Google fiber

Speed is Hard

Avery Pennarun

January 2015

Please sign-in: http://goo.gl/XskAvv

Presented at the University of Waterloo, January 15, 2015

What do we do?

Internet service!

...and TV service, and devices, and phone/email/web support,
and we dig holes and fix postal code maps and bill credit cards and
optimize routes for hundreds of install trucks and and...

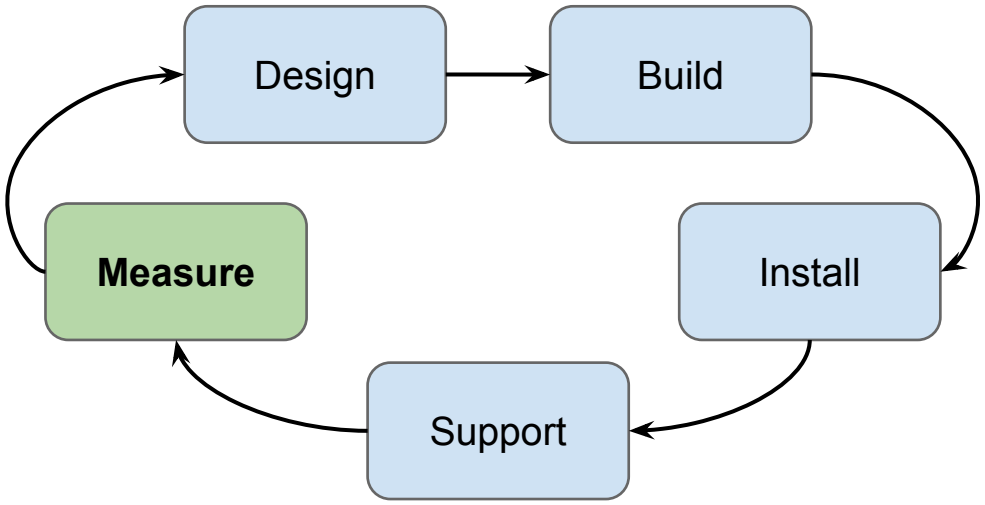
You might think running an internet service is just hooking up some fibers and hoping for the best. But it's actually a huge project, especially if you want to do a really great job. We manage the whole thing, including digging holes in the street to building devices, programming the devices, centrally managing the whole fleet using custom software, debugging live problems, answering support calls, etc.

And yes, you, dear Waterloo student, can apply for jobs doing any of those things.

My team mostly makes devices

My team in particular makes the firmware for in-home devices: fiber-to-ethernet adapter, firewall/router/wifi box, TV/DVR boxes, etc. We work along with a hardware team that actually does the hardware designs.

To make the whole Internet go fast



Our goal is to make the whole Internet experience faster and more pleasant. Not just the last hop to your house, but the whole end-to-end link. We carefully measure the results we get so that we can find bottlenecks and work to eliminate them, one by one.

How fast is a gigabit?

1000 Mbps
= 100 MBytes/sec
= 360,000 MBytes/hour
= 360 GB/hour
= 8640 GB/day
= 8.6 TB/day
= **~3 x 3 TB hard disks per day**
...per customer
…in both directions!

This simple calculation shows how much we're really talking about when we talk about a gigabit per second. Short version: it's a lot. Very few people will actually use a gigabit per second, 100% of the time, all day.

That's a huge change from the way we normally think of the Internet, in terms of scarcity. Suddenly Internet bandwidth is one of the least scarce things in your home network: the limitations move elsewhere.

So web pages load 100x faster?

It depends.

...and so here's the thing. 1000 Mbps really is at least 100x faster than a typical Internet connection. But just increasing the megabits doesn't automatically make all your web pages load faster in the same proportion. Downloads can go 100x faster, in many cases. And typical web pages do load faster, but 100x faster? Not exactly.

Why not?

- Fundamental limits of TCP
  tinyurl.com/bwlimits
- Latency ruins everything
- Computers are too slow
- Wires are too slow
- Wireless is **way** too slow

There are several reasons web page speeds don’t increase linearly, which we’ll cover in the next slides.

TCP Speed Limits: tinyurl.com/bwlimits

delay*bandwidth product:

the number of bytes “in flight” at one time.

the amount of memory you need per stream.

