Why the Internet only just works

M Handley

The core Internet protocols have not changed significantly in more than a decade, in spite of exponential growth in the number of Internet users and the speed of the fastest links. The requirements placed on the net are also changing, as digital convergence finally occurs. Will the Internet cope gracefully with all this change, or are the cracks already beginning to show? In this paper I examine how the Internet has coped with past challenges resulting in attempts to change the architecture and core protocols of the Internet. Unfortunately, the recent history of failed architectural changes does not bode well. With this history in mind, I explore some of the challenges currently facing the Internet.

1. Introduction

The Internet only just works. I want to make it clear though, right from the start, that this is not a forecast of imminent doom and disaster. My reasons for making this assertion are twofold. Firstly, I believe that this has historically been the natural state of the Internet and it is likely to remain so in future. Unless this is understood, then it is hard to understand which problems are really cause for concern, and which we can safely ignore or put off solving till some later date. Secondly, I want to discuss some problems that should be cause for concern, at least in the medium term.

2. The natural state of affairs

If we look back at the history of the Internet, the story is one of constant change. Indeed the phrase 'Internet time' is often used to express just how fast things change. But if we look at the core protocols that comprise the Internet at the lower levels of the stack, change has been comparatively slow and carefully considered.

2.1 1970-1993 — a history of change

The first large-scale packet switching network was the ARPAnet, which was used to come to grips with the main architectural issues that would go on to be the basis of the Internet. The basic protocol that underlay the ARPAnet was NCP [1], which combined addressing and transport into a single protocol. Many of the higher-level protocols that would go on to become common on the Internet were first deployed on the ARPAnet. The most obvious are remote log-in, e-mail, and file transfer, but there were also ARPAnet experiments with packet voice, which predate common usage of voice-over-IP by over twenty years.

The ARPAnet was very successful, but it was also clear that flexibility should be of prime importance in the design of a general-purpose successor [2], and as a result reliability was separated from addressing and packet transfer in the design of the Internet protocol suite, with IP being separated from TCP. The switchover to TCP/IP culminated in a *flag-day* on 1 January 1983 when all remaining ARPAnet nodes switched. There were approximately four hundred nodes; this was probably the last time such a flag-day was possible, and every change since then has needed to be incrementally deployable.

Changing a large network is very difficult. It is much easier to deploy a novel new protocol that fills a void than it is to replace an existing protocol that more or less works. Change is, however, possible when the motivation is sufficient. In 1982 the domain name system (DNS) was deployed, replacing the original `hosts.txt` file [3] for naming Internet systems. This was a clear response to a scaling problem, but the necessity for change was obvious, and the DNS not only solved the basic issue of distributing files of host names, but also allowed the change to decentralised administration of the namespace. Decentralised administration is one of the basic enablers of the rapid

growth of the Internet as a whole, so this change must have seemed inevitable[1]. Basically it was clear that maintaining `hosts.txt` was unfeasible as the Internet grew, so just around the time it stopped scaling a replacement was deployed. A decentralised naming system could have been developed years earlier, but there was no pressing need. Only as the scaling limits of the previous system were reached was the replacement deployed.

Another example of scaling problems comes in routing protocols. Link-state routing protocols [4] were developed as a direct response to the convergence problems suffered by distance-vector routing protocols as the Internet grew in size. Furthermore, the Exterior Gateway Protocol (EGP) [5] was a direct response to the scaling limitations of intra-domain routing, allowing routing within a network to be partially isolated from routing between networks. Each of these was a substantial change to the core protocols of the Internet, and was driven primarily by scaling problems as the previous generation of routing protocols started to reach its limits.

No-one likes changing such a key part of an operational network — such changes are driven by necessity. However, as the Internet was not a key infrastructure in the 1980s, the pain caused during such transitions was comparatively low. Besides, many people regarded the Internet as an interim solution that would eventually be replaced by the OSI protocols, and therefore, with the glare of political attention diverted elsewhere, the engineers of the Internet were allowed to do good engineering. They learned from their mistakes by trying things out for real and fixed problems as they became pressing.

In the mid-1980s the Internet suffered from a series of congestion collapses, where the network was operating at full capacity moving packets around, but no *useful* work was being done. This was one of the rare cases where something important actually failed *before* being fixed. The problem was TCP's retransmission strategy. Under certain circumstances the network could become completely clogged with packets which were unnecessarily retransmitted, to the point where no connection made useful progress.

Congestion is essentially a network-level problem rather than a transport-level problem, as both UDP and TCP flows can cause congestion. In reality what can usefully be controlled is the rate of transmission of a *flow*, but there is no generic 'session' level in the TCP/IP protocol stack. Perhaps the correct solution might have been to add a new layer to the stack to handle congestion in a protocol-independent manner, but this would have been a difficult change to deploy incrementally. Another possible solution might have been to use per-flow queuing, but this would have been expensive to deploy on the router hardware of the time. The simplest solution was to fix the immediate problem, and so TCP's congestion control mechanism was born [6]. This was backwards compatible, incrementally deployable, and did an excellent job of addressing the immediate problem at hand. There is no doubt that over the nearly two decades that followed, the safe adaptation provided by TCP congestion control has been instrumental in ensuring that the network has survived its growing pains in the many situations where demand outstripped capacity, whatever the reason.

