Network Protocols (destroyallsoftware.com)

842 points by signa11 on June 2, 2017 | hide | past | web | favorite | 75 comments

djrogers on June 2, 2017 [-]

Evan as a 20+ year network engineer, I don't think I've run across an article about networking that balances depth and breadth so well. All of the information presented is high-level enough to retain (at least as a big picture), but detailed enough to avoid hand-wavy 'magic networks' descriptions.

Bravo - well done.

Edit - Also worth adding that this article is a rarity in that the details are actually accurate! Even things I read in networking books and trades often have egregious errors - usually due to the breadth of the topic matter.

catern on June 3, 2017 [-]

It's slightly inaccurate in that it doesn't mention any other routing protocol than BGP. BGP is used between Autonomous Systems, but within ASs usually other routing protocols are used.

djrogers on June 4, 2017 [-]

To call this "inaccurate" because it's missing out on what is mostly an unnecessary detail is going way too far.

BGP is *the* routing protocol on the internet, and it's a fairly common one to see inside enterprise networks as well - iBGP is for routing inside an AS. Describing OSPF, IS-IS, etc would have just made things a little more complicated without adding anything.

dronemallone on June 9, 2017 [-]

BGP is a policy-based routing protocol, whereas OSPF/RIP are performance based. BGP is not "the" routing protocol, just that it's widely used because of its simplicity.

You should consider reading Kurose/Ross: http://eclass.uth.gr/eclass/modules/document/file.php/INFS13...

catern on June 5, 2017 [-]

Maybe I don't understand the realities of networking as well as I though. But my understand was that once packets enter my ISP's network, on my home connection, the rest of the way they get routed to me with protocols other than BGP. That's quite a lot of routing being done with non-BGP protocols.

dsr_ on June 5, 2017 [-]

Between major networks, BGP.

Inside the major network to your cable headend or similar: OSPF or IS-IS or something else - in the largest ISPs, a combination of iBGP to get things to right area and then OSPF, IS-IS or EIGRP in medium areas.

> Inside your house: almost always static routes, no protocol for exchanging reachability info.

ice109 on June 3, 2017 [-]

such as?

emcrazyone on June 3, 2017 [-]

Exactly, I thought the very same thing. In fact, I read it with a critical eye at first and as I got through I was quite pleased. I thought it was really cool to mention the 8b/10b encoding near the end. yes, well done indeed!

have_faith on June 2, 2017 [-]

As a front-end web developer with no formal computer science background or traditional programming experience I find these kinds of articles extremely valuable. I like to understand as much as possible, at least conceptually, what happens throughout the stack even if I don't touch it. Does anyone have any links to anything similar? perhaps for the Linux kernel or other lower level systems but with a top down overview like this? Especially anything that would build on this article. Effects and unexpected phenomena that manifest in networks like this also would be interesting.

rectang on June 2, 2017 [-]

For Linux/Unix, I'm going to recommend a book, which for now is probably too ambitious: "Computer Systems: A Programmer's Perspective" by Bryant and O'Hallaron, a.k.a. "CSAPP".

http://csapp.cs.cmu.edu/3e/perspective.html

This book is targeted at working programmers:

> Most books on systems—computer architecture, compilers, operating systems, and networking—are written as if the reader were going to design and implement such a system. We call this the “builder's persepective.” We believe that students should first learn about systems in terms of how they affect the behavior and performance of their programs—a “programmer's perspective.”

I'm recommending CSAPP to give you an idea of where the end lies. If you grok its material, you will know more about OS kernels than would be required for any typical front-end or back-end web development job.

eriknstr on June 2, 2017 [-]

I have a copy of an old edition of that book on my bookshelf, published 2003. I haven't gotten around to read it yet. I bought it because the cover stood out to me at a flea-market, and also because it was one of only a few books there which were about computers and I wanted to pick up some computer books for cheap.