0.1ms * 100 MBytes/sec = 0.01 MBytes

300ms * 1 MByte/sec = 0.3 MBytes

300ms * 100 MBytes/sec = **30 MBytes**

First of all, there are fundamental limits to any windowed transfer protocol, including TCP. Windowed transfer protocols are governed by something called the delay*bandwidth product: the amount of data that can be “in the pipe” at once. Basically, you multiply the bitrate (Mbits/sec) by the round trip time (sec), giving an amount of data (Mbits). If you think about it, that’s the amount of data you can send before you have any notification about whether the data has arrived.

Because data might be lost in transit (and often is), you need to be able to retransmit it if it gets lost. Thus, you need to maintain a “transmit window” of at least the delay*bandwidth product (and usually a bit more) on the sender side. Then, if you get notification from the receiver that some data has not arrived, you still have it available to be resent.

This works okay on fast-near networks (like gigabit ethernet). In that case, you might have, say, 0.1ms * 100 MBytes/sec = 0.01 MBytes = 10 KBytes of buffer.

It also works fine on slow-far networks (like 10 Mbit/sec = 1 MByte/sec Internet across the world). You have more latency, say 300ms, but less bandwidth. 300ms * 1 MByte/sec = 0.3 MBytes = 300 KBytes. A little more than a local network, but still pretty reasonable.

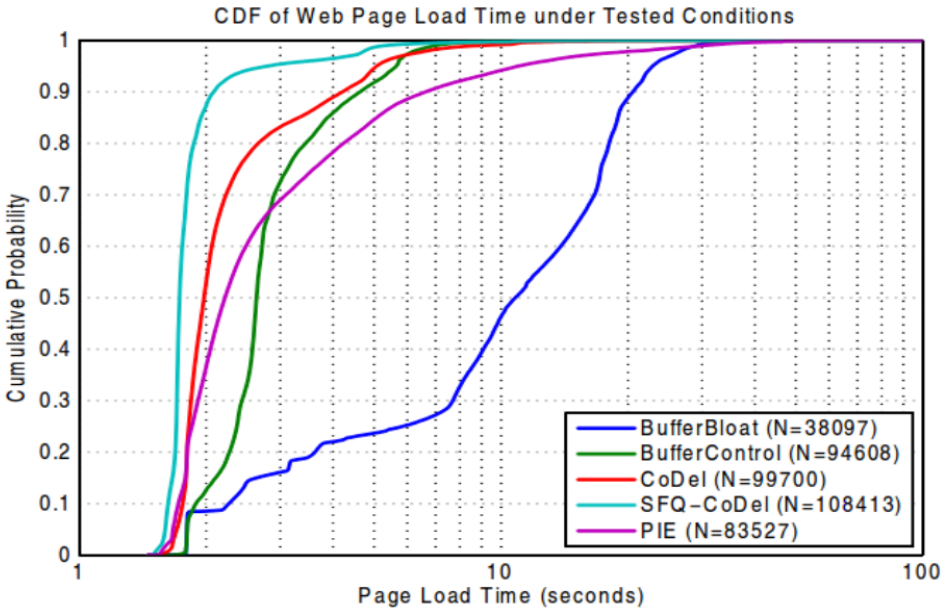
But nobody really expected to have fast-far links, with high latency and high speed. In our example, you go around the world, so say 300ms, but get 1000 Mbps = 100 MBytes/sec. That’s 300ms * 100 MBytes/sec = 30 MBytes of buffer required,

minimum.

TCP implementors assumed this would never happen, or if it did, they would have time to twiddle some settings in their OS to make it possible. We're now experiencing that time delay: many operating systems don't even have the ability to use a window size that large, or have it restricted to be much smaller by default, under the assumption that any program needing a window that large is probably just buggy.

Besides just limitations in current TCP implementations, there's the question of memory: if you need to reserve at least 30 MBytes of buffer per stream, and you might have several streams going at once, you're going to need a lot of memory. This is okay nowadays, because memory is pretty cheap, but people have been designing devices for years with less memory. For example, a typical off-the-shelf wifi router might have 32 MBytes of RAM *total*.

Latency ruins everything



- Optimized routes
- Peering

- SPDY
- HTTP/2
- QUIC

- Edge caching
- Prefetching

Another reason web pages aren't fully 100x faster is that, beyond a certain point, download speeds are dominated by latency. Here's a chart I borrowed from a different presentation by Dave Taht. It's not directly applicable to Google Fiber, but it shows the general idea of what we're talking about.

This is a "cumulative probability" graph. The different lines represent different queuing algorithms, which for this discussion is a stand-in for the amount of observed latency on a link. The X axis is the the total web page load time, and the Y axis is how likely it is that a page will load in that amount of time (X) or less.

