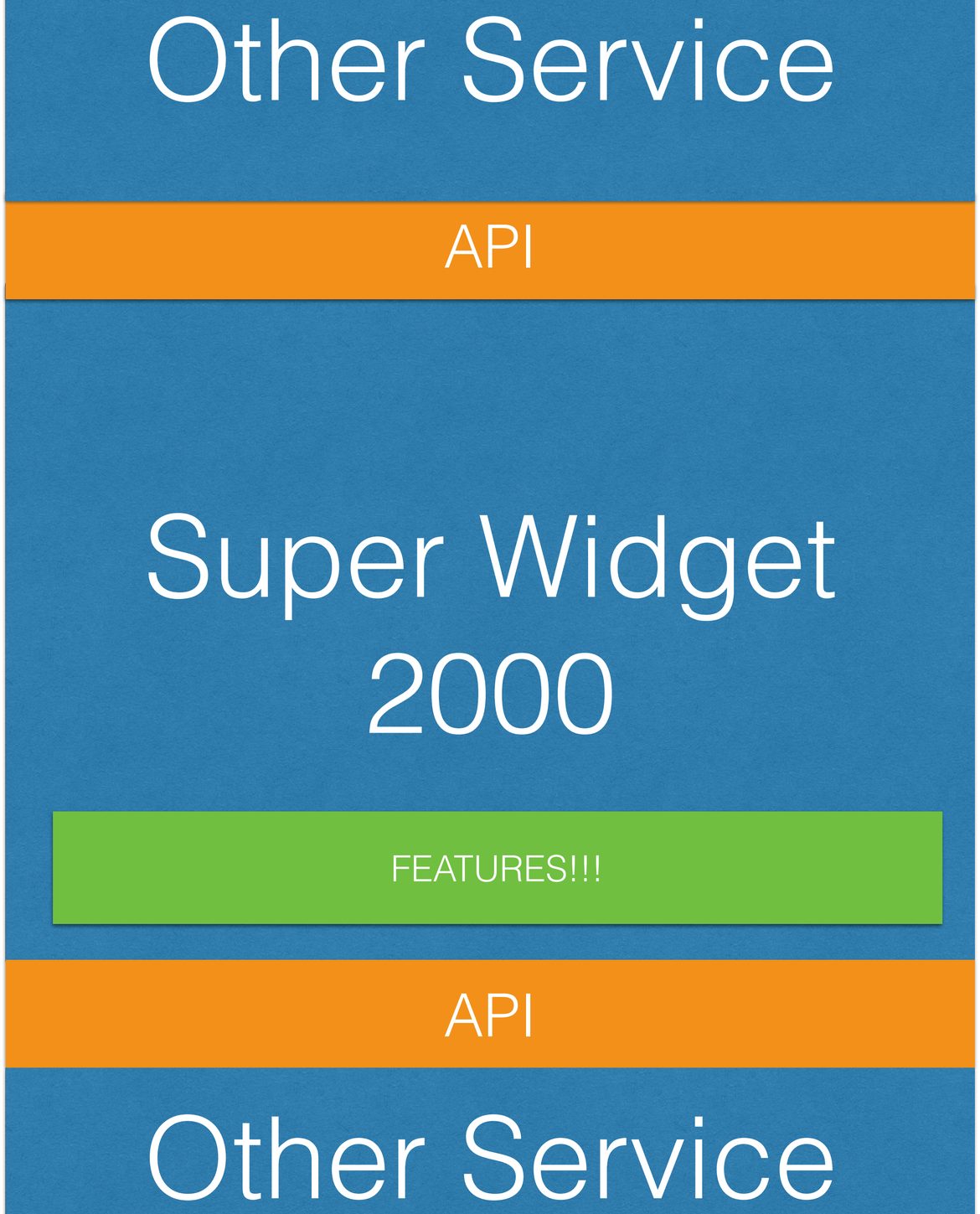
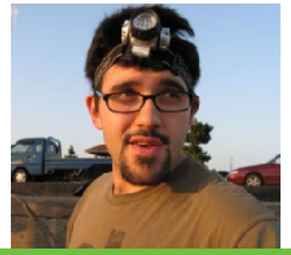
Design Philosophy in Networked Systems

Justine Sherry, Carnegie Mellon University

So you're building a widget.



So you're building a widget.



Add-on Extension

Other Service

Super Widget 2000

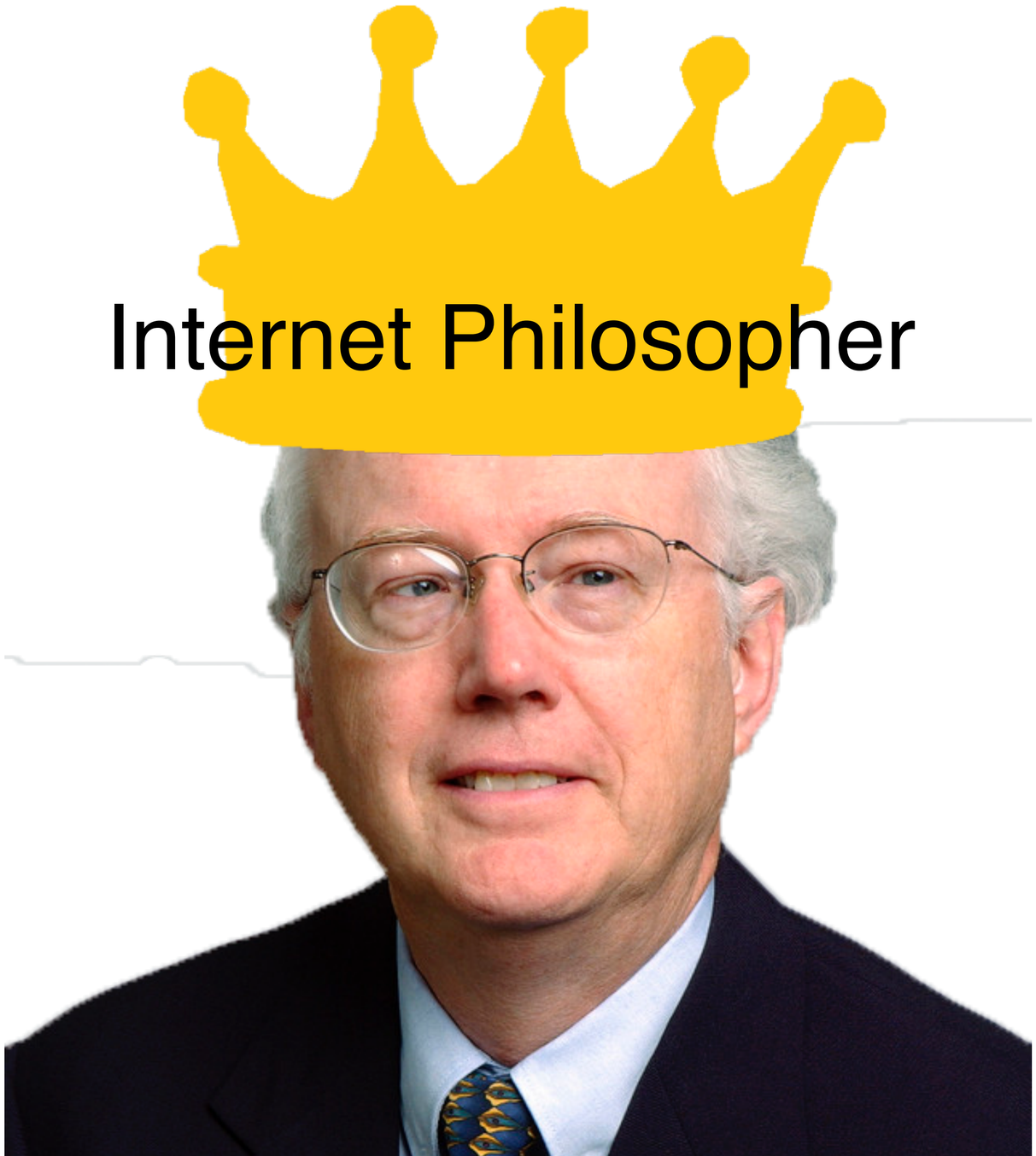
Other Other Service

Other Other Other Service

Platform X

Platform z

Design Philosophy:
Principles for thinking about the outside world and the impact of your widget.



This is
David Clark.

Three Papers

"END-TO-END ARGUMENTS IN SYSTEM DESIGN", Salzer, Reed and **Clark**.

"Tussle in Cyberspace: Defining Tomorrow's Internet",
Clark, Wrokowski, Sollins, and Braden.

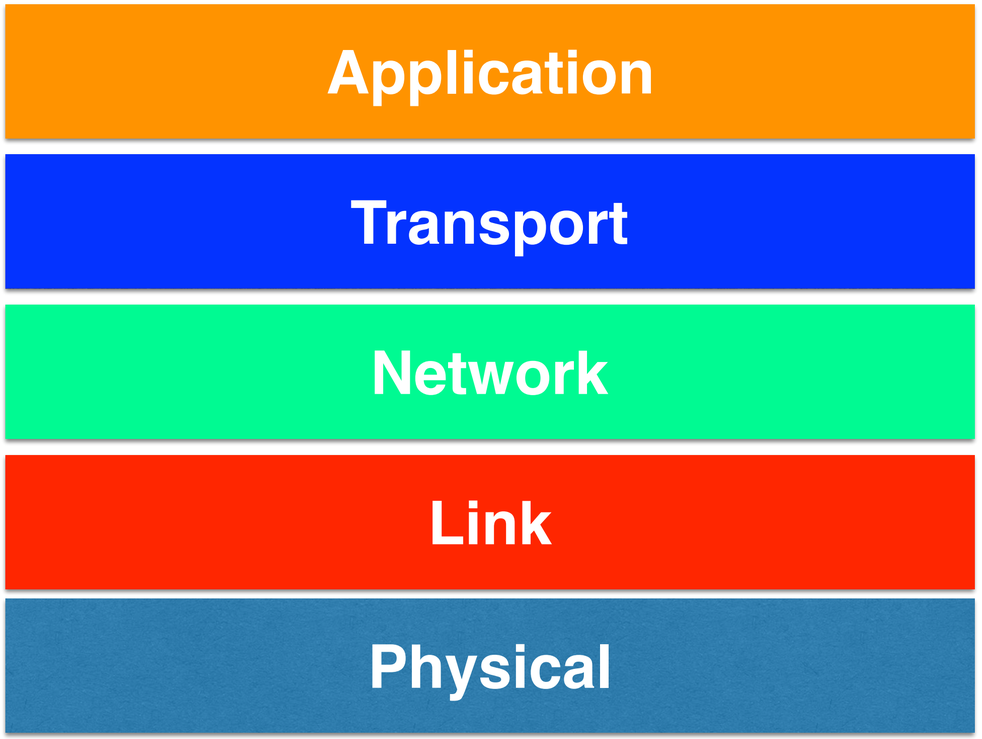
"The Design Philosophy of the DARPA Internet Protocols", **Clark, D.**

I met David Clark at a conference once.
He told me he liked my talk.
I fangirled only a little tiny bit.


Well, this has been sufficiently awkward.

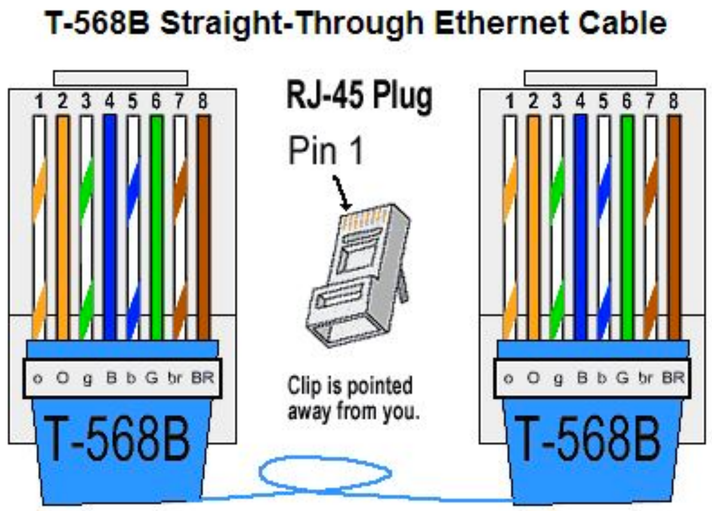
Layered Design: Background

Layers…



"How do I make bits out of electricity or light or sound or radio?"

Physical



Application

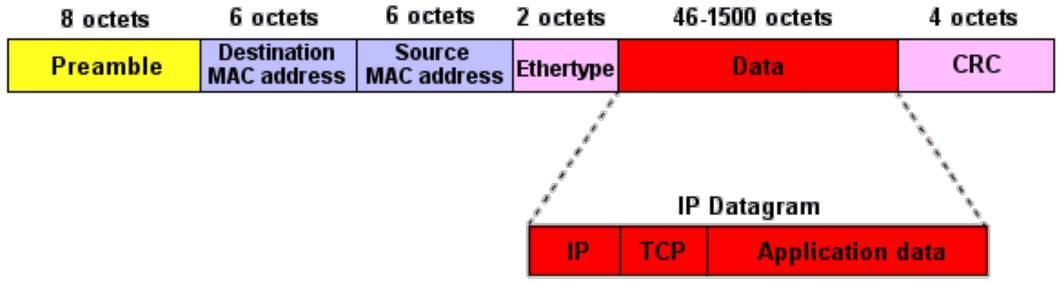
Transport

Network

Link

How do I turn those bits in to messages?

