Nuts and Bolts of Network Neutrality[1]

Edward W. Felten

Center for Information Technology Policy
Department of Computer Science, and
Woodrow Wilson School of Public and International Affairs
Princeton University

Version of July 6, 2006
felten@cs.princeton.edu

Network neutrality is a vexing issue. Proponents of neutrality regulation argue that the free, innovative Internet of today is threatened and government action is needed to protect it. Opponents argue that regulation is not needed, or will be flawed in practice, or is a bad idea even in principle.

One of the reasons the network neutrality debate is so murky is that relatively few people understand the mechanics of network discrimination. In reasoning about net neutrality it helps to understand the technical motivations for discrimination, the various kinds of discrimination and how they would actually be put into practice, and what countermeasures would then be available to users and regulators. These are what I want to explain in this essay.

It's not my goal to answer every question about net neutrality—that would require a book, not an essay. What I want to do is fill in some of the technical background in a way that illuminates the core issues, in the hope of providing a little clarity to the discussion.

1 Intelligence at the Edges vs. in the Middle

The Internet consists of a set of end-user computers connected by infrastructure that carries data between those computers. This infrastructure is basically a set of routers (think: metal boxes with electronics inside) connected by links (think: long wires). Packets of data get passed from one router to another, via links. A packet is forwarded from router to router, until it arrives at its destination.

[1] This paper is available online at http://itpolicy.princeton.edu/pub/neutrality.pdf. Thanks to Alex Halderman and Cameron Wilson for feedback on earlier versions.

The Internet is unusual among networks in putting most of the intelligence in the computers at the edge of the network, rather than in the infrastructure at the heart of the network. The routers in the middle forward packets with only minor processing—all the heavy lifting takes place on the transmitting and receiving computers. This approach of putting intelligence at the edge of the network is known as the end-to-end principle, and it is one of the keys to the Internet's success thus far.

Putting the intelligence in the edge computers has several advantages. (1) Edge computers account for most of the devices involved in the network, so the edge computers collectively have most of the memory and processing power available to the network, and it makes sense to put the intelligence where these resources are available. (2) Edge computers have a better idea what the network's users want, because they are owned and controlled directly by users. (3) Innovation usually happens faster at the edge of the network.

In a sense, the net neutrality debate is a fight between the edges and the middle over control of the network. Neutrality regulation is generally supported by companies that provide services at the edge of the network, and is generally opposed by companies that manage the middle of the network. Each group wants the part of the network that it controls to have most of the intelligence, because more opportunities to innovate—and profit from innovation—are available to those who control the intelligent parts of the network.

Take-home lesson:

THIS IS PARTLY A FIGHT TO CONTROL INNOVATION ON THE INTERNET.

2 Minimal vs. Non-minimal Discrimination

Focus now on a single router (in the "middle" of the network). It has several incoming links on which packets arrive, and several outgoing links on which it can send packets. When a packet shows up on an incoming link, the router determines on which outgoing link the packet should be forwarded. If that outgoing link is available, the packet can be sent out on it immediately. But if the outgoing link is busy transmitting another packet, the newly arrived packet will have to wait—it will be "buffered" in the router's memory, waiting its turn until the outgoing link is free.

Buffering lets the router deal with temporary surges in traffic. But if packets keep showing up faster than they can be sent out on some outgoing link, the number of buffered packets will grow and grow, and eventually the router will run out of buffer memory. At that point, if one more packet shows up, the router has no choice but to discard a packet. It can discard the newly arriving packet, or it can make room for the new packet by discarding an older packet waiting in the buffer, but something has to be discarded.[2]

When a router is forced to discard a packet, it can discard any packet it likes. One possibility is to assign priorities to the packets, and always discard the packet with lowest priority. This mechanism defines one type of network discrimination, which prioritizes packets and discards low-priority packets first, but only discards packets when that is absolutely necessary. I'll call it *minimal discrimination*, because it only discriminates when it can't serve everybody. With minimal discrimination, if the network is not crowded, lots of low-priority packets can get through. Only when there is an unavoidable conflict with high-priority packets is a low-priority packet inconvenienced.