For this particular set of web pages, you can see that the probability of loading in less than, say, 1 second is zero. The probability of loading in less than 100 seconds is 100%. The light blue line on the left (labeled SFQ-CoDel) is best, because it has the highest probability of loading pages in the shortest amount of time.

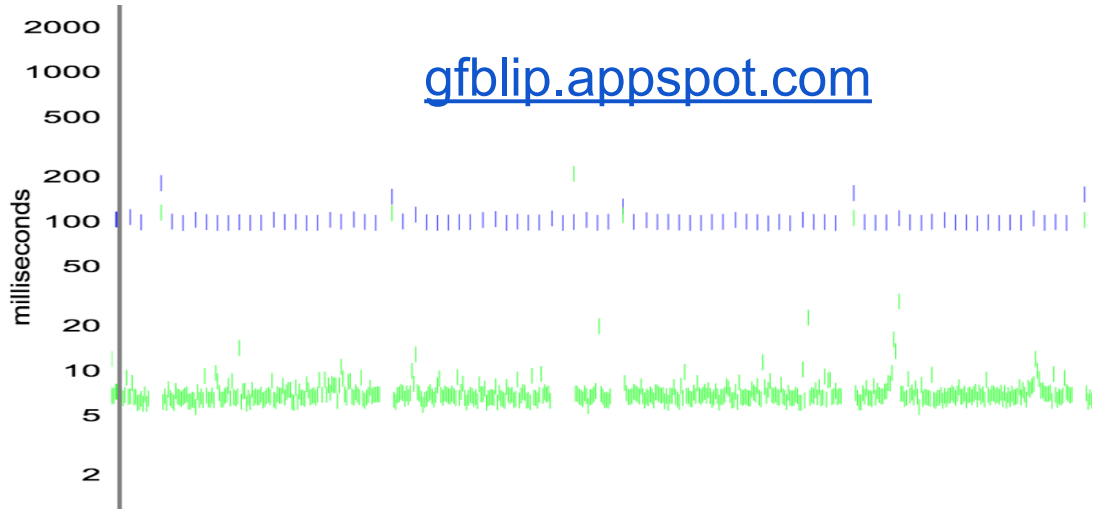
These different queuing algorithms can improve things even on slower links, by reducing unnecessary latency (aka "bufferbloat" - google it) using software techniques.

But some latency, such as much of the latency on a Google Fiber network, can't be directly removed like that. It's caused by limitations of the speed of light. A signal going across the continent or around the world simply takes a certain amount of time to arrive, and that amount of time has become significant. You can increase bandwidth as much as you want through brute force - just add more fibers, if you must

- but you can’t increase the speed of light.

So, beyond a certain speed, page load time is limited by latency. If you’re a web designer, you can help with this by minimizing the round trips needed to load your page. Google projects like SPDY and QUIC are all about this kind of optimization and are slowly paying off.

Measure Latency from Anywhere



I made a tool called gfblip that allows you to measure latency.

It turns out that the worst part about flakey wifi links isn't really the speed, it's the variable latency. gfblip runs on any javascript-capable browser (including phones and tablets) so it's a good way to get a good overview of what to expect. Because it does so many requests in a short time, you can also use it for finding wifi dead zones in your house: load it on your phone and wave it around to see when the connection gets bad or good.

You can find the source at http://github.com/apenwarr/blip.

Computers are too slow

- 1000 Mbps @ 1 GHz CPU
  = 1 bit per Hz
  = ~32 asm instructions per 32-bit word
  ...which is not enough for modern programmers
- Hardware assisted forwarding/firewall/NAT

Besides TCP windowing and latency, another problem with speed is that a gigabit actually takes a lot of CPU power to process. To put this in perspective, if you have a one-core 1 GHz CPU (which is pretty typical for a wifi router), and a 1000 Mbit/sec connection, that gives you about 1 bit per Hz. Assuming you can group bits into 32-bit CPU words, that means you can have 32 asm instructions per word… assuming your asm instructions are perfectly pipelined and branch predicted and all run in exactly 1 clock cycle, which is pretty optimistic.

Naturally, most programs don't have this level of optimization, so most 1 GHz 1-core CPUs aren't able to push a full gigabit. And splitting the work between multiple cores is pretty tricky, even if your CPU has multiple cores. So what we find is that even though we can deliver a gigabit to your house, many older PCs can't keep up.