Link



Application

Transport

Network

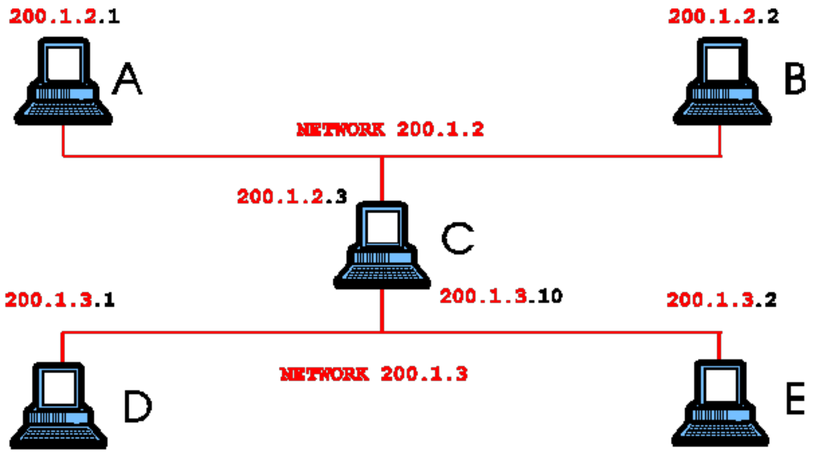
Physical

How do I deliver the bits across the Internet?

Application

Transport

Network

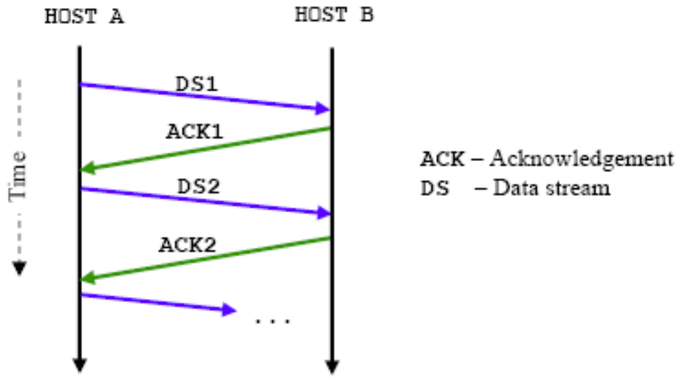


Link

Physical

How do I make sure none of the bits got lost, or corrupted? And that they show up in order?

Transport



Application

Network

Link

Physical

How do I interpret the bits to do cool stuff?

Application

Transport

Network

Link

Physical

hey zeeshan check out this sweet paper!

Application
Transport
Network
Link
Physical

Application
Transport
Network
Link
Physical

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

...
10101010
11110110
10000011
01010111

Application

Transport

Network

Link

Physical

label all the pieces so none go missing

Application

Transport

Network

Link

Physical

1 2 3

Application

Transport

Network

Link

Physical

put an address on my envelopes/packets

Application
Transport
Network
Link
Physical

1 2 3

Application
Transport
Network
Link
Physical

and send those packets down to bits

Application
Transport
Network
Link
Physical

1 2 3

To: Z To: Z To: Z

Application
Transport
Network
Link
Physical

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

1 2 3

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

Yep that's me

1 2 3

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

Did we get all the pieces?

1 2 3

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

...
10101010
11110110
10000011
01010111

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

Application

Transport

Network

Link

Physical

sweet paper, thx justine

Application
Transport
Network
Link
Physical

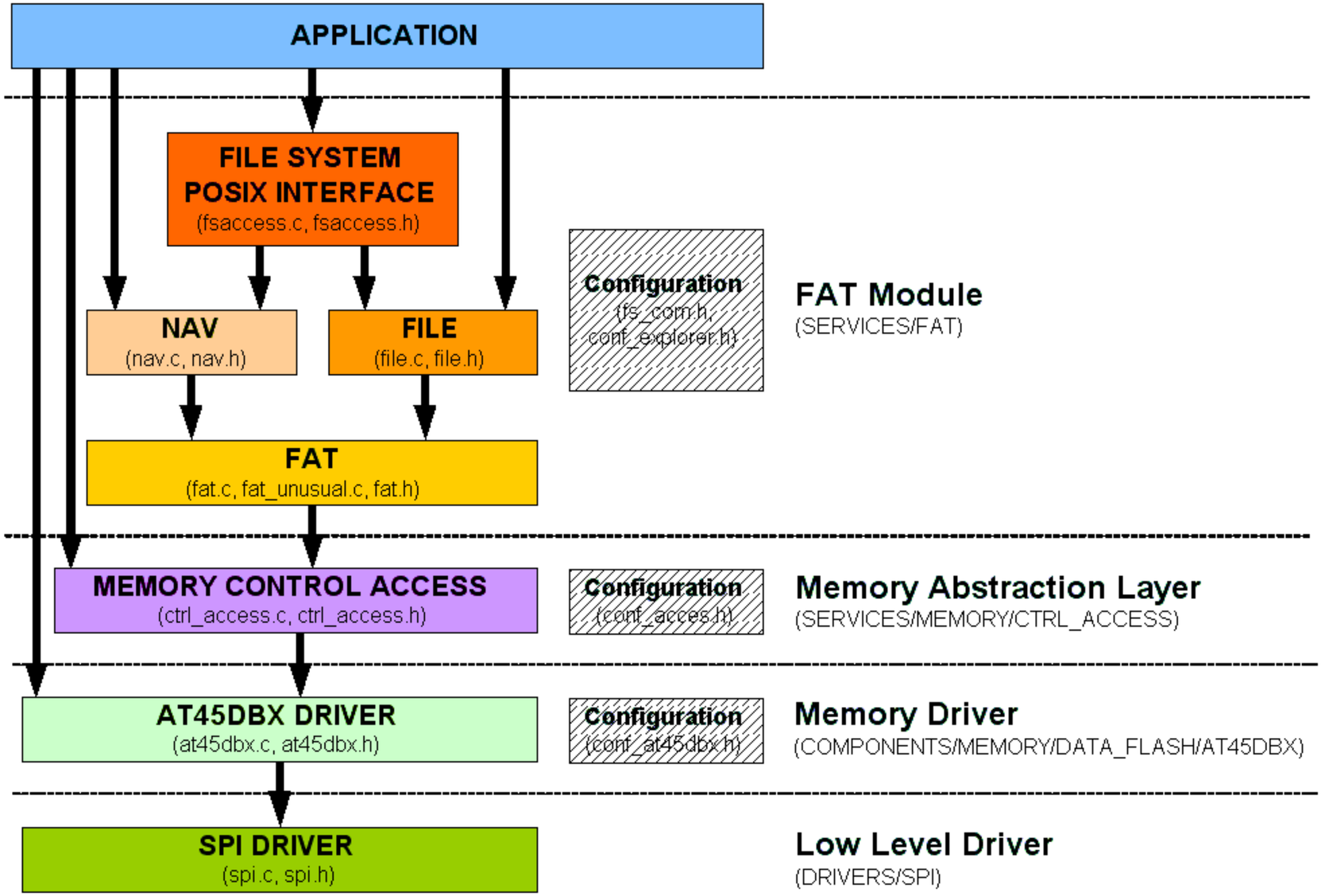
Application
Transport
Network
Link
Physical

Modular, Layered Systems

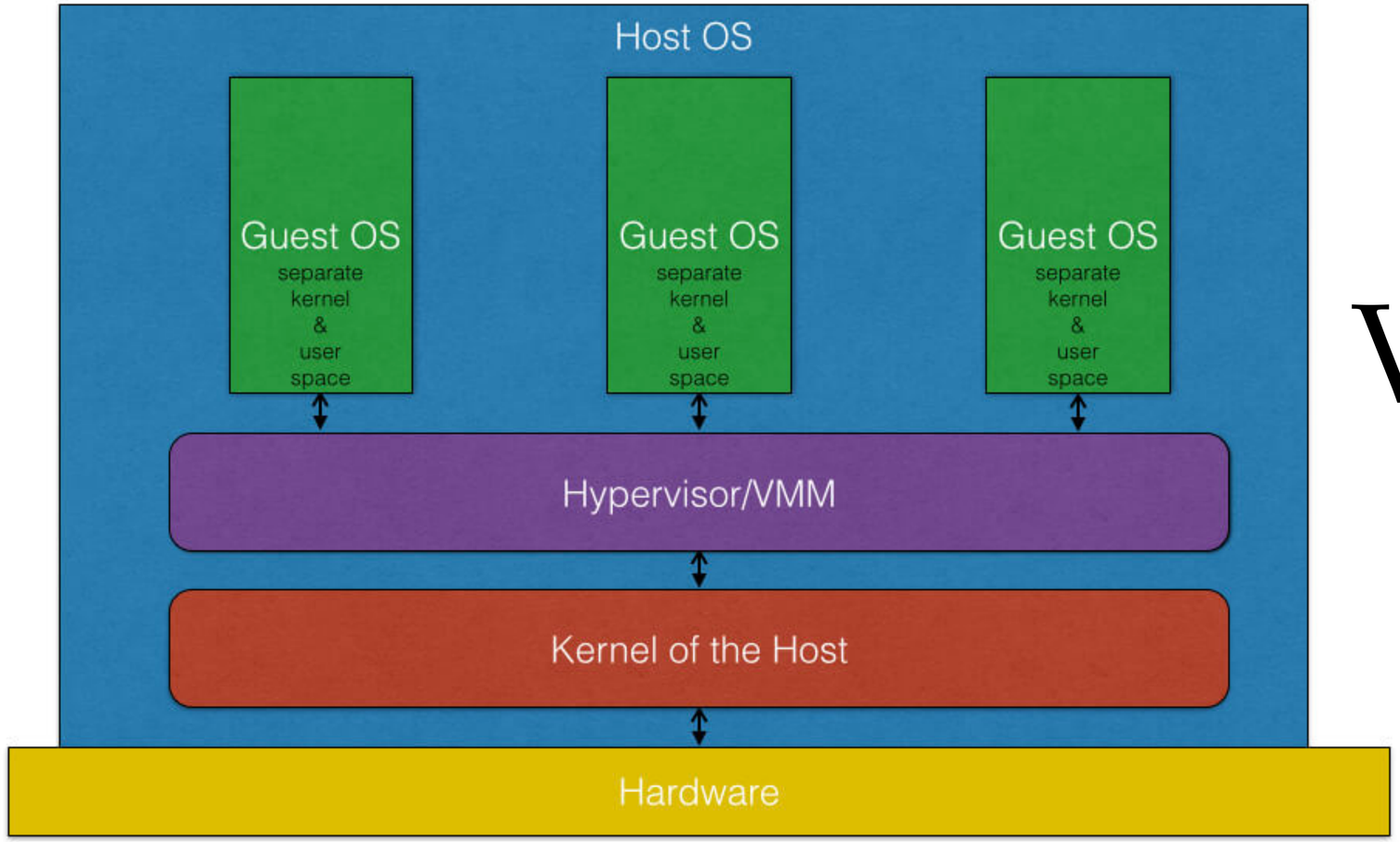
- Components are implemented independently with APIs in between them.
- Different layers may have multiple implementations (e.g. WiFi vs Wired Ethernet as your physical layer)
  - … and even different APIs and capabilities (e.g. TCP provides reliable, in-order delivery but UDP does not provide reliability or ordering. Both are transport protocols!)
- To make the end-to-end system work, data passes through multiple layers.

Networks aren’t the only systems with layers.