In contrast, there is another, more drastic form of discrimination, in which routers discard some low-priority packets even when it is possible to forward or deliver every packet. A router might, for example, limit low-priority packets to 20% of the network's capacity, even if part of the other 80% is idle. I'll call this *non-minimal discrimination*. One of the basic questions to ask about any network discrimination regime is whether it is minimal or non-minimal in this sense, and one of the basic questions to ask about any rule limiting discrimination is how it applies to minimal versus non-minimal discrimination. We can imagine a policy, for example, that allows minimal discrimination but limits or bans non-minimal discrimination.

This distinction matters, I think, because minimal and non-minimal discrimination are supported by different arguments. Minimal discrimination sometimes may be an engineering necessity due to the finite speed of network links, but non-minimal discrimination is never technologically necessary—it makes service worse for low-priority packets, but doesn't help high-priority packets. Non-minimal discrimination can only be justified by a more complicated economic argument, for example that non-minimal discrimination allows forms of price discrimination that increase social welfare; vague arguments that network operators have to reserve some fraction of capacity for some purpose won't cut it.

Take-home lesson:

DISCRIMINATION HAS HARSHER AND MILDER FORMS. BLOCKING A PACKET IS HARSHER THAN JUST LOWERING ITS PRIORITY.

3 Delay Discrimination

Discrimination doesn't have to operate by dropping packets. It can also work by reordering packets.

[2] This is an illustration of the "best effort" principle, one of the clever engineering decisions that make the Internet feasible. The Internet will do its best to deliver each packet promptly, but it doesn't make any guarantees. It's up to software on the end computers to detect dropped packets and recover. Your computer's software can, and probably often does, recover from dropped packets.

Recall that packets sometimes have to be buffered (i.e., to wait) at a router if they need to be sent over an outgoing network link that is busy. When an outgoing link becomes available, there may be several buffered packets that are waiting to be transmitted on that link. You might expect the router to send the packet that has been waiting the longest—a first-come, first-served rule. Often that is what happens, but the Internet Protocol doesn't require routers to forward packets in any particular order. In principle a router can choose any packet it likes to forward next. This suggests an obvious mechanism for discriminating between two categories of traffic: a network provider can program its routers to always forward high-priority packets before low-priority packets. Low-priority packets feel this discrimination as an extra delay in passing through the network.

The distinction between minimal and non-minimal discrimination applies here too. A minimal form of delay discrimination only delays low-priority packets when it is necessary to delay some packet—for example when multiple packets are waiting for a link that can only transmit one packet at a time. There is also a non-minimal form of delay discrimination in which a low-priority packet may be delayed even when the link it needs is available. As before, a net neutrality rule might want to treat minimal and non-minimal delay discrimination differently.

One interesting consequence of minimal delay discrimination is that it hurts some applications more than others. Internet traffic is usually bursty, with periods of relatively low activity punctuated by occasional bursts of packets. When you browse the Web, for example, you generate little or no traffic while you're reading a page, but there is a burst of traffic when your browser needs to fetch a new page from a server. If a network provider is using minimal delay discrimination, and the high-priority traffic is bursty, then low-priority traffic will usually sail through the network with little delay, but will experience noticeable delay whenever there is a burst of high-priority traffic. The technical term for this kind of on-again, off-again delay is "jitter."

Some applications can handle jitter with no problem. If you're downloading a large file, you care more about the average packet arrival rate (the download speed) than about when any particular packet arrives. If you're browsing the web, modest jitter will cause, at worst, a slight delay in downloading some pages. If you're watching a streaming video, your player will buffer the stream so jitter won't bother you much. On the other hand, applications like online gaming or Internet telephony (VoIP), which rely on steady streaming of interactive, realtime communication, can suffer a lot if there is jitter. Users report that VoIP services like Vonage and Skype can become unusable when subjected to network jitter.

Since residential Internet Service Providers (ISPs) are often phone companies, or at least offer home phone service, they may have a special incentive to discriminate against competing Internet phone services. Causing jitter for such services, whether by minimal or non-minimal delay discrimination, could be an effective tactic for an ISP that wants to drive customers away from independent Internet telephone services.

Take-home lesson:

DISCRIMINATION HURTS SOME APPLICATIONS MORE THAN OTHERS.
VOIP SERVICES ARE ESPECIALLY VULNERABLE TO DISCRIMINATION.