One option (which we use in our own GFiber boxes, since they have the same problems!) is to use hardware-assisted forwarding. So interestingly, our GFiber Network Box can *route* 1000 Mbits of traffic between your computer and the Internet without a problem, but it can't actually *generate* 1000 Mbits of traffic on its own, because its CPU is too slow. That's okay though, because the network box is mainly about routing.

Wires are too slow

- Most homes don't have ethernet
- USB 2.0 max speed: ~280 Mbps
- Short-range coax cable (MoCA) max speed:
  - 2012-2013: ~200 Mbps
  - 2014: ~400 Mbps
  - 2016: ~800 Mbps?

Even if your CPU is fast enough, just distributing 1000 Mbps through your house is pretty tricky. Gigabit ethernet works, but most people don't have ethernet cables distributed all through their house.

[During the talk, I polled the audience, which was the Computer Science Club at the University of Waterloo. Of course *they* all have gigabit ethernet all through their houses. But they're… not normal. :)]

Plus, many laptops don't even have ethernet ports nowadays. Even if you buy a supposedly gigabit ethernet port to connect to USB 2.0, the maximum speed USB 2.0 can theoretically go is about 280 Mbit/sec, far less than a gigabit.

What you *do* probably have in your house is coaxial cable, originally placed there for old-style analog TV connections. Google Fiber uses a standard called MoCA to distribute Internet through your house on these pre-existing cables. MoCA is getting faster and faster. Google Fiber's MoCA in 2012-2013 could do about 200 Mbps. In 2014 we started upgrading to 400 Mbps. Later, new MoCA chipsets promise to get able to go up to 800 Mbps, but we don't have those yet. And even if we did, 800 Mbps is *still* not 1000 Mbps.

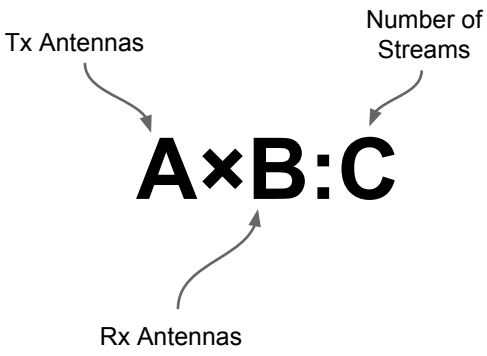
Of course, 400 Mbps (or even 200 Mbps) is still pretty fast. You can do about 40 x 5 Mbps Netflix streams with 200 Mbps. For most people, that's enough. But we're always aiming higher.

Wireless is <u>way</u> too slow

Also, it's all anyone cares about.

All that said, of course, nobody wants wires anyway. They want wireless. And current wifi standards, despite marketing hype with exaggerated claims, goes much slower than a gigabit, most of the time.

MIMO



Band

2.4 / 5 GHz

Width

20 / 40 / 80 MHz

Standard

802.11n / ac

Power Limit

Low / High

Wifi has many different parameters that affect speed, some of which I'll cover in subsequent slides.

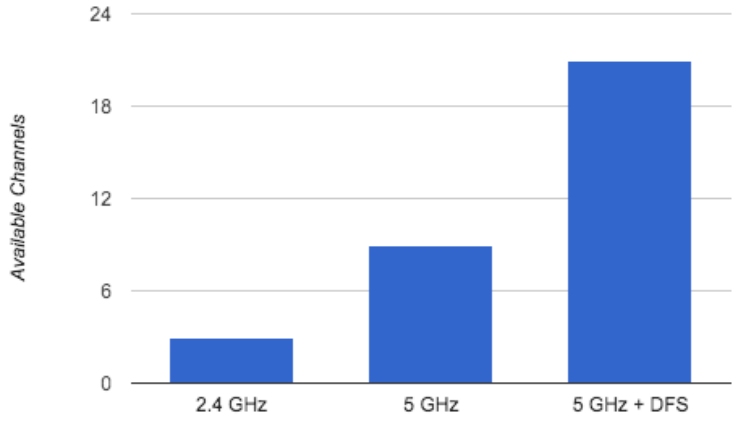
How do I get the best wifi?

- Buy a really good quality router (most suck)
- Use 802.11ac
- Use 5 GHz band
- Use a high-power-limit channel (>= 149)
- Use 80 MHz width
- Use 3x3 or better equipment

...but here's the super quick summary in case you just want to get straight to optimizing your wifi.