Storage system layers…



Virtualization Layers

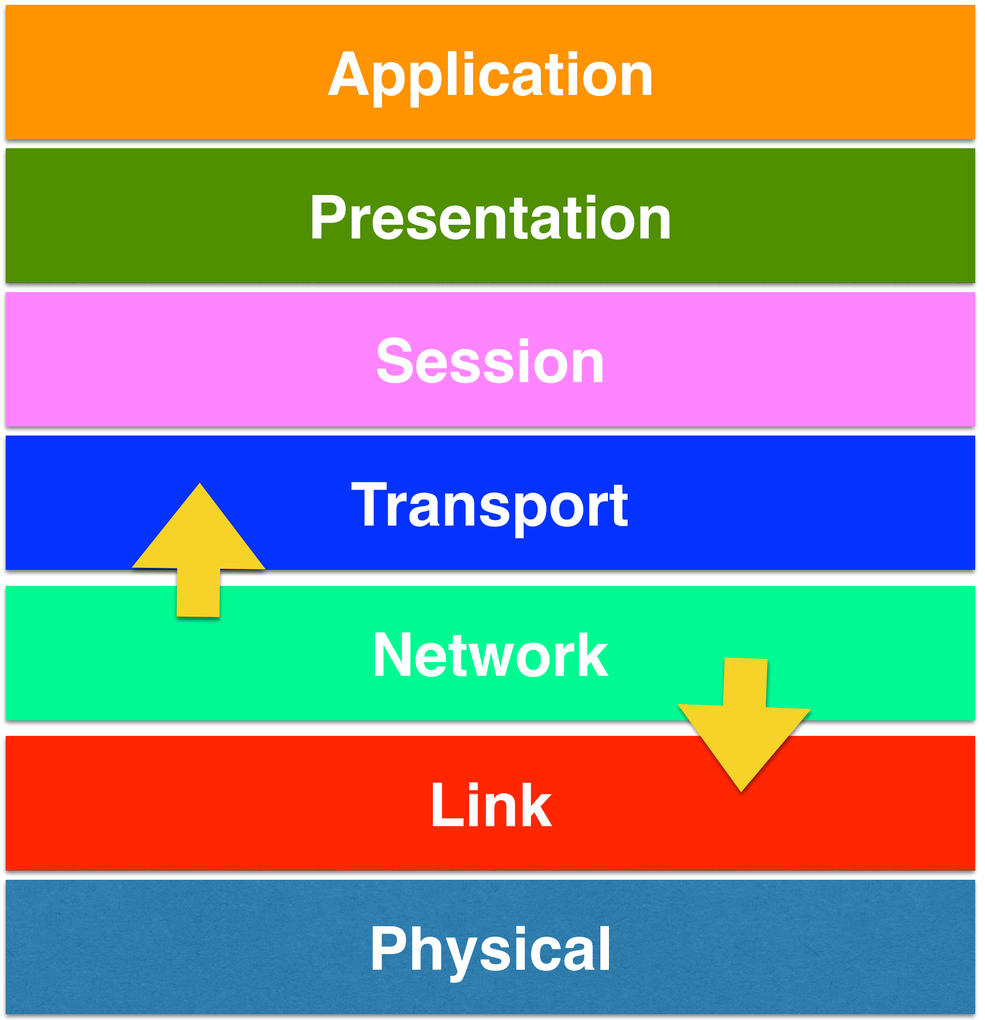


Hypervisor based Virtualization

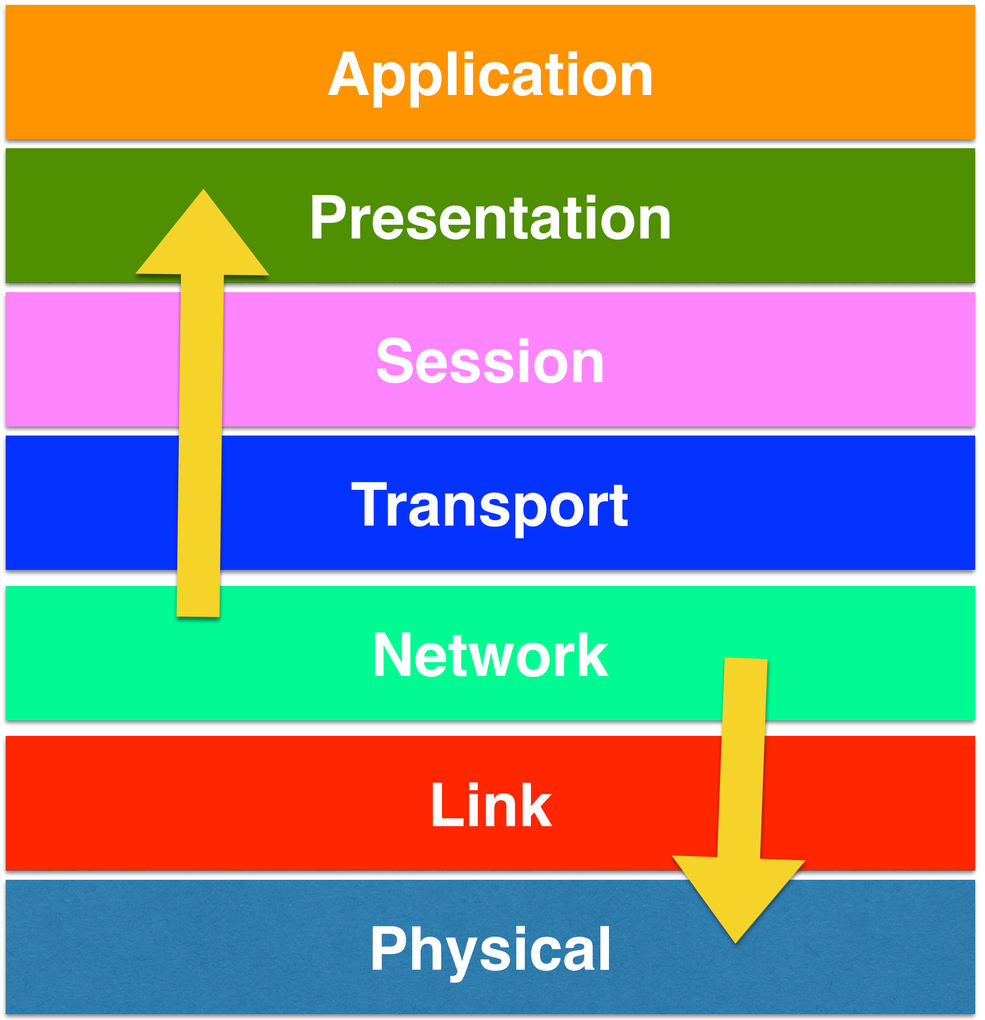
Today’s questions!

How does what I implement in my layer impact the outside world?

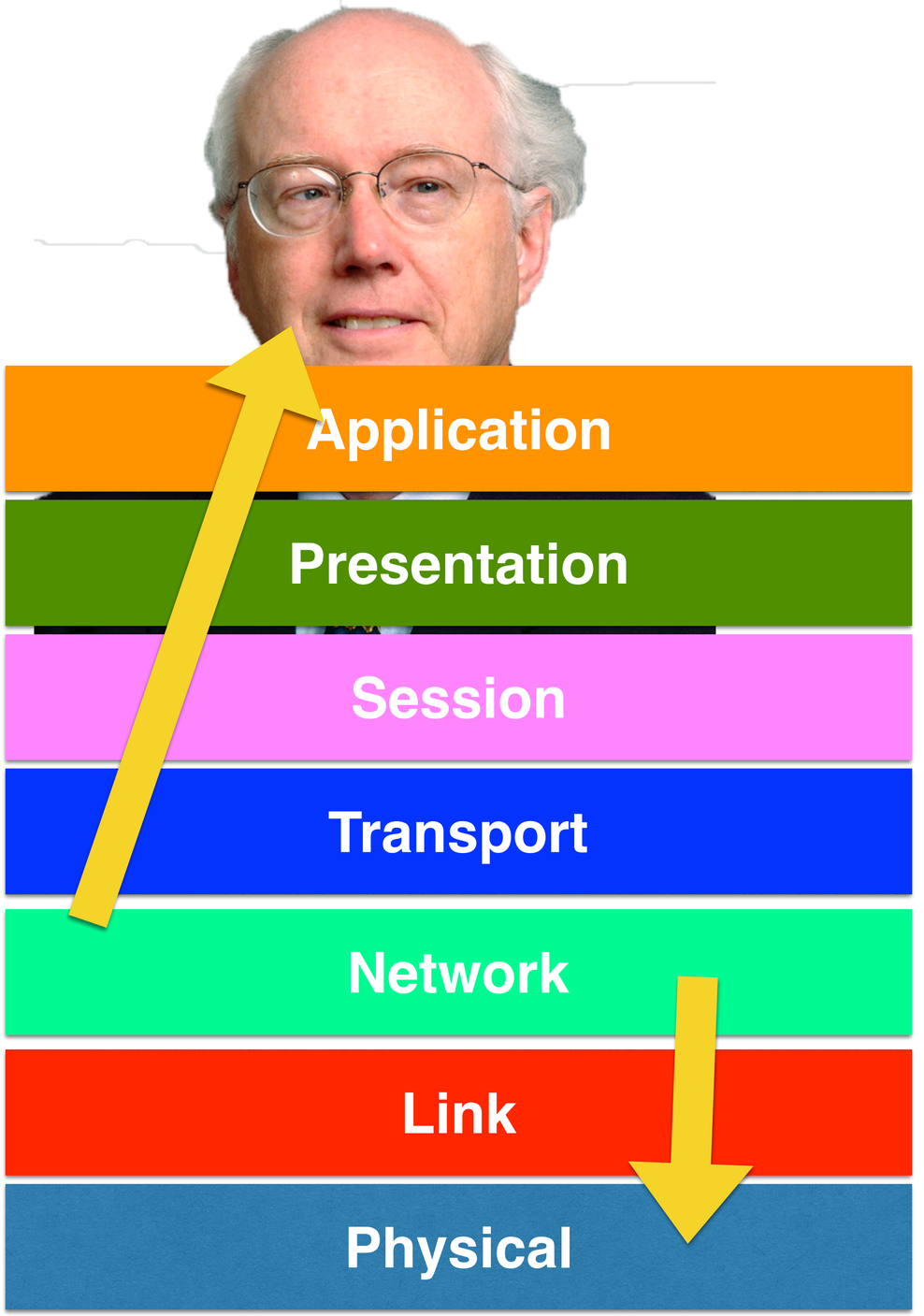
“Sure, I think about that when I design APIs!”



What about all the other layers?



What about the people on top?



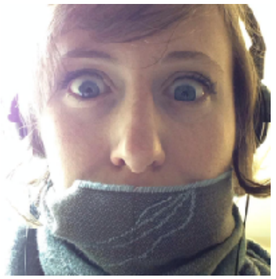
Without further ado, paper #1:

J. H. Saltzer, D. P. Reed, and D. D. Clark. 1984. End-to-end arguments in system design. ACM Trans. Comput. Syst. 2, 4 (November 1984), 277-288.

“Careful File Transfer”

At host A the file transfer program calls upon the file system to read the file from the disk, where it resides on several tracks, and the file system passes it to the file transfer program in fixed-size blocks chosen to be disk-format independent.

A



File System

Program

Network

B

File System

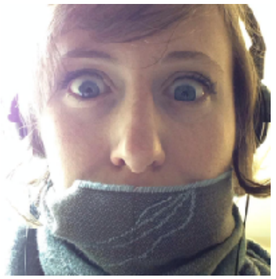
Program

Network

“Careful File Transfer”

At host A the file transfer program calls upon the file system to read the file from the disk, where it resides on several tracks, and the file system passes it to the file transfer program in fixed-size blocks chosen to be disk-format independent.

A



File System

Program

Network

B

File System

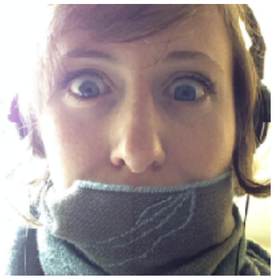
Program

Network

“Careful File Transfer”

At host A the file transfer program calls upon the file system to read the file from the disk, where it resides on several tracks, and the file system passes it to the file transfer program in fixed-size blocks chosen to be disk-format independent.

A



File System

Program

Network

B

File System

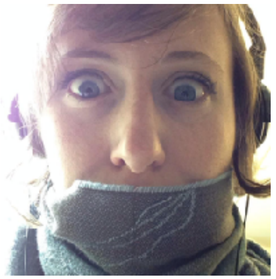
Program

Network

“Careful File Transfer”

Also at host A the file transfer program asks the data communication system to transmit the file using some communication protocol that involves splitting the data into packets. The packet size is typically different from the file block size and the disk track size.

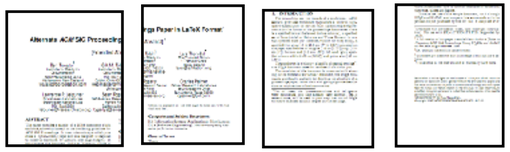
A



File System

Program

Network



B

File System

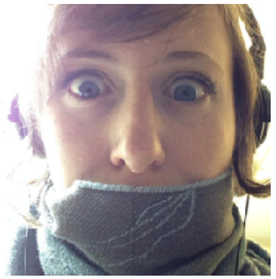
Program

Network

“Careful File Transfer”

Also at host A the file transfer program asks the data communication system to transmit the file using some communication protocol that involves splitting the data into packets. The packet size is typically different from the file block size and the disk track size.

A



File System

Program

Network

B

File System

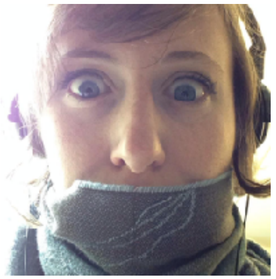
Program

Network

“Careful File Transfer”

The data communication network moves the packets from computer A to computer B.

A



B

File System

Program

Network

File System

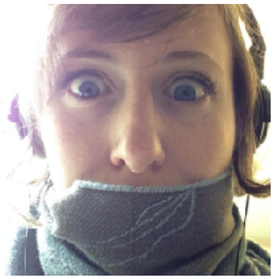
Program

Network

“Careful File Transfer”

At host B a data communication program removes the packets from the data communication protocol and hands the contained data on to a second part of the file transfer application, the part that operates within host B.