Here is a photo of my copy of the book: http://i.imgur.com/2sd5ivf.jpg. Don't the front look nice? :)

arawde on June 2, 2017 [-]

This was the textbook for my intermediate operating systems class, and I'd second this recommendation.

closeparen on June 3, 2017 [-]

This was the textbook for my first-year intro to systems and I'd recommend it too!

mafuyu on June 2, 2017 [-]

I took this course with O'Hallaron and it's fantastic. I highly recommend doing the labs that come with it- they're fun and give you lots of working experience with the material.

indigochill on June 2, 2017 [-]

Depending on how deep you want to go:

https://www.coursera.org/learn/build-a-computer - This is a fantastic no-prerequisites course that has you actually implementing a functional CPU in 6 weeks starting from building basic logic gates out of NAND gates. There's a hands-on project every week to help you solidify your understanding of the lectures. The resources are also all available at their site: http://www.nand2tetris.org/. In case that sounds intimidating, I have no formal CS education either and couldn't implement any of the basic logic gates before I took the course, and I built a CPU by the end so I'm confident others can too. They have a "Part 2" to this course which has you build from the CPU to a functional operating system which would get you an introduction to how kernels work in general, although they recommend some programming experience for that part.

Hacking: The Art of Exploitation - Despite the name, this book is actually an excellent introduction to low-level behavior both on the CPU and on networks (it's split into distinct sections to cover each topic). I would recommend picking this up after the course linked above, because it's somewhat brief with its explanation of CPU architecture. A beginner's course in C programming might also be advisable since this does use some basic C code which might be challenging if you've never been exposed to it before.

pmf on June 2, 2017 [-]

Pick a simple driver (for example, for a serial device / UART; [0] should be the Raspberry Pi's UART); prefer ARM, since this is more straightforward. From there, you can work your way up from how data arrives in controller registers, to how the driver handles the interrupt (you'll learn about ISRs and DSRs and how to read a microcontroller datasheet), to how the driver notifies the application program thread waiting on the data (you'll learn about scheduling), to how the read syscall transfers memory from kernel space to userspace. I'd not recommend diving directly into the network driver stack, since this has a lot or intermediary layers that do not further basic understanding[1].

Perhaps you should first try to get familiar with the Linux kernel's way of doing OO in C (if you are unfamiliar with this style; if you have done any modern C programming, it should not be a problem). (Do not start with the platform startup code; this would probably be too much for the start.)

[0] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/lin... (maybe a bit too complex, since it uses DMA)

[1] To be fair, the serial I/O layer has the tty layer as intermediary, so it's admittedly only slightly better

jmagoon on June 2, 2017 [-]

I'm pretty surprised that no one has recommended the other articles on the same

site. They are pretty excellent: https://www.destroyallsoftware.com/compendium

The screencasts are fantastic as well.

> Sundiata on June 2, 2017 [-]
>
> At $29 a month it's pretty steep though. I'm unable to afford it right now, however after reading the linked article I'm tempted to get a subscription.

dcosson on June 2, 2017 [-]

After you read a bit on it, you can really learn a lot by opening wireshark and watching what's going on, googling for tutorials and documentation to understand the settings.

As a front end developer there is actually a lot of networking involved making requests to the servers. It's really helpful to have a high level picture of how http works, and what common networking failure modes are, so you can use better abstractions in your code, expect common failures and shown nice error messages or retry requests, test your app's connections using wirshark to make sure there aren't extra round trips slowing it down, etc.

zxcmx on June 2, 2017 [-]

IMHO the classic is still TCP/IP illustrated by Richard W. Stevens.

avichalp on June 2, 2017 [-]

Check these comics they are mostly about linux and networking: https://jvns.ca/zines/

> sakawa on June 2, 2017 [-]
>
> I really hope that's not my bias; I really like the format, but I find them lacking of some background information, or sometimes going.. stupid. ._.
>
> > always_good on June 2, 2017 [-]
> >
> > What bias could you be referring to? And what does "sometimes going stupid" mean?

falcolas on June 2, 2017 [-]

Even as a relatively network-savy developer, the 8 to 10 bit translation done over the wire was news to me. I love these kinds of implementation details.