- Use a really good router. Most routers are average (bunched together, statistically). Some are much better than average (standing out clearly from all the rest). The latter cost extra, but the added happiness is generally worth it. Of course Google Fiber routers are in the latter category, but you probably can't have one because Google Fiber is only available in a small number of cities right now and it's probably not yours.
- Use 802.11ac, which is the latest wifi standard. It has capacity roughly 2x as much as the previous 802.11n.
- Use the 5 GHz band whenever possible. 2.4 GHz is cluttered and slow.
- Use a high-powered channel. I'll get to that in another slide, but not all 5 GHz channels are created equal. Pay attention!
- Use 80 MHz channel width (only available with 802.11ac).
- Use 3x3 or better (ie. 3 antennas) wifi equipment. That means both your access point *and* your laptop, if you want the best speeds.

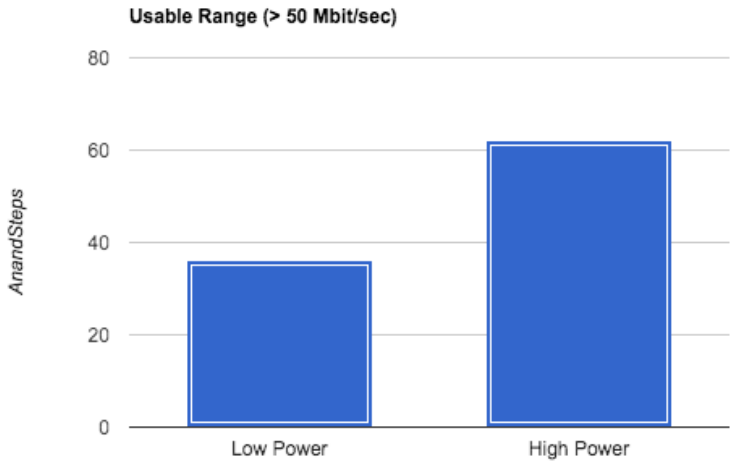
Bands: 2.4 GHz vs 5 GHz



This is a visual depiction of how many more channels are available in the 5 GHz band (and the trickier, but even less cluttered, 5 GHz “dynamic frequency selection” bands).

Edward Tufte would probably have a heart attack if he saw this chart, which we can summarize as “big bar is bigger.” Please don’t show it to him.

Channel Power Limits



Avery's First Rule of Wifi:

5 GHz goes through walls just fine.

Another similar chart, this time for channel numbers.

On the 5 GHz wifi band, in North America, low-numbered channels (rule of thumb: channels < 100) have a low maximum power limit set by legal regulations. High-numbered channels (rule of thumb: channels > 100) have a higher maximum power limit comparable to 2.4 GHz.

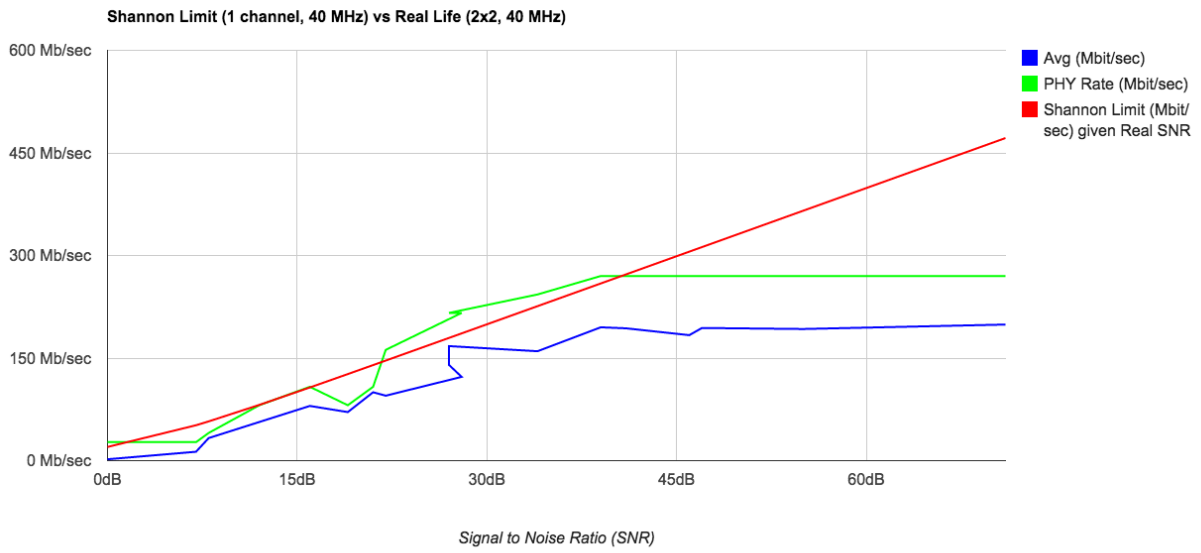
[Note: the legal limit for the low-numbered channels has been increased, at least in the US, so that all the channels can use equal power. However, most devices have not been updated for this yet. Some may never be. So it'll be safer for quite some time to use the high powered channels if you want maximum range.]