A



File System

Program

Network

B

File System

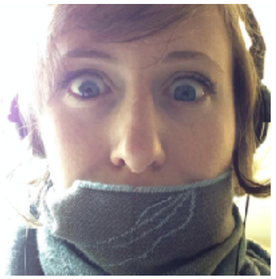
Program

Network

“Careful File Transfer”

At host B, the file transfer program asks the file system to write the received data on the disk of host B.

A



File System

Program

Network

B

File System

Program

Network

What if Zeeshan later reads the file and find it is corrupted? What could have gone wrong?

The file, though originally written correctly onto the disk at host A, if read now may contain incorrect data, perhaps because of hardware faults in the disk storage system.

The software of the file system, the file transfer program, or the data communication system might make a mistake in buffering and copying the data of the file, either at host A or host B.

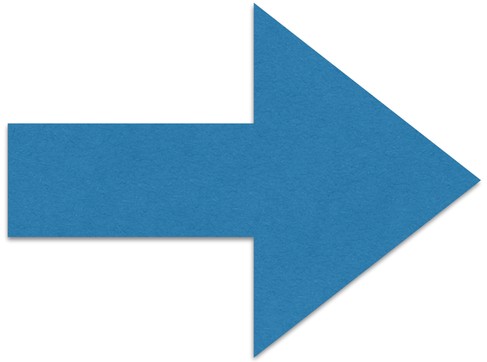
The hardware processor or its local memory might have a transient error while doing the buffering and copying, either at host A or host B.

The communication system might drop or change the bits in a packet, or lose a packet or deliver a packet more than once.

Either of the hosts may crash part way through the transaction after performing an unknown amount (perhaps all) of the transaction.

How do we re-design our system to make sure the file doesn’t get corrupted?

B

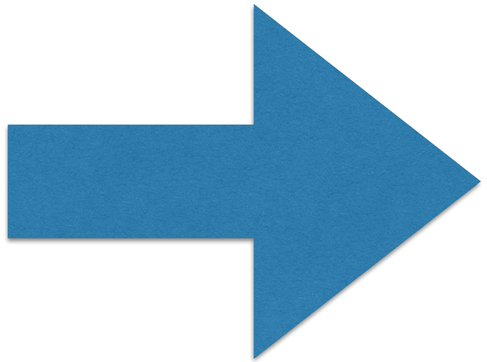


File System

Program

Network

B

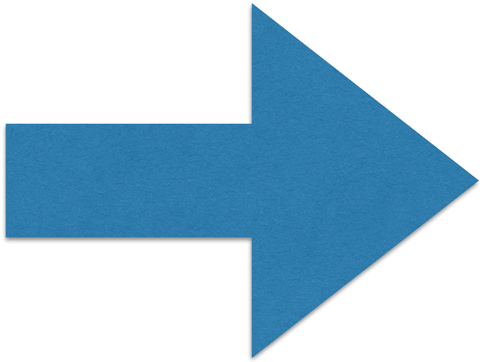


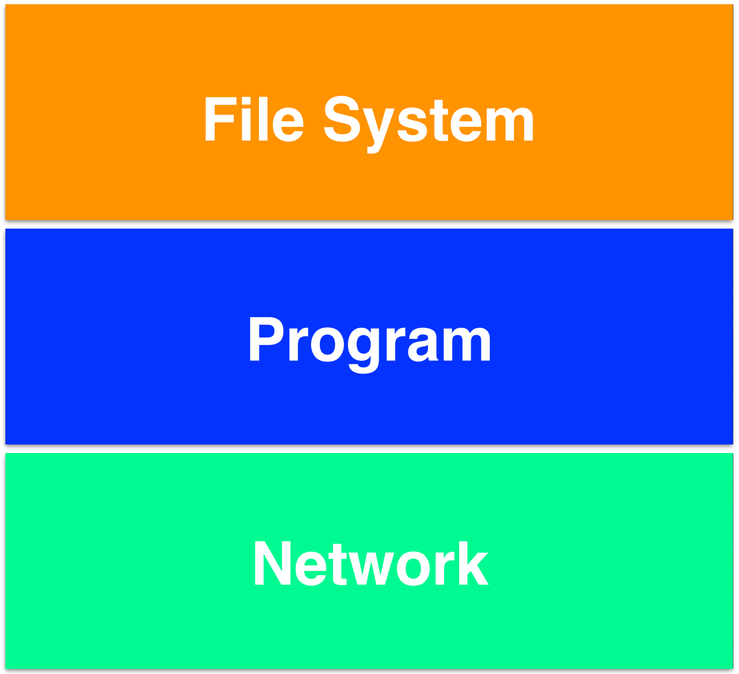
File System

Program

Network

B





The End-to-End Argument

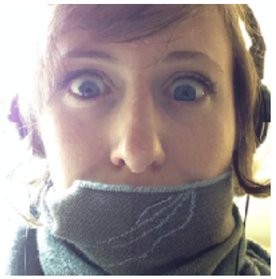
[If] the function in question can completely and correctly be implemented with the knowledge and help of the application standing at the endpoints of the communication system:

[Then] providing that questioned function as a feature of the communication system [or lower layer] is not possible.

[However], sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.

Let's say we had a perfectly reliable network

A



B

Would that solve our reliability problem?

The file, though originally written correctly onto the disk at host A, if read now may contain incorrect data, perhaps because of hardware faults in the disk storage system.

The software of the file system, the file transfer program or the data communication system might make a mistake in buffering and copying the data of the file, either at host A or host B.

The hardware processor or its local memory might have a transient error while doing the buffering and copying, either at host A or host B.

The communication system might drop or change the bits in a packet, or lose a packet or deliver a packet more than once.

Either of the hosts may crash part way through the transaction after performing an unknown amount (perhaps all) of the transaction.

Would that solve our reliability problem?

The file, though originally written correctly onto the disk at host A, if read now may contain incorrect data, perhaps because of hardware faults in the disk storage system.

The software of the file system, the file transfer program or the data communication system might make a mistake in buffering and copying the data of the file, either at host A or host B.

The hardware processor or its local memory might have a transient error while doing the buffering and copying, either at host A or host B.

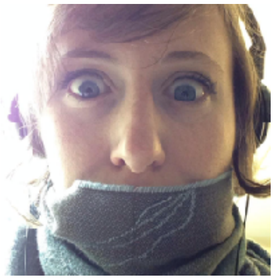
The communication system might drop or change the bits in a packet, or lose a packet or deliver a packet more than once.

Either of the hosts may crash part way through the transaction after performing an unknown amount (perhaps all) of the transaction.

Well, that wasn’t very helpful…

“End to End Check and Retry”

A



File System

Program

Network

B

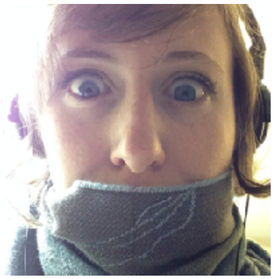
File System

Program

Network

“End to End Check and Retry”

A



Read file and its checksum from disk.
Verify file + checksum.
Send File AND Checksum.

File System

Program

Network

B

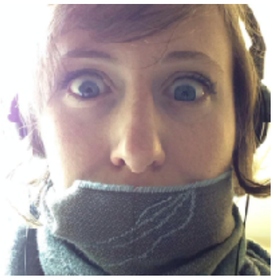
File System

Program

Network

“End to End Check and Retry”

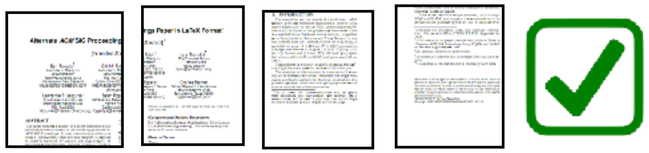
A



File System

Program

Network



B

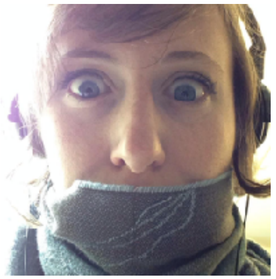
File System

Program

Network

“End to End Check and Retry”

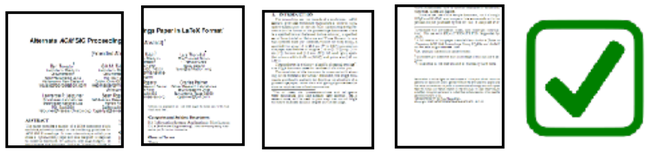
A



File System

Program

Network



B

File System

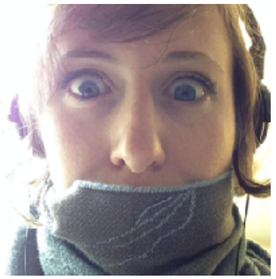
Program

Network

“Careful File Transfer”

The data communication network moves the packets from computer A to computer B.

A



B

File System

Program

Network

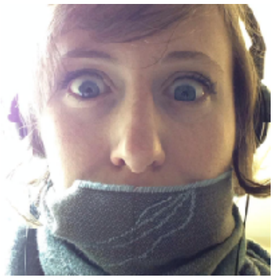
File System

Program

Network

“End to End Check and Retry”

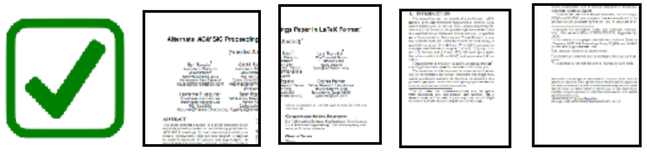
A



File System

Program

Network



B

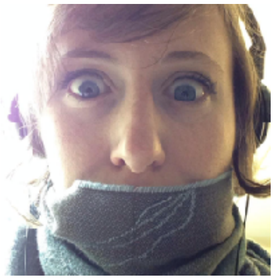
File System

Program

Network

“End to End Check and Retry”

A



File System

Program

Network

Write file and checksum to disk.
Then read back and double-check that checksum + file verify.

B

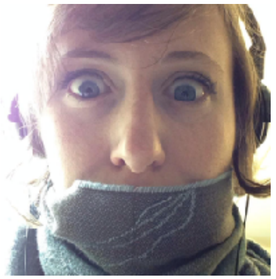
File System

Program

Network

“End to End Check and Retry”

A



File System

Program

Network

If Checksum doesn’t match?
Just ask Justine to re-send.
(ie, try all over again!)

B

File System

Program

Network

Would that solve our reliability problem?

The file, though originally written correctly onto the disk at host A, if read now may contain incorrect data, perhaps because of hardware faults in the disk storage system.

The software of the file system, the file transfer program, or the data communication system might make a mistake in buffering and copying the data of the file, either at host A or host B.

The hardware processor or its local memory might have a transient error while doing the buffering and copying, either at host A or host B.

The communication system might drop or change the bits in a packet, or lose a packet or deliver a packet more than once.

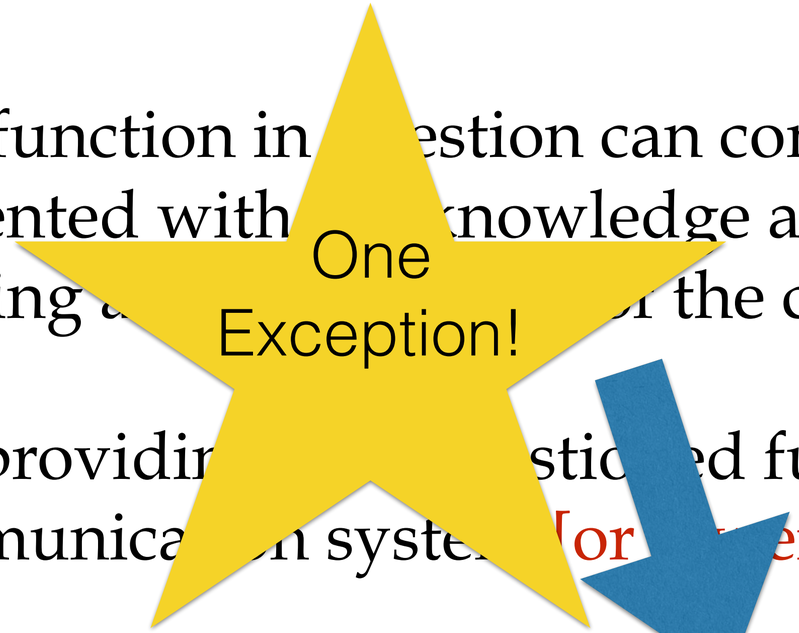
Either of the hosts may crash part way through the transaction after performing an unknown amount (perhaps all) of the transaction.

Lesson: If you can do it at the "higher" layer, don't bother implementing it at a lower layer.

Other places to apply E2E in Networks

- Encryption
- First-in-first-out ordering
- Duplicate message surpression
- Multi-message transactions

The End-to-End Argument



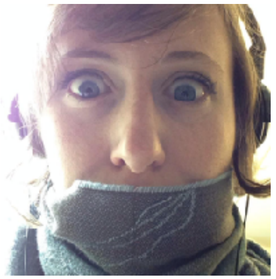
[If] the function in estion can completely and correctly be implemented with knowledge and help of the application standing a the communication system:

[Then] providi sti ed function as a feature of the communica n syste [or er layer] is not possible.

[However], sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.

What if 90% of my loss really was happening at the network layer?