4 Detecting Discrimination

The kinds of discrimination I have just described will often be experienced by users as decreased network performance. However, as the following hypothetical example illustrates, it is often difficult to distinguish between performance problems resulting from undesirable forms of discrimination and ones due to other causes.

Suppose we discover that customers of TelCo, a residential ISP, are having trouble using the VoipCo Internet phone service, because of jitter problems. What might be causing this? One possibility is that TelCo is using delay discrimination, either minimal or non-minimal, with the goal of causing this problem. Many people would want rules against this kind of behavior.

Another possibility is that TelCo isn't trying to cause problems for VoipCo users, and in fact TelCo's management of its network is completely reasonable and nondiscriminatory, but for reasons beyond TelCo's control its network happens to have higher jitter than other networks have. Perhaps the jitter problems are temporary. In this case, most people would agree that net neutrality rules shouldn't punish TelCo for something that isn't really its fault.

The most challenging possibility, from a policy standpoint, is that TelCo didn't take any obvious steps to cause the problem but is happy that it exists, and is subtly managing its network in a way that fosters jitter. Network management is complicated, and many management decisions could impact jitter one way or the other. A network provider who wants to cause high jitter can do so, and might have pretextual excuses for all of the steps it takes. Can regulators distinguish this kind of stratagem from the case of fair and justified engineering decisions that happen to cause a little temporary jitter?

Surely some discriminatory strategies are so obvious, and the offered engineering pretexts so weak, that we could block or punish them without worrying about being wrong. But there would be hard cases too. Net neutrality regulation, even if justified, will inevitably lead to some difficult line-drawing.

There is a useful analogy to employment discrimination.[3] Company A might say, "We won't hire women." Company B might say (falsely) that it is perfectly willing to hire a woman if she is the best-qualified candidate, but might in fact seek out reasons not to hire a woman in every particular case. Company C might have no intention of discriminating but might follow policies that have the unintended side effect of causing

[3] In making this analogy, I'm not claiming any kind of moral equivalence between employment discrimination and network discrimination. That would be silly—packets are not people. The point of the analogy is simply that anti-discrimination rules raise difficult enforcement issues.

fewer women to be hired. Company D might have adopted those same policies with the intent of discriminating. Company E might behave in an entirely fair and evenhanded way but have relatively few women on its payroll due to chance or other factors beyond its control. The blatant discrimination of Company A is easy to detect and address, but it could be difficult in practice to tell Companies B, C, D, and E apart. An enforcement regime that tries to distinguish them will be costly and will make some errors. This does not necessarily tell us not to establish such an enforcement regime, but it does give us reason to think carefully before doing so.

Take-home lesson:

ANTI-DISCRIMINATION RULES CAN BE HARD
TO WRITE, AND HARD TO ENFORCE.

5 Discrimination, Congestion, and Cooperation

Let's turn now to how the Internet responds to congestion, and how network discrimination might affect that response. I described previously how network congestion causes Internet routers to discard some data packets. Every dropped packet has some computer at the edge of the network waiting for it. Eventually the waiting computer and its communication partner will figure out that the packet must have been dropped, and from this they will deduce that the network is congested. So they will re-send the dropped packet, but in response to the probable congestion they will slow down the rate at which they transmit data. Once enough packets are dropped, and enough computers slow down their packet transmission, the congestion will clear up.

This is a very indirect way of coping with congestion—drop packets, wait for endpoint computers to notice the missing packets, and respond by slowing down—but it works pretty well. One interesting aspect of this system is that it is voluntary—the system relies on endpoint computers to slow down when they see congestion, but nothing forces them to do so. We can think of this as a kind of deal between endpoint computers, in which each one promises to slow down if its packets are dropped. (Notice that this is another application of the end-to-end principle we discussed earlier.)

But there is an incentive to defect from this deal. Suppose that you defect—when your packets are dropped you keep on sending packets as fast as you can—but everybody else keeps the deal. When your packets are dropped, the congestion will continue. Then other people's packets will be dropped, until enough of those people slow down and the congestion eases. By ignoring the congestion signals you are getting more than your fair share of the network resources.

Despite the incentive to defect, most people keep the deal by using networking software that slows down as expected in response to congestion. Why is this? One way to look at it is that there is a sort of social contract by which users cooperate with their peers, and software vendors cooperate by writing software that causes users to keep the deal.