> packetized on June 2, 2017 [-]
>
> If you really want a fun time read about bipolar encoding & bipolar violations in T1s/E1s, which were once a source of much consternation to me - in a former life.
>
> https://en.m.wikipedia.org/wiki/Bipolar_violation

dominotw on June 2, 2017 [-]

Linux Inside gitbook

https://www.gitbook.com/book/0xax/linux-insides/details

ajdecon on June 2, 2017 [-]

For networking in particular, Andrew Tanenbaum's book "Computer Networking" [0] is quite good, and covers everything from the physical layer on up. The most recent edition is from 2010, but much of this doesn't change quickly so it's still quite relevant. I've found it extremely helpful at times.

[0] https://www.goodreads.com/book/show/8515228-computer-network...

jsingleton on June 2, 2017 [-]

Shameless plug but I cover a lot of lower level network and hardware effects for web developers in this book: https://unop.uk/book/

It's mainly aimed at people interested in ASP.NET Core but there are a lot of general performance tips in there. I have a EEE background and I think understanding things down to the transistor level can really help you write better software.

BTW Gary's WAT video is great fun if you haven't seen it.

readittwice on June 2, 2017 [-]

Same for me, although I studied computer science. I enjoy to read such nice written articles about technology. Even though most of the text was about stuff I already knew, there still was something new (maybe I just forgot?) for me: retransmission via ACK messages. But even if not, I still appreciate reading about this stuff again to freshen my knowledge. If I don't work on/with these topics regularly, I tend to forget about some of the details.

teh_klev on June 2, 2017 [-]

I can thoroughly recommend "Internet Routing Architectures 2nd Edition" by Sam Halabi. It's an older text, but much of it is still relevant:

http://amzn.eu/5tCOQfW

It was my goto book when I worked on BGP-4 odds and sods.

foota on June 2, 2017 [-]

You might find this interesting in the vein of weird networking things: https://www.ibiblio.org/harris/500milemail.html

Quequau on June 2, 2017 [-]

As an old dude who went through engineering before networks were a major thing and worked some years on embedded devices, I also find these sorts of articles extremely interesting.

Radle on June 2, 2017 [-]

I agree, I had a course about this topic in university and I was like "This is why I am here".

There are things you don't know that you don't know. And those always get you.

manigandham on June 2, 2017 [-]

I always recommend the *High Performance Browser Networking* book by Ilya Grigorik for a fantastic overview of modern web protocols. It's also free to read online.

https://hpbn.co/

devy on June 2, 2017 [-]

By the way, this is Gary Bernhardt's personal site. His lightning talk "Wat"[1] from CodeMash 2012 was one of my all time favorite talks, witty and right on!

[1] https://www.destroyallsoftware.com/talks/wat

andars on June 2, 2017 [-]

> In reality, our 5-volt CMOS system will consider anything above 1.67 volts to be a 1, and anything below 1.67 to be 0.

Worth noting that the region from 1.67 V to 3.33 V is undefined and systems in practice will not behave nicely for signals in this range. A CMOS logic 1 needs to be above 2/3 Vdd to be reliably recognized.

upofadown on June 2, 2017 [-]

Expending on your correction, there are some appropriate diagrams in this article:

* https://www.allaboutcircuits.com/textbook/digital/chpt-3/log...

Any practical binary logic scheme is going to have an undefined zone in the middle because the gain of the devices used for the logic is not infinite and there would be problems caused if the gain was too high.

pololee on June 2, 2017 [-]

Very good reading. I'd also recommend Van Jacobson's talk https://www.youtube.com/watch?v=gqGEMQveoqg You'll learn some good stories about internet history.