A



File System

Program

Network

As a performance optimization, you might want to implement it in the lower layer anyway (redundantly).

B

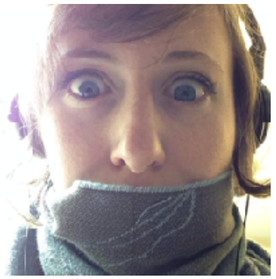
File System

Program

Network

“End to End Check and Retry” + A Reliable Network

A



B

Anyone have any other examples where this plays out?

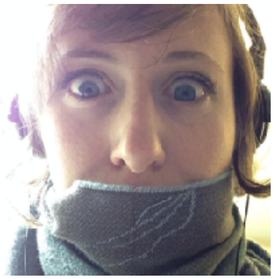
Why the heck is this the classic article networking grad students argue about?

The “Strong” End-to-End Argument

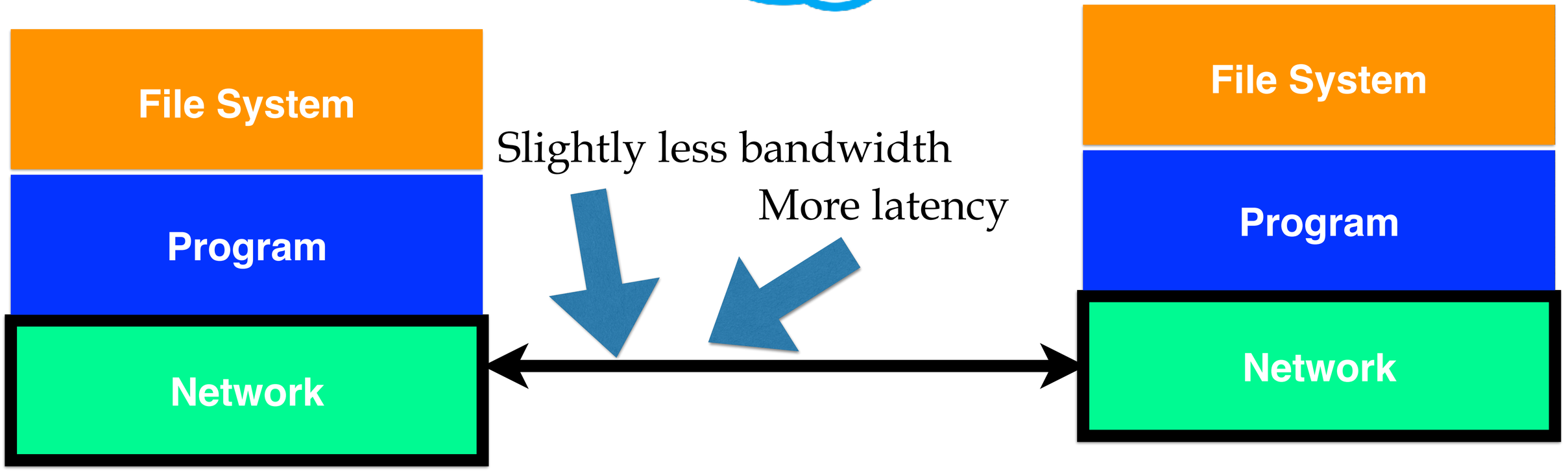
It’s not just a waste of time to put non-essential functionality at lower layers: *it’s actually harmful*.

“End to End Check and Retry” + A Reliable Network

A

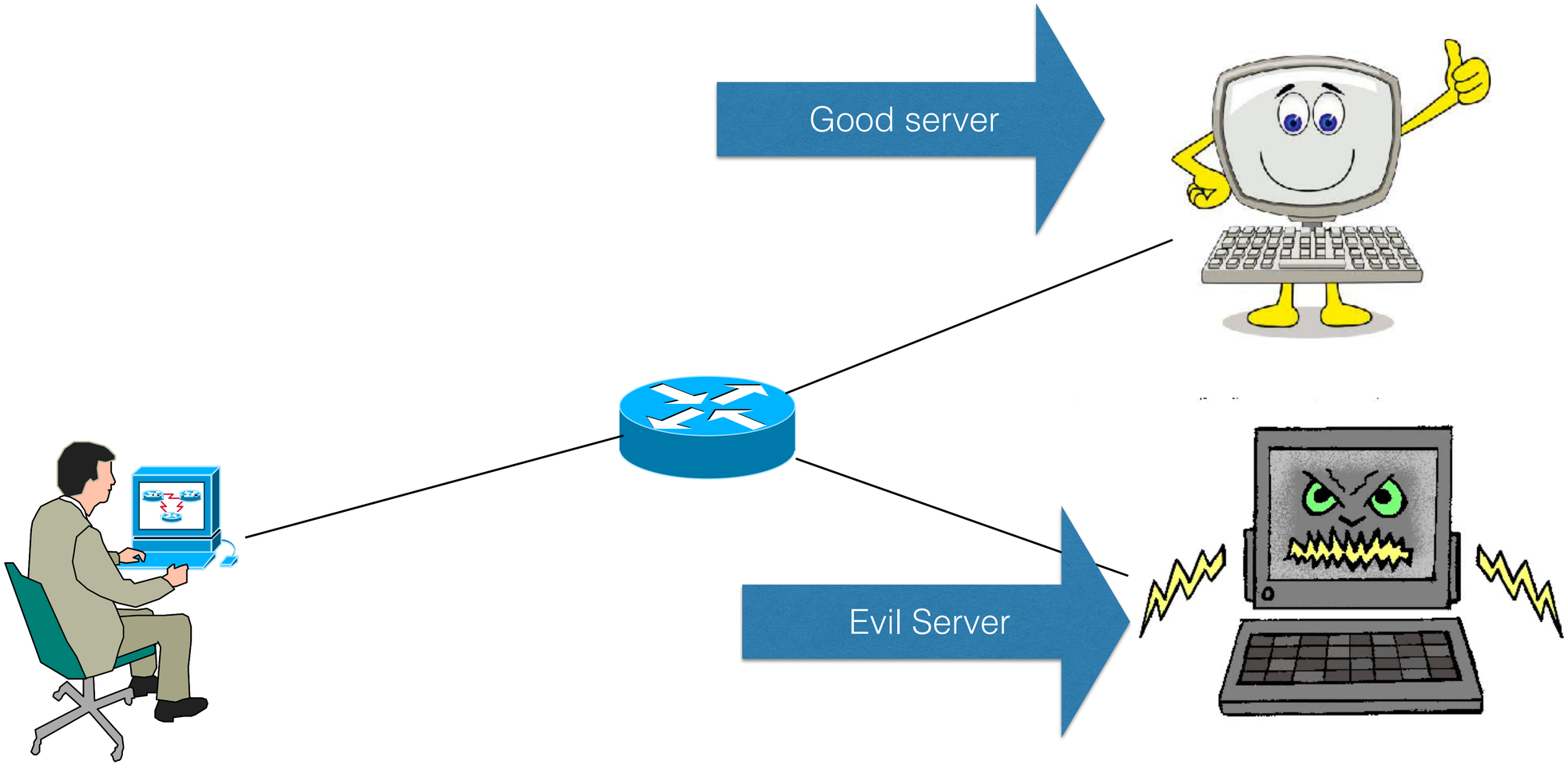


B

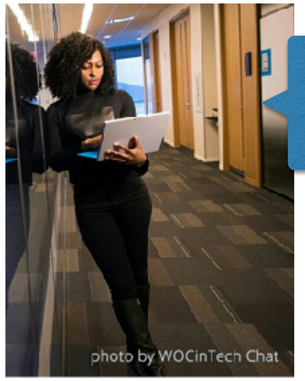


Some applications may be *constrained* by the new functionality.

Firewalls and Intrusion Detection



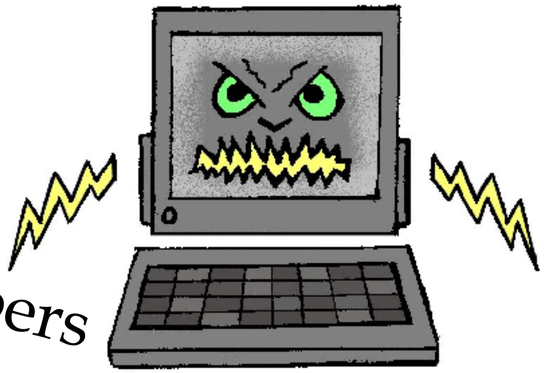
Firewalls and Intrusion Detection



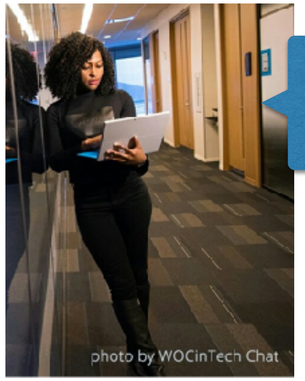
I need to protect my users!

Web traffic, email

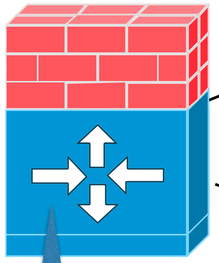
IRC, strange port numbers



Firewalls and Intrusion Detection

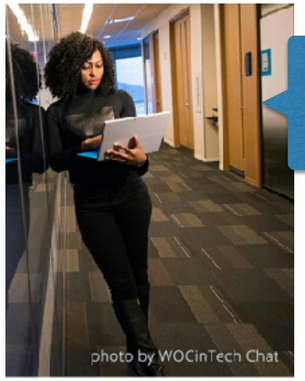


I need to protect my users!

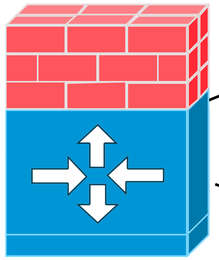


Only allow web and email!

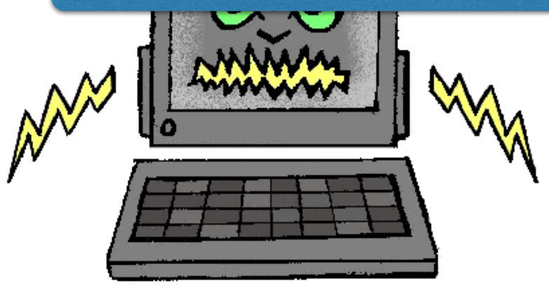
Firewalls and Intrusion Detection



I need to protect my users!



But what if I have a cool new app?



HTTP for Everything?

HTTP as the Narrow Waist of the Future Internet

Lucian Popa
U.C. Berkeley / ICSI

Ali Ghodsi
U.C. Berkeley

Ion Stoica
U.C. Berkeley

ABSTRACT

Over the past decade a variety of network architectures have been proposed to address IP's limitations in terms of flexible forwarding, security, and data distribution. Meanwhile, fueled by the explosive growth of video traffic and HTTP infrastructure (*e.g.*, CDNs, web caches), HTTP has became the de-facto protocol for deploying new services and applications. Given these developments, we argue that these architectures should be evaluated not only with respect to IP, but also with respect to HTTP, and that HTTP could be a fertile ground (more so than IP) for deploying the newly proposed functionalities. In this paper, we take a step in this direction, and find that HTTP already provides many of the desired properties for new Internet architectures. HTTP is a content centric protocol, provides middlebox support in the form of reverse and forward proxies, and leverages DNS to decouple names from addresses. We then investigate HTTP's limitations, and propose an extension, called S-GET that provides support for low-latency applications, such as VoIP and chat.

1. INTRODUCTION

During the past decade, a plethora of new Internet architectures have been proposed to address the shortcomings of IP in terms of flexibility, scale, and security (*e.g.,* [10, 20, 27, 28, 31, 42, 47, 49]). Some of these limitations have been traced to IP's inability to decouple the concepts of address and identity, its lack of explicit support for middleboxes, mobility, and content distribution.

Meanwhile, industry has been pushing through changes that are having a profound impact on the Internet. In particular, we are witnessing an explosive growth of HTTP traf-

In this paper, we take this trend to its logical conclusion and consider the scenario where HTTP becomes the de facto "narrow waist" of the Internet—that is, the vast majority of traffic runs over HTTP instead of directly over IP, and HTTP itself might run on top of network layers other than IP.

Given such scenario, we argue that we should start evaluating HTTP with respect to the existing Internet architecture proposals, and, in the process, answer the following questions: What are the properties aimed by these architectures that HTTP already provides, and what are the ones it does not? What are HTTP's main drawbacks? Can these drawbacks be addressed by extending HTTP, or are they the result of fundamental limitations of HTTP?

