is the specification of the channels you shop for. If a network design could work easily well over wires as over a structured peer-to-peer topology, that is a strength.

Such haggling failures also lead to poor adaptability. For example, the specification for the BGP routing protocol requires the use of TCP, and although the HTTP standard admits the existence of non-TCP transports, it is not clear how one might negotiate to use any other. The result is that alternative protocols such as SCTP see very little use, even when widely implemented within operating systems.

The dual problem of specifying too much is not specifying enough. It can be seen in many media streaming protocols, which frequently build on top of best-effort channels, despite really using partially-reliable, flow-controlled channels. The protocol then has to implement reliability and flow control; the added complexity is what lead to the modern practice of doing most streaming over TCP, even though its specification is actually too strong, simply because it already implements reliability and flow control.

4.2 The Composition Principle.

Compose networks by sharing channels. There are some topologies that can be achieved when networks can mutually share resources, that are impossible with strict layering. For example, if a network partitions internally, the BGP routing

protocol could only connect half of it to the Internet, even if both halves remain fully connected to the outside. If, instead, the network could be seen at a finer granularity, both halves could continue to use the resources available to them; if, further, the internal routing could make use of external links, the partition could be worked around until it is repaired.

The composition principle is also the reminder that composition is desirable, and that any new functionality a network creates should be offered back to the market. If not, that particular implementation might benefit, but it will not be reusable when it comes time to solve new problems.

What functionality should be offered? How much is too much to combine in a single offer? There are many examples of mistakes of this kind in the real world. For example, IP performs fragmentation and reassembly, checksumming, multiplexing up to 2^8 higher-level transport channels, and globally identifying up to 2^{32} endpoints. TCP multiplexes up to 2^{16} higher-level application channels, provides redundant checksumming, adapts a message-oriented channel into a byte-oriented channel, reliably delivers each byte in order, and performs flow control.

Those protocols have been designed that way because those features have been deemed so useful to every potential communication that implementing them separately and paying for the overhead of explicit composition would be wasteful. However, with TCP providing reliable byte streams, and UDP providing unreliable message streams, there is no standard overlay for reliable message streams. The gap is a classic example of the failure of making protocols too big. Alternatives such as SCTP had to be implemented to provide functionality that had already been implemented, such as TCP's reliability, but which came attached to additional undesired features, such as TCP's byte stream abstraction.

Fortunately, there is already a design principle to guide us here: separation of concerns [6]. Two channels should be separate if the interaction between them is small or simple enough that each can be meaningfully studied and advanced in isolation from the other. The information-hiding principle [8] is almost identical, and suggests dividing abstractions based on how well it is possible to hide their implementations.

5. CONCLUSION

> While the components of a complete solution ... have been the subject of extensive research and standardization for more than 10 years, end-to-end signaled QoS for the Internet has not become a reality.
>
> —*Some Internet Structural Guidelines and Philosophy* (RFC 3439) [3]

The future of the Internet is exciting. It is making promises, from an efficient power grid to more accessible health care, and the technology to make it possible is tantalizingly within reach. What distinguishes the new promises from the old ones, however, is that it unless we can justify placing our trust in the solutions, it would be too dangerous to adopt them.

The models we use for reasoning today are limited in scale and composability, and, worse, those limits have resulted in a reality that diverges widely from the model. Drawing on old ideas, such as the channel as the fundamental unit of communication, I have proposed a new model that enables technical solutions to be reused more flexibly and better reasoned about in isolation.

The first step in making the exciting future possible is to talk about dependability and what we expect. Language will be crucial for that discussion, and I have tried to provide some of it in a new way, unencumbered by the old models.

Finally, we will have to start building systems with a thought for justifying their correctness from the beginning. There already exist some valuable design principles that keep us honest in that regard, and I have proposed two new ones, the haggling principle and the composition principle, to try to keep the channel market model in mind.

Providing a language is barely even starting the discussion, but I hope that the next steps will lead us places we can't even imagine yet.

6. REFERENCES

[1] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. on Dependable and Secure Computing*, 1(1), 2004.

[2] R. Braden. Requirements for Internet hosts—communication layers. RFC 1122.

[3] R. Bush and D. Meyer. Some Internet architectural guidelines and philosophy. RFC 3439.

[4] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's internet. *IEEE/ACM Trans. Netw.*, 13(3), June 2005.

[5] E. W. Dijkstra. The humble programmer. *Commun. ACM*, 15(10), 1972. Turing Award lecture.

[6] E. W. Dijkstra. On the role of scientific thought. In *Selected Writings on Computing: A Personal Perspective*, pages 60–66. Springer-Verlag, 1982.

[7] B. W. Lampson. Hints for computer system design. *Operating Systems Review*, 15(5), Oct. 1983.

[8] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12), 1972.

[9] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ToCS*, 2(4), 1984.

[10] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 1948.