He has done a lot of work on TCP congestion control, especially the fast retransmission idea by using duplicate ACKs.

There is an interesting thing about TCP. A lot of popular TCP implementation use city names, e.g. TCP Reno, TCP Vegas, TCP Westwood.

TCP Westwood is a very interesting implementation. It has very intelligent way to estimate bandwidth, (not just based on duplicate ACKs). You may find this paper very interesting. http://netlab.cs.ucla.edu/internal/wiki-internal/files/rohit...

dronemallone on June 9, 2017 [-]

This article is missing quite a few things that's of interest to programmers:

1. IPv4 fragmentation & reassembly

2. Centralized (Dijkstra/OSPF) vs. Distributed routing (distance vector/RIP) - stuff you see in Algorithms class

3. TCP mechanisms: congestion control mechanisms that the user can configure (Cubic/westwood/new reno), flow control, retransmission timer calculation (exponential weighted moving average)

4. explicit congestion notification and other add-ons, which can be enabled in the OS by the user

5. Active Queue Management and enabling QoS on your system: stochastic fair queueing for example

6. the recently introduced TCP Fast Open mechanism

7. TCP auto tuning: http://kb.pert.geant.net/PERTKB/TCPBufferAutoTuning

8. Ethernet physical layer: you've left out modulation, and only discussed encoding

9. Multicast and spanning tree protocols

bogomipz on June 2, 2017 [-]

His articles are always fantastic. I wish he would consider publishing a book however as $29/month to subscribe to a blog feels a bit steep.

elmigranto on June 2, 2017 [-]

Being a blog is a temporary medium change. There are many hours of unix, git, and programming screencasts and a series on computation in those $29. If "per month" part feels too steep, nothing stops you from making it one time thing by downloading everything and cancelling subsequent payments.

https://www.destroyallsoftware.com/blog/2016/state-of-das-de...

bogomipz on June 2, 2017 [-]

Oh that's interesting. Thanks for the link. The live streaming idea is certainly exciting and compelling. Cheers.

Obi_Juan_Kenobi on June 2, 2017 [-]

$29/month feels alright if you're using it as a professional resource, i.e. you're an employed programmer.

As a student/learning resource, it seems quite steep. Yet, compared to traditional education, it's a screaming steal.

Pricing is hard.

sn9 on June 2, 2017 [-]

A monthly $30 will soon grow more expensive than a few decent textbooks that cover the same material in greater depth.

gary_bernhardt on June 2, 2017 [-]

One of Destroy All Software's primary value propositions (even included in the subtitle of the front page of destroyallsoftware.com) is terseness. My customers pay me more money because I produce small things. You can get quantity anywhere.

sn9 on June 3, 2017 [-]

No I get that, and I enjoy your writing. It's certainly a valuable niche to occupy.

I just think people tend to underestimate the ROI on buying and working through a textbook.

Alex3917 on June 2, 2017 [-]

So does BGP consider the amount of time it takes to traverse each hop, or are routing tables built only based on the minimum number of hops it takes to reach each destination?

dsr_ on June 2, 2017 [-]

The first thing that a BGP speaking router does is to look up the destination IP address in its table of autonomous system numbers (ASNs). An AS is a group of networks that are all under the control of the same policy maker. If you get your internet connectivity from one ISP, you fit in their AS. If you talk to multiple ISPs,

you need to be your own AS (and have routable IP addresses assigned to that AS). Finally, the largest networks may be a confederation of several AS, either as a result of purchases or by deliberate policy.

BGP transmits reachability information by building paths of AS that are neighbors. Then, typically, it selects the shortest path of AS hops. (There are a lot of knobs to twiddle with here, both on the advertising side and on the deciding side.)

BGP doesn't care or know about numbers of router hops, just the number of AS between here and there.