One of the reasons users comply, I think, is a sense of fairness. If I believe that the burdens of congestion control fall about equally on everybody, at least in the long run, then it seems fair to me to slow down my own transmissions when my turn comes. One time I might be the one whose packets get dropped, so I will slow down. Another time, by chance, somebody else's packets may be dropped, so it will be their turn to slow down. Everybody gets their turn.[4]

But now suppose that the network starts singling out some people and dropping their packets first. Now the burden of congestion control falls heavily on them—they have to slow down and others can just keep going. Suddenly the I'll-slow-down-if-you-do deal doesn't seem so fair, and the designated victims are more likely to defect from the deal and just keep sending data even when the network tells them to slow down.

The implications for network discrimination are clear. If the network discriminates by sending misleading signals about congestion, and sending them preferentially to certain machines or certain applications, the incentive for those machines and applications to stick to the social contract and do their share to control congestion will weaken. Will this lead to a wave of defections that destroys the Net? Probably not, but I can't be sure. I do think this is something we should think about.

We should also listen to the broader lesson of this analysis. If the network discriminates, users and applications will react by changing their behavior.

Take-home lesson:

NETWORK DISCRIMINATION WILL HAVE UNPREDICTABLE EFFECTS.

6 Encryption as a Countermeasure

Scenarios for network discrimination typically involve an ISP that looks at users' traffic and imposes delays or other performance penalties on certain types of traffic. To do this, the ISP must be able to tell the targeted data packets apart from ordinary packets. For example, to penalize VoIP traffic, the ISP will want to distinguish VoIP packets from ordinary packets.

Normally, the ISP can distinguish VoIP packets by looking for characteristic values at certain places in the packet. One way for users to fight back is to encrypt their packets, on the theory that encrypted packets will all look like gibberish to the ISP, so the ISP won't be able to tell one type of packet from another.

To do this, the user would probably use a Virtual Private Network (VPN). Whenever the user's computer wanted to send a packet, it would encrypt that packet and then send the encrypted packet to a "gateway" computer that was outside the ISP's network. The

[4] I'm not claiming that the average user has thought through these issues carefully. But many software providers have made decisions about what to do, and those decisions factor in users' wants and needs. Software developers act as proxies for users in making these decisions.

gateway computer would then decrypt the packet and send it on to its intended destination. Incoming packets would follow the same path in reverse—they would be sent to the gateway, where they would be encrypted and forwarded on to the user's computer. The ISP would see nothing but a bi-directional stream of packets, all encrypted, flowing between the user's computer and the gateway.

The most the user can hope for from a VPN is to force the ISP to handle all of the user's packets in the same way. The ISP can still penalize *all* of the user's packets, or it can single out randomly chosen packets for special treatment, but those are the only forms of discrimination available to it. The VPN has some cost—packets must be encrypted, decrypted, and forwarded—but the user might consider the cost worthwhile if it stops the ISP's network discrimination.

(In practice, things are a bit more complicated. The ISP might be able to infer which packets are which by observing the size and timing of packets. For example, a sequence of packets, all of a certain size and flowing with metronome-like regularity in both directions, is probably a voice conversation. The user might use countermeasures, such as altering the size and timing of packets, but that can be costly too. To simplify our discussion, let's pretend that the VPN gives the ISP no way to distinguish packets from each other.)

The VPN user and the ISP are playing an interesting game of chicken. The ISP wants to discriminate against some of the user's packets, but doesn't want to inconvenience the user so badly that the user discontinues the service (or demands a much lower price). The user responds by making his packets indistinguishable and daring the ISP to discriminate against all of them. The ISP can back down, by easing off on discrimination in order to keep the user happy—or the ISP can call the user's bluff and hamper all or most of the user's traffic.

But the ISP can use a different and more effective strategy. If the ISP wants to hamper a particular application, and there is a way to manipulate the user's traffic that affects that application much more than it does other applications, then the ISP has a way to punish the targeted application. Recall from earlier that VoIP is especially sensitive to jitter (unpredictable changes in delay), but most other applications can tolerate jitter without much trouble. If the ISP imposes jitter on all of the user's packets, the result will be a big problem for VoIP services, but will not have much impact on other applications.