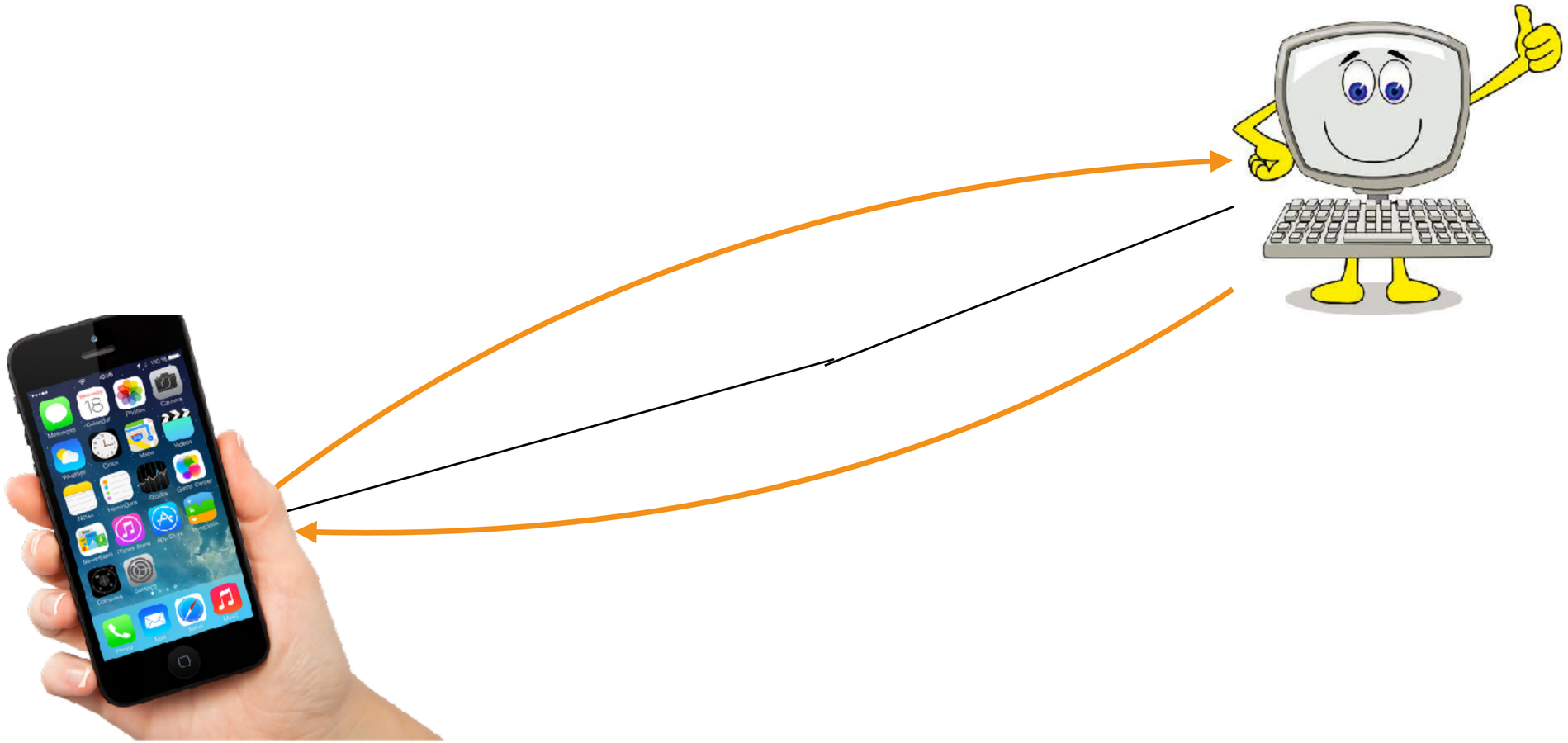
In this paper, we find that HTTP addresses many of the limitations of IP, limitations which have been the target of several recently proposed network architectures. First, HTTP is a content-centric protocol, as each HTTP method specifies the name of the resource (content) it operates on. This allows proxies along a request's path to cache the content, or to redirect the request to the closest or least loaded server storing a copy of the content. Building a content-centric network, albeit at the network layer, has been one of the main goals of recently proposed architectures such as DONA [28] and CCN [27].

Second, HTTP supports both reverse and forward proxies [3, 16]. This allows senders and receivers to add middleboxes on the data path, functionality proposed by many to be incorporated in the Internet [10, 20, 28, 42, 47].

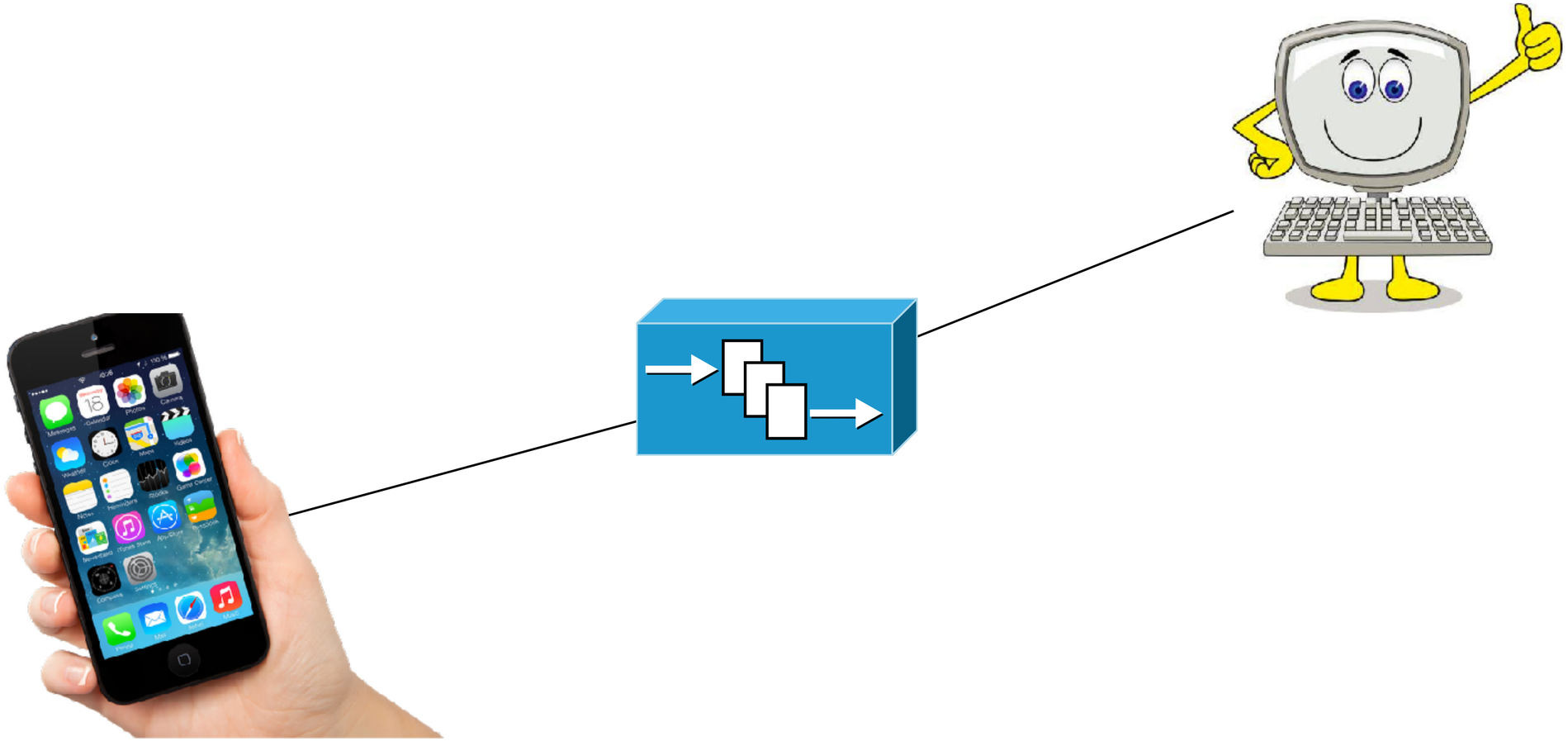
Third, HTTP uses DNS names to refer to content. This enables data "mobility" in the context of a DNS name, and basic anycast functionality via DNS round-robin or modified DNS resolution (typically done by CDNs).

"Many firewalls and intrusion detection systems only accept HTTP. So, we should build everything using HTTP now."