There are lots of "network engineer in a box" products that map out performance and then twiddle knobs on your BGP router to select for best performance (or least cost, if some of your paths are more expensive than others.)

tyingq on June 2, 2017 [-]

BGP doesn't inherently measure time for hops. The base setup is to use the path with the lowest number of unique Autonomous Systems (AS). But, there are also concepts of weight, where you can influence the chosen path manually. And, there's also "Multi-Exit Discriminator", or MED, where you can pass a metric to a peer on what the best path to a specific AS is.

Even that's an unfair simplification, though. Cisco has a pretty good page that shows how path choices are made: http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-...

There are products, though, that tweak these settings to pick for best performance, or lowest bandwidth cost.

bogomipz on June 2, 2017 [-]

The latter, BGP is simply a path metric. Your question reminded me though that Internap used to sell an appliance called FCP Flow Control Platform that would take the top 10% of destination ASNs from your edge(via Netflow)and run ping probes to them and allow you set the preference via iBGP to prefer routes with the lowest latency. It was nice in that you could route around network "brown outs" upstream. It has since turned into their Miro platform. There was also a similar appliance called Route Science. I was always surprised that there wasn't ever an open source version of something similar. Although you don't hear a lot about these appliances today. I think maybe because of the proliferation of CDN edge networks.

http://www.internap.com/network-services/miro-controller/

IamMario on June 2, 2017 [-]

I am out for a few years and haven't done too much with BGP, but iirc it heavily depends on your BGP configuration. BGP is a highly complex routing protocol and you can tweak it to your exact needs. On default it is just like RIP and takes the shortest path, except that RIP counts each router and BGP only counts the outbound routers.

> bogomipz on June 2, 2017 [-]
>
> >" BGP is a highly complex routing protocol and you can tweak it to your exact needs."
>
> BGP itself is actually a relatively simple protocol. The complexity really comes from all the available configuration options.

iajr39r4 on June 2, 2017 [-]

A bit off topic but, I noticed most of the titles on destroyallsoftware are rendered as SVGs.

Why is that a better idea than just normal text?

tomr_stargazer on June 2, 2017 [-]

Gary is a connoisseur for pixel-perfect large text that exactly fills the given space:

https://twitter.com/garybernhardt/status/780497807434100736
https://twitter.com/garybernhardt/status/760648828772941824

and

"What I really want to do is tell CSS to make text a certain width, but technology is not yet advanced enough!" https://twitter.com/garybernhardt/status/754036917864247297

jancsika on June 3, 2017 [-]

Interesting. Looks like in Inkscape he uses the OS's font-rendering engine to render the text in the chosen font, then converts the rendered font to an SVG path.

So the benefit appears to be text that is guaranteed to fit a particular pixel-width (minus any anti-aliasing discrepancies among browsers). The cost would be baking one particular OS's font-rendering idiosyncrasies into the SVG-path page titles, but that's partially hidden by the choice of thick bold-faced fonts that hide hinting discrepancies (plus using all caps).

Makes me wonder if one could infer which OS he used to generate the SVG title by converting the text to a path on each platform where Inkscape runs, then comparing each to his SVG title.

BFatts on June 2, 2017 [-]

So you cannot directly copy their text, I would assume, and republish it since they charge a subscription for access to their articles.

tyingq on June 2, 2017 [-]

It's just the main headline title and subtitle. The actual article text and subheaders are regular html.

So I would guess it's a just a personal preference for that typography that they felt they couldn't get some other way.

The svg images do have correct alt tags with the right text.

steveklabnik on June 2, 2017 [-]

I believe that you're accurate; Gary cares a _lot_ about presentation, I vaguely remember seeing him tweet about the kerning here or something.

elmigranto on June 2, 2017 [-]

Firstly, it's only (sub)titles.

Secondly, those can actually be selected with mouse (and copied), accessed by a screen reader and even `innerText`ed.

Ericson2314 on June 3, 2017 [-]

It's good, but... I wish it were more critical? Excluding the gross control plain (as it wasn't the focus), there's some awkward overlap between IP and Ethernet (link aspects).