Attempts by ISPs to discriminate, and by users to evade discrimination, lead to a technical battle of measure and countermeasure that can have harmful effects. Resources are wasted, on both sides, and collateral damage is possible. Consider the example above, where an ISP blocks or degrades encrypted traffic, in order to keep customers from using encryption to evade the ISP's packet classifiers. In doing this, the ISP is effectively imposing a performance tax on the use of encryption. This will cause users to encrypt less, which will put their security and privacy at risk. After all, any packet that can be inspected by the ISP can also be inspected by an intruder.

Take-home lesson:

TECHNICAL COUNTERMEASURES, SUCH AS ENCRYPTION, CANNOT FULLY SHIELD USERS FROM DISCRIMINATION.

7 Quality of Service

One of the standard arguments against network neutrality rules is that network providers need to provide Quality of Service (QoS) guarantees to certain kinds of traffic, such as video. If QoS is necessary, the argument goes, and if net neutrality rules would hamper QoS by requiring all traffic to be treated the same, then net neutrality rules must be harmful. In this section, I want to unpack this reasoning and see how it holds up in light of computer science research and engineering experience.

First, I need to make clear that guaranteeing QoS for an application means more than just giving it lots of bandwidth or prioritizing its traffic above other applications. Those things might be helpful, but they're not QoS (or at least not the kind I'm talking about here). What QoS mechanisms (try to) do is to make specific performance guarantees to an application over a short window of time—in other words, they want not just good performance on average, but performance that is smooth and predictable.

An example may clarify this point. As discussed above, some applications are more sensitive to jitter than others. If you're loading a web page, and your network connection hiccups so that you get no traffic for (say) half a second, you may notice a short pause but it won't be a big deal. But if you're having a voice conversation with somebody, a half-second gap will be very annoying. Web browsing needs decent bandwidth on average, but voice conversations needs better protection against short delays. That protection is QoS.

The reason we don't need special QoS mechanisms for browsing is that the broadband Internet already provides performance that is almost always steady enough over the time intervals that matter for browsing. Sometimes, too, there are simple tricks that can turn an application that cares about short delays into one that cares only about longer delays. For example, watching prerecorded audio or video streams doesn't need QoS, because you can use buffering. If you're watching a video, you can download every frame ten seconds before you're going to watch it; then a hiccup of a few seconds won't be a problem. This is why streaming audio and video work perfectly well today (when there is enough average bandwidth).

There are two other important cases where QoS isn't needed. First, if an application needs higher average speed than the Net can provide, than QoS won't help it—QoS makes the Net's speed steadier but not faster. Second—and less obvious—if an app needs much *less* average speed than the Net can provide, then QoS might also be unnecessary. If speed doesn't drop entirely to zero but fluctuates, with peaks and valleys, then even the valleys may be high enough to give the application what it needs. This is starting to happen for voice conversations—many VoIP systems seem to work pretty well without any special QoS support in the network.

We can't say that QoS is *never* needed, but experience does teach that it's easy, especially for non-experts, to overestimate the importance of QoS. That's why I'm not convinced—though I could be, with more evidence—that QoS is a strong argument against net neutrality rules.

Take-home lesson:

QUALITY OF SERVICE (QOS) GUARANTEES ARE LESS IMPORTANT THAT YOU MIGHT THINK.

8 Should We Adopt a Network Neutrality Policy?

Readers looking here for a simple policy prescription will be disappointed. The network neutrality issue is more complex and subtle than most of the advocates on either side would have you believe. Net neutrality advocates are right to worry that ISPs can discriminate—and have the means and motive to do so—in ways that might be difficult to stop. Opponents are right to say that enforcing neutrality rules may be difficult and error-prone. Both sides are right to say that making the wrong decision can lead to unintended side-effects and hamper the Internet's development.

There is a good policy argument in favor of doing nothing and letting the situation develop further. The present situation, with the network neutrality issue on the table in Washington but no rules yet adopted, is in many ways ideal. ISPs, knowing that discriminating now would make regulation seem more necessary, are on their best behavior; and with no rules yet adopted we don't have to face the difficult issues of line-drawing and enforcement. Enacting strong regulation now would risk side-effects, and passing toothless regulation now would remove the threat of regulation. If it is possible to maintain the threat of regulation while leaving the issue unresolved, time will teach us more about what regulation, if any, is needed.