Many people believe that 5 GHz signal simply doesn't travel as far as 2.4 GHz, because it gets blocked by walls and has worse propagation characteristics. This is technically true, but not nearly as much of an impact as people tend to think. For the most part, you shouldn't even notice the difference. But what *does* matter is this power level difference.

I had my summer student, Anand Khare, test a bunch of wifi routers by setting them up and measuring how far he could walk away before the signal dropped out. The rather inaccurate unit for this measurement is what I christened "AnandSteps." As you can see, on high-powered channels you can get roughly 60% more distance in our tests. That's a big deal, and easily explains the misperception that 5 GHz doesn't travel as far.

Trivia: many wifi routers choose a channel automatically based on which channel is the least crowded. Which channel do you think will be least crowded: the low-powered ones that don’t go through walls, or the high-powered ones that do?

Minor Violations of the Laws of Physics



I'm going to just ruin the suspense and tell you now: we're not actually going to violate any laws of physics. However, modern wifi does violate some *perceived* laws of physics.

This chart plots the so-called Shannon Limit (Shannon-Hartley Theorem) in red. The Shannon Limit defines the theoretical maximum amount of data you can squeeze into a particular amount of wireless spectrum given a particular amount of signal power (strictly speaking, the ratio of signal power to the background noise, also known as the signal-to-noise ratio, SNR).

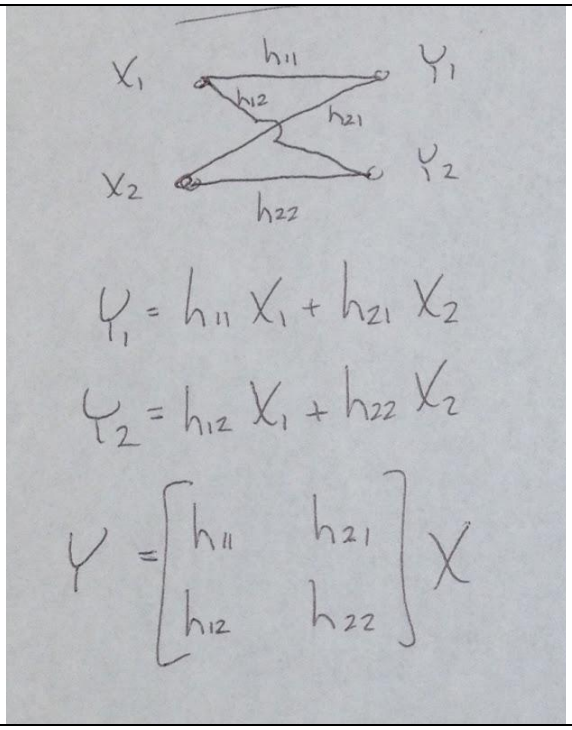
The green and blue lines show the transfer rates Anand and I observed in our tests, given different SNR measurements. The green line (PHY) rate is a technical measurement of actual bits transmitted, while the blue line is, essentially, the *useful* bits transmitted. Communication protocols always have overhead, so the difference between the two is the overhead.

The blue and green lines flatten out beyond a certain signal strength (it looks like about 40 dB to me) because the wifi standard we were testing (802.11n at the time) simply doesn't know how to go any faster. 802.11ac could have taken a bit better advantage of the extra signal power.

Of course, the neat part of this is how the green line actually goes *above* the red line - the theoretical limit - for some parts of the graph. By definition, that's theoretically impossible. But not really, it's just that traditionally, we have interpreted the theory in

a needlessly strict way. Starting with 802.11n, we re-interpreted the rules by introducing what's called MIMO.

MIMO



MIMO stands for multi-input multi-output, which is a fancy way of saying you have two or more antennas on each end, with electronics to drive each one separately.