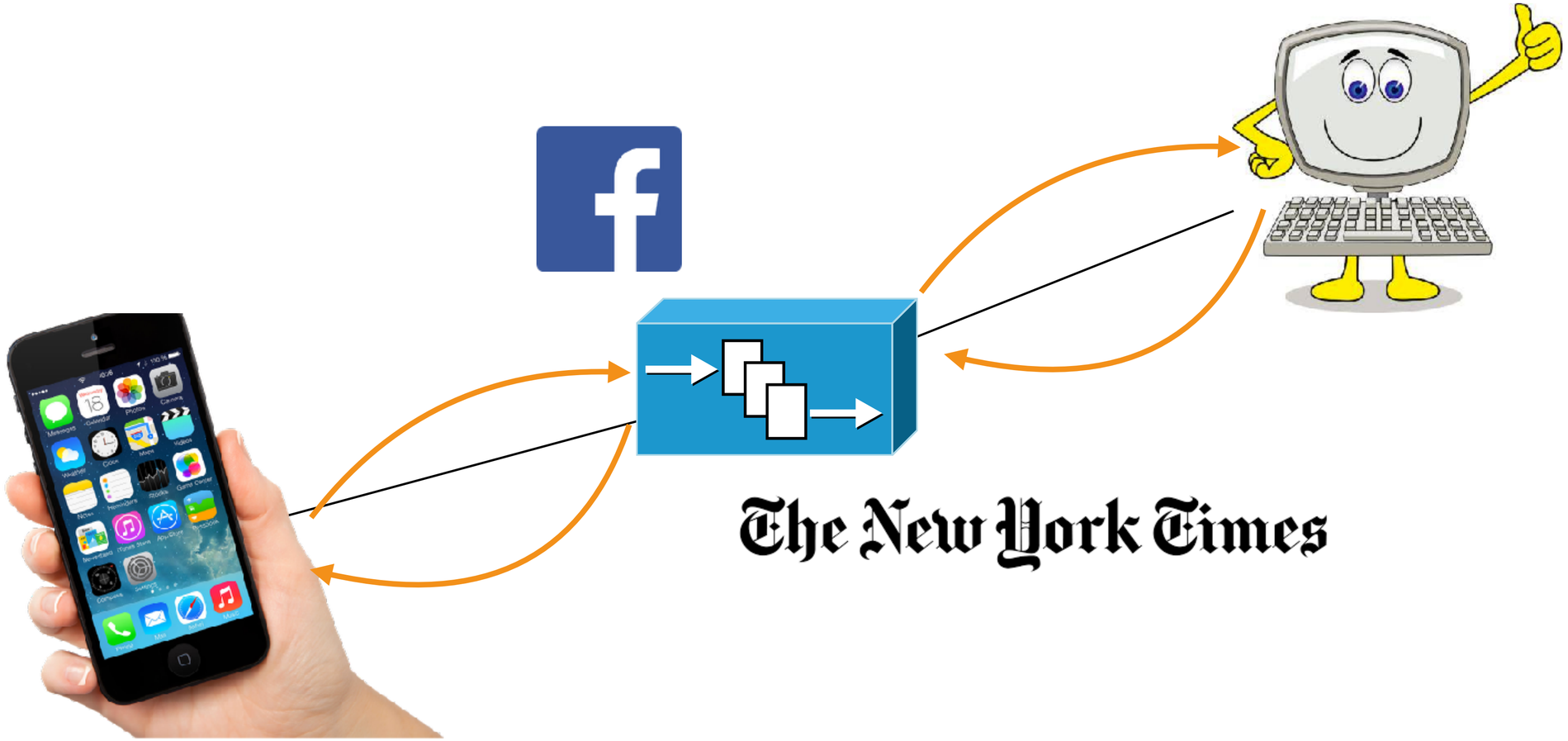
Cache + TCP Optimization



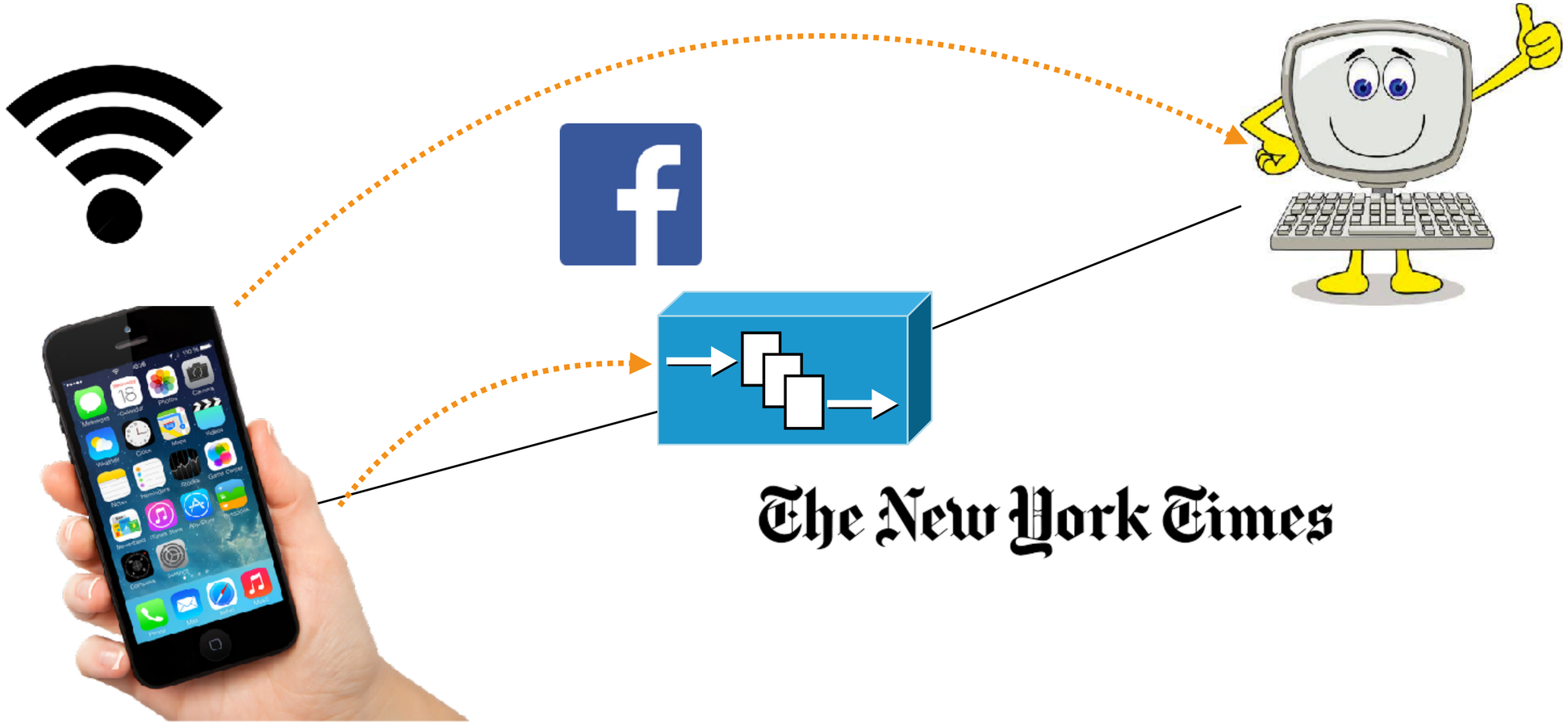
Cache + TCP Optimization



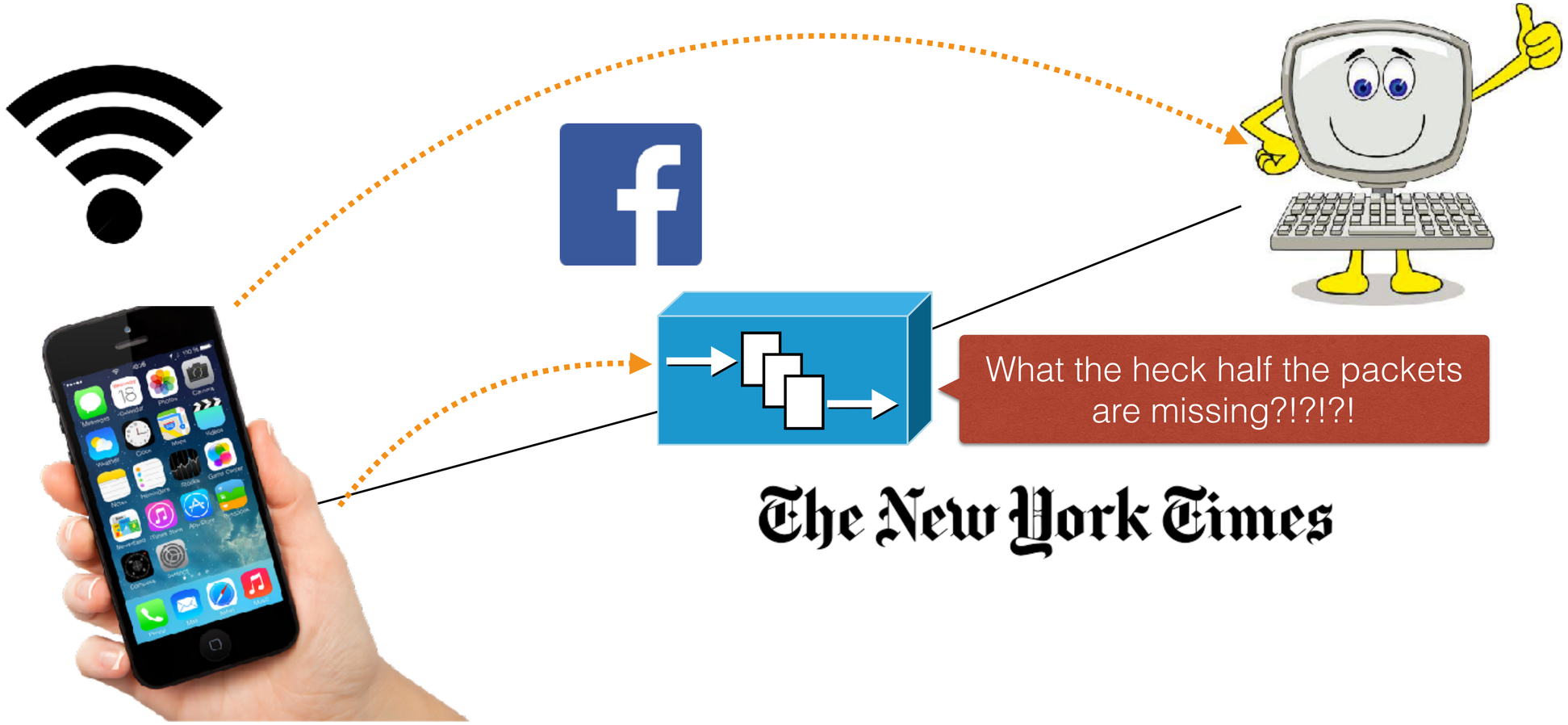
Cache + TCP Optimization



Cache + TCP Optimization



Cache + TCP Optimization



Can we extend existing protocols?

Is it Still Possible to Extend TCP?

Michio Honda*, Yoshifumi Nishida*, Costin Raiciu†, Adam Greenhalgh‡,
Mark Handley‡, Hideyuki Tokuda*
Keio University*, Universitatea Politehnica Bucuresti†, University College London‡
{micchie,nishida}@sfc.wide.ad.jp, costin.raiciu@cs.pub.ro
{a.greenhalgh,m.handley}@cs.ucl.ac.uk, hxt@ht.sfc.keio.ac.jp

ABSTRACT

We've known for a while that the Internet has ossified as a result of the race to optimize existing applications or enhance security. NATs, performance-enhancing-proxies, firewalls and traffic normalizers are only a few of the middleboxes that are deployed in the network and look beyond the IP header to do their job. IP itself can't be extended because "IP options are not an option" [10]. Is the same true for TCP?

In this paper we develop a measurement methodology for evaluating middlebox behavior relating to TCP extensions and present the results of measurements conducted from multiple vantage points. The short answer is that we can still extend TCP, but extensions' design is very constrained as it needs to take into account prevalent middlebox behaviors. For instance, absolute sequence numbers cannot be embedded in options, as middleboxes can rewrite ISN and preserve undefined options. Sequence numbering also must be consistent for a TCP connection, because many middleboxes only allow through contiguous flows.

We used these findings to analyze three proposed extensions to TCP. We find that MPTCP is likely to work correctly in the Internet or fallback to regular TCP. TcpCrypt seems ready to be deployed, however it is fragile if resegmentation does happen—for instance with hardware offload. Finally, TCP extended options in its current form is not safe to deploy.

Categories and Subject Descriptors

C.2.2 [**Computer-communication Networks**]: Network Protocols;

1. INTRODUCTION

The Internet was designed to be extensible; routers only care about IP headers, not what the packets contain, and protocols such as IP and TCP were designed with options fields that could be used to add additional functionality. The great virtue of the Internet was always that it was *stupid*; it did no task especially well, but it was extremely flexible and general, allowing a proliferation of protocols and applications that the original designers could never have foreseen.

Unfortunately the Internet, as it is deployed, is no longer the Internet as it was designed. IP options have been unusable for twenty years[10] as they cause routers to process packets on their slow path. Above IP, the Internet has benefited (or suffered, depending on your viewpoint) from decades of optimizations and security enhancements. To improve performance [2, 7, 18, 3], reduce security exposure [15, 29], enhance control, and work around address space shortages [22], the Internet has experienced an invasion of middleboxes that *do* care about what the packets contain, and perform processing at layer 4 or higher *within* the network.

The problem now faced by designers of new protocols is that there is no longer a well defined or understood way to extend network functionality, short of implementing everything over HTTP[25]. Recently we have been working on adding both multipath support[11] and native encryption[5] to TCP. The obvious way to do this, in both cases, is to use TCP options. In the case of multipath, we would also like to stripe data across more than one path. At the end systems, the protocol design issues were mostly conventional. However, it became increasingly clear that no one, not the IETF, not

"We tried to extend TCP to support multipath, but devices in the network kept rejecting our connections because they looked weird."

Quick pause
(People often have complaints to share about this problem!)

End to End Argument: Recap

- Basic argument: If you can implement functionality correctly and completely at endpoints, do it there and not at a lower layer.
  - It saves on redundant work in the system, and avoids confusion later. Exceptions okay for performance optimizations.
- Strong argument: Avoid putting unneeded functionality at lower layers of your system altogether because it's harmful!
  - Extra functionality at low layers constrains how applications are designed at higher layers.

Who decides what optimizations/ constraints are okay to deploy and what constraints aren’t?

…bringing us to Paper #2.

David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. 2005. Tussle in cyberspace: defining tomorrow's internet. IEEE/ACM Trans. Netw. 13, 3 (June 2005), 462-475.

“The Internet is not a single happy family of people dedicated to universal packet carriage.”

People want to talk in private.

Government wants to wiretap their conversations.

Conservative governments put their users behind firewalls.

Users install VPNs to tunnel through firewalls.

ISPs give users a single IP address and support “just one host”.

Users install address translators to support multiple hosts.

Tussle: “the ongoing contention among parties with conflicting interests.”

“Different parties adapt a mix of mechanisms to achieve their conflicting goals, and others respond by adapting the mechanisms to push back.”

A recent example…

INFINITE LOOP —

New guidelines outline what iPhone data Apple can give to police

Most iCloud data and some data from passcode-locked devices can be provided.

ANDREW CUNNINGHAM - 5/8/2014, 10:25 AM

INFINITE LOOP —

Apple expands data encryption under iOS 8, making handover to cops moot

"Apple cannot bypass your passcode and therefore cannot access this data."

CYRUS FARIVAR - 9/17/2014, 9:57 PM

LAW & DISORDER —

Judge: Apple must help FBI unlock San Bernardino shooter's iPhone

Specifically, Apple must create custom firmware file so FBI can brute force passcode.

CYRUS FARIVAR - 2/16/2016, 6:26 PM

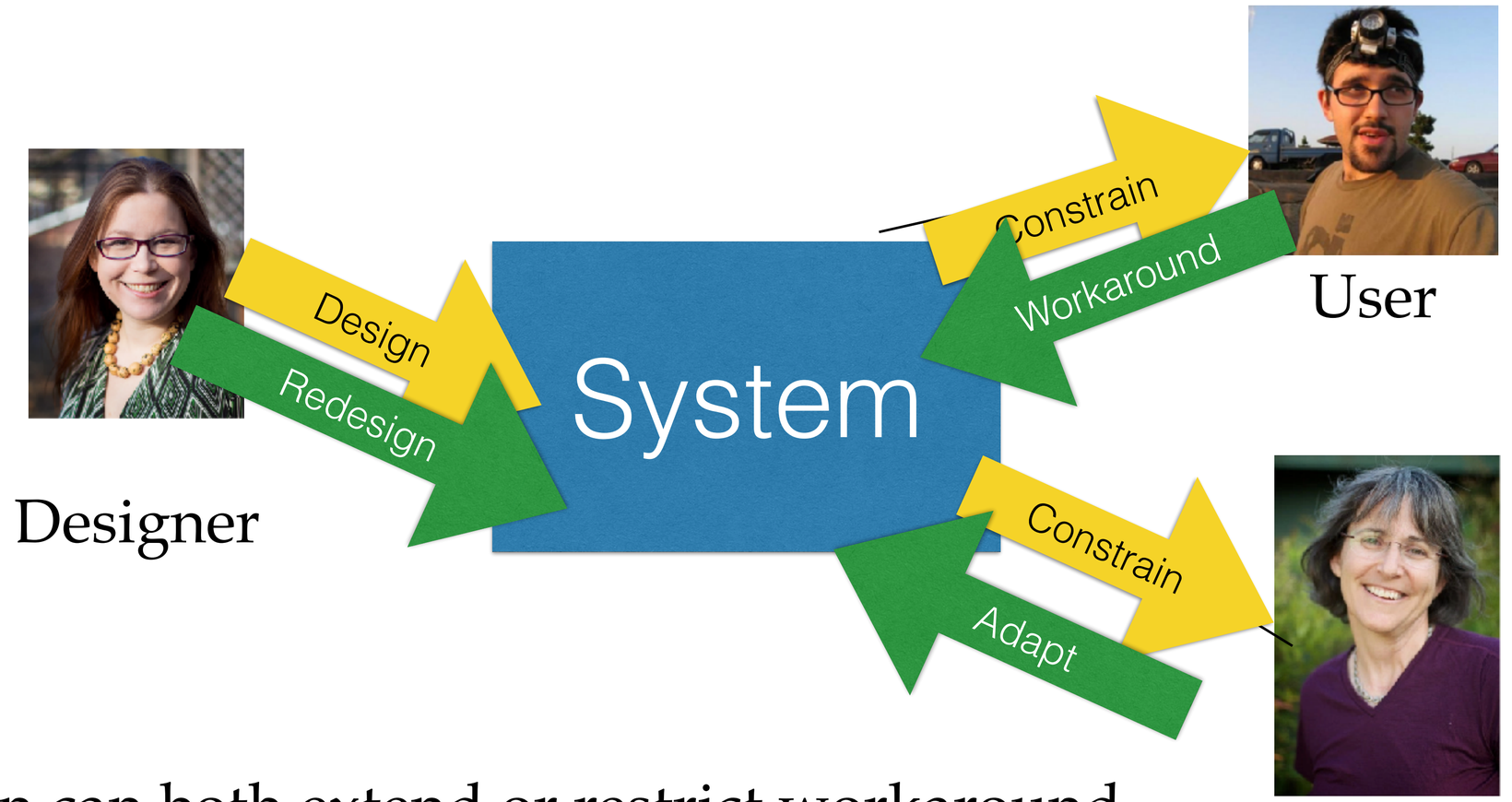
OUR TAX DOLLARS AT WORK —

FBI paid at least $1.3M for zero-day to get into San Bernardino iPhone

FBI Director James Comey: "But it was, in my view, worth it."