My guess, that I'd love to see explicitly confirmed, is that it goes back to the internet as the internetwork—lingua franca between existing networks an idea predating the more technically-motivated concept of layed protocols providing compounding abstractions.

I don't want to sound like a nit of an otherwise great piece, but without criticism the history seems inevitable. Alternatives and hypotheticals are good to keep design space from atrophing in the face of collective amnesia.

tyingq on June 2, 2017 [-]

I assume it's not mentioned to keep the article brief, but most devices these days support MTU sizes greater than 1500 bytes. Jumbo Frames[1] allow for ethernet packets of up to 9216 bytes.

Since they have to be fragmented back down to 1500 for devices that don't support them, however, it's typically only used in closed internal networks, like a SAN. People typically see about a 5% to 10% bump in performance.

[1]https://en.wikipedia.org/wiki/Jumbo_frame

packetized on June 2, 2017 [-]

I think you might be conflating MTU and frame size - IPv4 MTU (L3) can be up to 64kB, whereas jumbo frames (L2) can be up to the stated 9kB.

tyingq on June 2, 2017 [-]

Well yes, but they are closely related. You set the MTU/MRU to take advantage of the availability of larger frames. There isn't some separate adjustment on Linux, for example.

Myrmornis on June 3, 2017 [-]

This is great. Along similar lines, "Foundations of Network Programming" by Brandon Rhodes (and originally John Goerzen) is fantastic (and not just for python programmers as the python API is a pretty transparent wrapper over POSIX APIs).

lttlrck on June 3, 2017 [-]

Alternative explanation of the 1500 byte MTU given in the last paragraph:

https://networkengineering.stackexchange.com/a/2964

notdonspaulding on June 2, 2017 [-]

I love Gary Bernhardt's stuff.

I love this article because of the depth and detail which can be expected of his work, but also because you get all the way to the last sentence before he reveals the question which inspired him to do the deep dive.

paulddraper on June 3, 2017 [-]

> An interpacket gap of 96 bits (12 bytes) where the line is left idle. Presumably, this is to let the devices rest because they are tired.

RubenSandwich on June 2, 2017 [-]

I hate to be that guy. But I don't think this link was meant to be for the general public. Gary Bernhardt, the author of this piece, posted this link to this Twitter followers about 2 weeks ago to receive feedback. Remove the hash at the end of the URL, '/97d3ba4c24d21147', and you'll see you'll be redirected to purchase a subscription to Gary's screencasts and articles.

So if you are enjoying this article consider purchasing a subscription and supporting more work like this.

doty on June 2, 2017 [-]

If you are subscribed, the footer of the page reads as follows:

You can share this article! This URL can be posted anywhere you like, including in public. Anyone clicking on it can read the article without logging in. The URL is specific to your subscribed account, but no one outside of Destroy All Software can identify your account simply from the URL.

gary_bernhardt on June 2, 2017 [-]

No, it's OK for this to be on HN. I'm doing an experiment with the compendium where the individual articles are publicly linkable, but navigating to one from the index requires a subscription.

frant-hartm on June 2, 2017 [-]

Isn't this a freely available promo article?

> This article is part of The Programmer's Compendium. > You can get access to the full Programmer's Compendium by subscribing to Destroy All Software.

ben0x539 on June 2, 2017 [-]

I think the idea is that every article is a freely available promo article as it comes out, and then gets relegated to the subscribers-only backlog afterwards.

RubenSandwich on June 2, 2017 [-]

Good point, I didn't notice that on the bottom. But I think my argument still stands that a link posted to his Twitter followers is a lot different then it posted for all of HN.

TeMPOraL on June 2, 2017 [-]

I disagree. Twitter is a broadcast medium. That's unlike e.g. Facebook, when by default you target a specific audience (your Facebook friends).

tokenizerrr on June 2, 2017 [-]

With post to his Twitter followers you mean publicly on his timeline?