This helpful diagram was drawn up and scanned by a wifi expert at Google when I sent him an email that was incorrect. Sending smart people assertions of incorrect things is, by the way, one of the best ways to get them to answer quickly (although it only works the first few times).

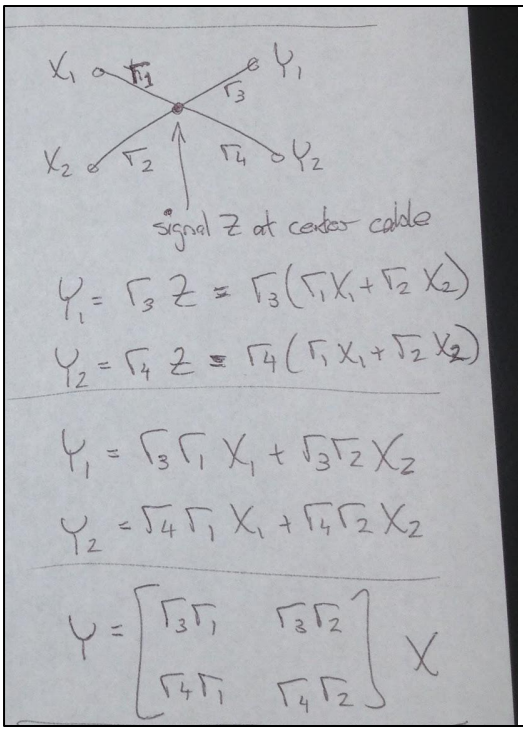
Anyway, the idea is relatively simple. If you transmit a signal from each antenna X1 and X2, and your receive antennas are Y1 and Y2, then *each* signal is subject to the Shannon limit. It's a bit like if you had two wires side by side: the signal in each one would be subject to the Shannon limit.

However, there's a catch. You don't, of course, have two wires side by side. Wireless signals radiate from X1 and X2 in all directions. So actually, Y1 receives some combination of signals X1 and X2, as you can see in the diagram. So does Y2… but a different combination. (The combination is different because with different path lengths, the radio sine waves will combine in different phases. Since the length X1->Y1 is shorter than X2->Y1, and X1->Y2 is *longer* than X2->Y2, we know Y1 and Y2 see different strengths and phases between the two.)

So on the receiver, you have to do some math to untangle things. Basically, it's two equations in two unknowns, which is solvable using a simple formula (invert a 2x2 matrix).

What's really interesting is you can keep doing this by adding more and more antennas (at the cost of more and more compute power and antenna hardware, of course). It becomes troublesome, though, because you have to be able to invert matrices pretty quickly. 2x2 and 3x3 matrix inversion is pretty easy, but larger ones get progressively more complicated.

Still, you can't argue with results: wifi speeds increase about linearly with the number of MIMO streams. So a 3x3 device is 3x faster than a 1x1 device.



1-Wire MIMO

$$\det(H) = (\Gamma_3\Gamma_1)(\Gamma_4\Gamma_2) - (\Gamma_3\Gamma_2)(\Gamma_4\Gamma_1)$$
$$= \Gamma_1\Gamma_2\Gamma_3\Gamma_4 - \Gamma_1\Gamma_2\Gamma_3\Gamma_4$$
$$= \phi$$