CYRUS FARIVAR - 4/21/2016, 12:30 PM

Actor Model of Tussle

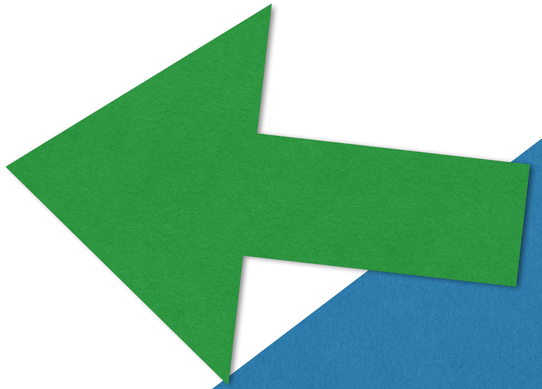


Redesign can both extend or restrict workaround.

“There is an open question … to whether the tussle will be driven out of the actor network, the actors will be forced into alignment, or whether this dynamic semi-stable system of today will persist.”

How do we design for tussle?

Design to stay out of the tussle.



Design to influence the tussle.

“Design for variation in outcome, so that the outcome can be different in different places, and the tussle takes place within the design, not by distorting or violating it. Do not design so as to dictate the outcome.”

Two Design Principles to Achieve This

- Modularize the design along tussle boundaries, so that one tussle does not spill over and distort unrelated issues.
- Design for choice, to permit the different players to express their preferences.

Modular Design: Make the tussle battle someone else's problem

Modular Design: Bad Example

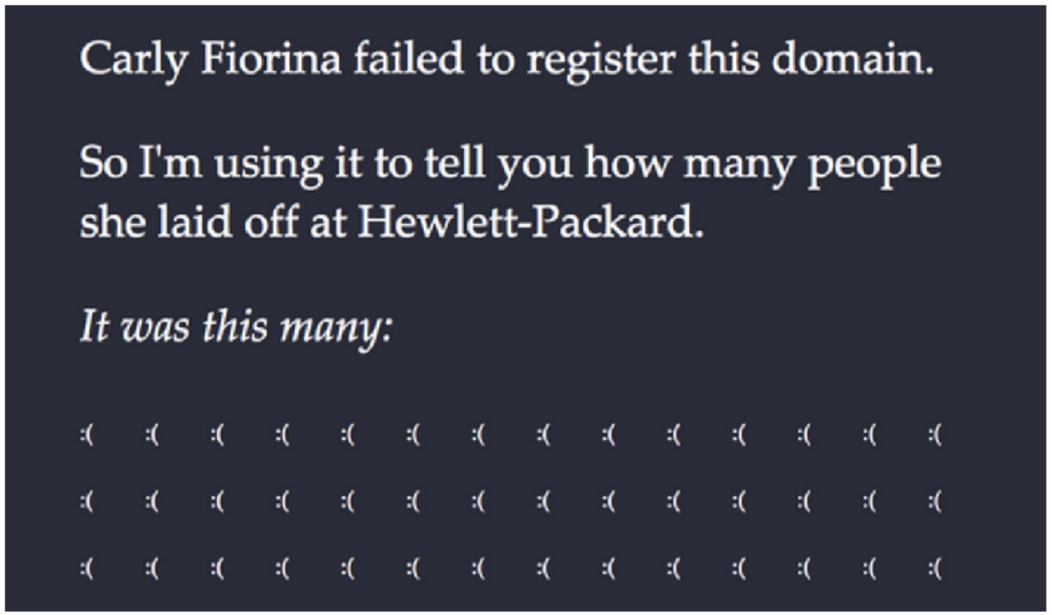
11:30 AM ET

By First Draft

Share

Tweet

Running for President? Better Name Your Website Domain Early



A partial screenshot of a message on carlyfiorina.org that is critical of her tenure as chief executive for Hewlett-Packard.

A message to anyone considering a political campaign: buy your domain names. Every possible one. And do so early.

Problem: DNS represents both "service ID" and "trademark"

I'm just trying to design a globally-addressable system and now I'm supposed to be an Intellectual Property expert?

Modular Design: Better Example

IP HEADER

0		15 16	31	
4 bit version	4 bit header length	8 bit type of service (TOS)	16 bit total length (in bytes)	4
16 bit identification		3 bit flags	13 bit fragmentation offset	8
8 bit time to live (TTL)	8 bit protocol	16 bit header checksum		12
32 bit source ip address				16
32 bit destination ip address				20
options (if any: next tcp | udp header)				
DATA				

Figure #10

It's not my job to know what is or isn't high priority: you tell me.

Design for Choice: let the user decide

That time the EU made MSFT design for choice.

We’re used to thinking about design for choice

- TLS supports multiple crypto algorithms, depending on who you trust :-)
- We have many different email providers.
- We have many different operating systems.

What's the difference between modular design and designing for choice?

Modular Design:

A global decision that constrains *everyone* must be made.

Don't make your system specific to any particular decision — let local government, enterprise, community decide what is right for them.

Design for Choice:

Decision is does not need to be the same for everyone.

Make sure your system supports lots of choices side by side — every individual user gets to decide for themselves.

Places where tussle plays out

- Some ISPs ban users from hosting web servers and block port 80. User solution? Run traffic over some other port.
- Residential broadband access: will we have choice between providers?
- Trust: I want to communicate with you, but you don’t want to communicate with me.

Can anyone think of other examples of systems designed for choice, or designed modularly?

There is no such thing as value-neutral design. What choices designers include or exclude, what interfaces are defined or not, what protocols are open or proprietary, can have a profound influence on the shape of the Internet, the motivations of the players, and the potential for distortion of the architecture.

Don't assume that you design the answer. You are designing a playing field, not the outcome.

[Editorial Note]

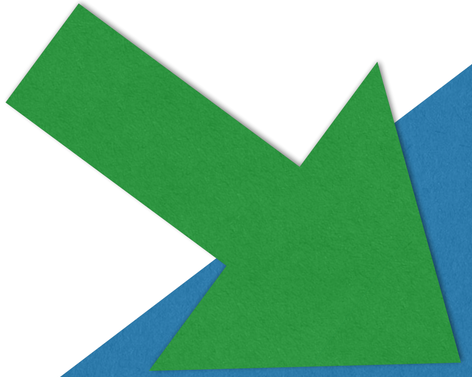
Paper's bias:

Openness, choice, and tussle are good.

A lot of pressure to end/restrict choice.

- App stores create “walled gardens” of whose apps will be permitted and whose won’t.
- Protocol standardization efforts influenced to restrict strong cryptography… preventing privacy.
- Firewalls, IDSes, Filters blocking protecting — and censoring, filtering out competition, or downgrading — traffic.
- Typically: corporations and governments who want to **control** what’s going on.

Design to stay out of the tussle.



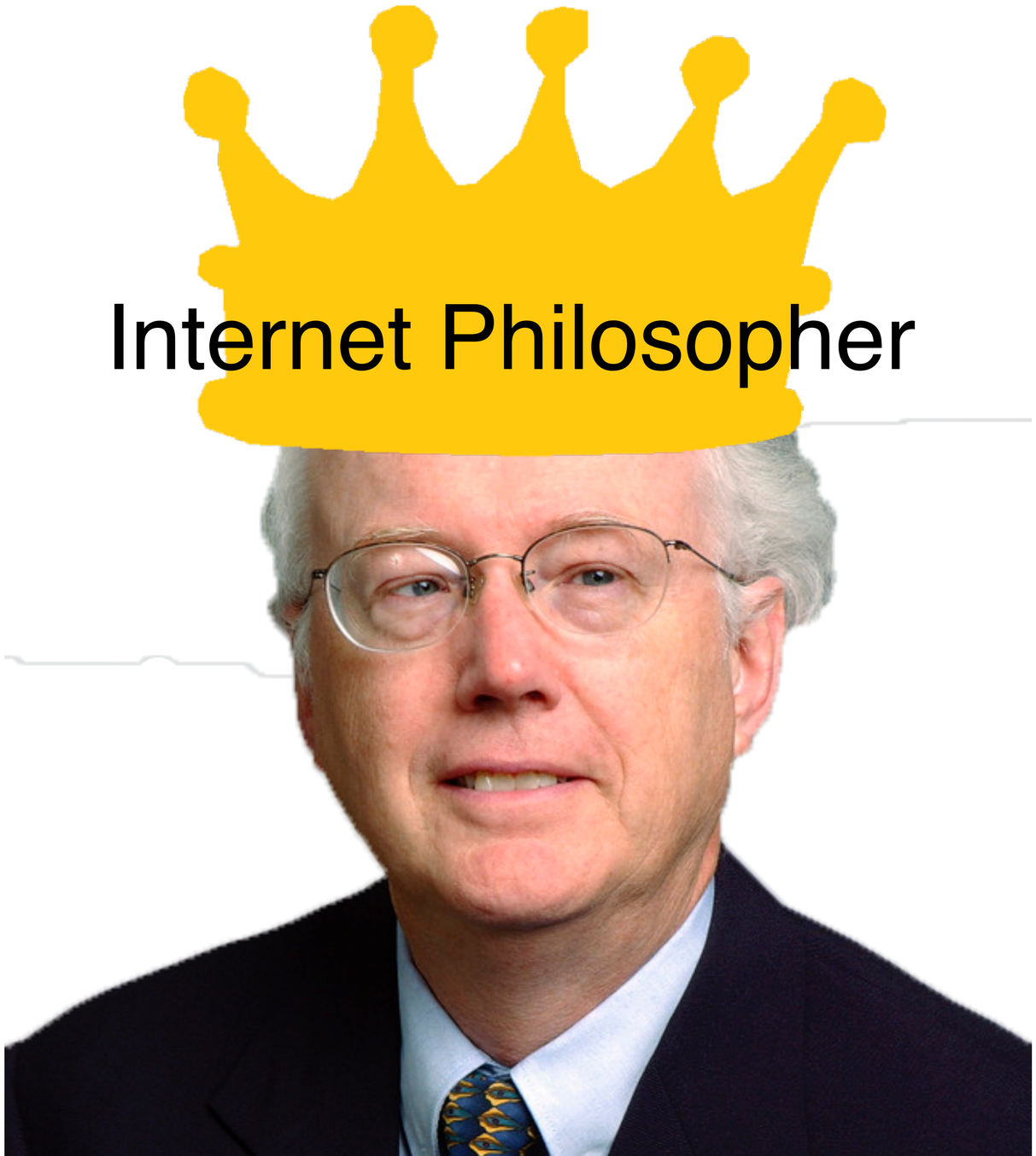
Design to influence the tussle.

In that everyone (rich, poor, citizen, government) else gets a say in how it plays out.

“Evolution and enhancement of existing, mature applications is inevitable.”

“Keeping the net open and transparent for new applications is the most important goal.”

“The most we can do to protect maturing applications is to bias the tussle.”



mic drop.

Back to you and your widget.

Now armed with some design principles

- End-to-End design: implement complex functionality at high layers of your system.
- Designing for choice: when multiple options about a tussle point are possible, allow users to select what's best.
- Modularity: when a choice has to be made about a tussle point, don't bake the answer in to your system.

Recommended Reading

"The Design Philosophy of the DARPA Internet Protocols", Clark, D. SIGCOMM 1988

"HTTP as the Narrow Waist of the Future Internet." Lucian Popa, Ali Ghodsi, and Ion Stoica. HotNets 2010.

"Is it still possible to extend TCP?" Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Internet Measurement Conference, 2011.

"The Evolution The Evolution of Layered Protocol Stacks Leads to an Hourglass-Shaped Architecture." S. Akhshabi, C. Dovrolis. ACM SIGCOMM 2011.

Are you giving your users choice, or adding constraints?

Me: @justinesherry / me@justinesherry.com