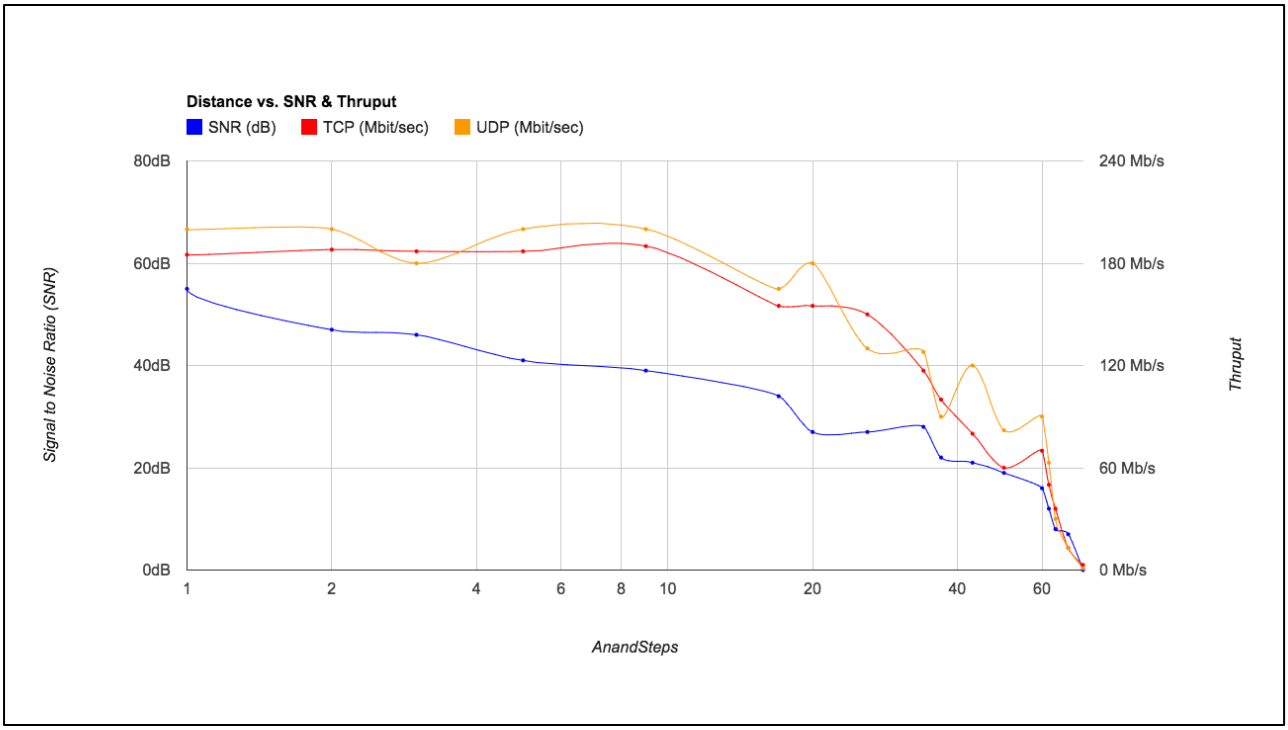
Beamforming

apenwarr.ca/beamlab

Beamforming is a bit like MIMO, but does a different trick with the phases between multiple signals. Instead of sending completely different signals out from X1 and X2, we send the same signal, but phase shifted, to try to optimize the constructive interference at Y1 or Y2, but destructive interference elsewhere. I already wrote a lot about beamforming here: http://apenwarr.ca/log/?m=201408#01

Interestingly though, because of the way signals work, you should be able to do MIMO and beamforming at the same time. You "simply" calculate X1a and X2a phases for the first signal, and different X1b and X2b phases for the second signal, and add them together linearly to produce the complete X1 and X2. In theory.

In practice, it's pretty hard to get all the phases right. But that's why radio engineers have jobs!



One last thing about wifi: all the stuff about theoretical maximum wifi speeds is based on having a really good signal to noise ratio. As you move away from the access point, the signal strength drops, and so your speed drops. With Google Fiber, of course, even the close-up scenario is slower than 1000 Mbps, and it just gets slower from there.

This is a chart of 2x2 802.11n (a kind of old standard, but pretty typical in customer devices right now) as Anand walks away from a wifi access point. Notice how things are pretty good for a while (SNR is high enough that 802.11n is maxed out) and then speed starts to dive.

This chart doesn't feature any walls, but note that a single wall really cuts your SNR a lot even though it's not very many AnandSteps away.

On the other hand, things are pretty good up to 10 steps away or so, which can cover a small-ish living room. So there's hope… if you can get a router into every room.

Wifi and Asymmetry

isoping:
github.com/dtaht/isochronous

I didn't have much time to talk about this, but note that wifi speeds tend to be different in the upstream and downstream directions. That's because the transmitter in your phone is constrained by battery life, so doesn't use such high power, whereas the access point is plugged into the wall. Transmitting at lower power means the SNR is less, so the Shannon limit is less, and so on.

The good news is in terms of raw number of bytes, most people mostly download. So at least the direction with the most speed is the one that most people want to have the most speed.

Anyway, isoping is a program I wrote that can measure the difference between packet delays and loss in the upstream and downstream direction. This is great for determining whether it's your client device or your server that's having trouble getting packets through.

Questions?

Avery Pennarun <apenwarr@google.com>

- tinyurl.com/bwlimits (and read up on SPDY and QUIC)
- gfblip.appspot.com latency tester
- apenwarr.ca/beamlab beamforming simulator
- github.com/dtaht/isochronous asymmetry tester
- GFiber devices are mostly open source!
  gfiber.googlesource.com