Despite its success, TCP's congestion control mechanism was never intended to be the only component of congestion control on the Internet. How could it be? It only manages TCP flows, and while these have always comprised the majority of Internet traffic, other protocols also exist and are equally capable of causing congestion. In many ways then, performing congestion control in TCP is a suboptimal place to solve the problem because it is insufficiently general, just as NCP was insufficiently general. However, by 1988 the Internet was already large enough that it was hard to change the core protocols, and solving the problem in TCP was *good enough*.

In the early 1990s the National Science Foundation funded a new Internet backbone connecting academic institutions in the USA. However, the Internet was already becoming a commercial enterprise, and commercial traffic was prohibited from using the NSFnet backbone. Thus was born the need for *policy routing*, whereby each network could decide for itself which routes to use and to propagate to other networks, while the network as a whole still maintained routing integrity. The requirements had changed, and by force of necessity, the routing protocols needed to change too, or the Internet would cease to function effectively. The Border Gateway Protocol (BGP) [7] was the result, and versions of BGP have been used for inter-domain routing ever since.

The final change to the core of the Internet was more subtle. The original Internet addressing scheme divided unicast addresses into three classes of subnet — A, B, and C, with 16m, 65k and 256 addresses respectively in a subnet of each class. The problem was that while class A subnets were larger than most organisations needed and class C subnets were smaller than needed, class B subnets were just about right, and this space was rapidly becoming exhausted. The solution was to abandon classful addressing altogether, and transition to a classless routing scheme based on

[1]Although the precise details of the protocol to do this are far from inevitable.

explicit specification of the address prefix length in routes. This necessitated changing all the end-hosts and all the routers on the Internet. For the most part, the end-host changes could be postponed pending the natural OS upgrade cycle, as the changes could be deployed in a backwards compatible manner. From the routing point of view, the changes were quite significant, requiring a change to both the forwarding plane to implement longest-prefix match, and to the routing plane, switching to BGP4 which propagates prefix lengths along with the subnet addresses. Fortunately most of the backbone routers at the time were from one vendor (Cisco), with forwarding performed in software, so the upgrade was relatively simple compared to the hardware forwarding planes frequently used today. In addition, the Classless Inter-Domain Routing (CIDR) changes were backwards compatible, as initially the prefix length could be assumed if it was not specified. Thus, although this was a substantial change, it was not too painful in practice.

If we consider the growth of the Internet (Fig 1), what is remarkable is not that scaling problems existed, but that the problems were not much worse than they were. When ISPs are faced with growth curves like these, there is very little room for long-term thinking; just surviving the next 18 months without making a major tactical error is difficult enough. It is therefore not at all surprising that for much of its early life the Internet only just worked. A whole succession of technical problems arose, primarily due to scaling issues, and most of them were solved *just in time* to avoid a serious melt-down.

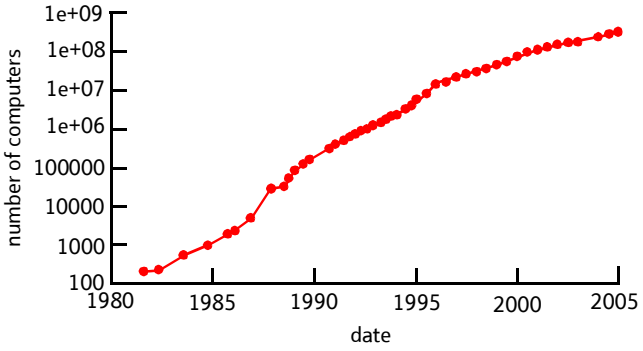

Fig 1 Exponential growth of the Internet.

2.2 *1993 to the present — the failures*

CIDR was introduced in 1993, which roughly coincided with the Internet transitioning from being primarily an academic network to being primarily a commercial network. One of the main drivers of this transition was of course the World Wide Web, which really started to take off in 1993 with the release of NCSA Mosaic.

Since 1993, there has been no significant change to the core protocols that form the basis of the Internet. This is not to say that there has been no change — in fact there have been a large number of small tweaks to TCP and BGP and most of the other core protocols, and the physical-layer mechanisms that underlie the Internet have changed radically. But the core protocols remain almost unchanged nonetheless, with mechanisms such as MPLS [8] sometimes being used to work around some of the limitations of the Internet protocols within an ISP.

Several attempts have been made to change the core protocols or enhance their functionality. In the area of congestion control, Explicit Congestion Notification (ECN) [9] was standardised, but is not widely deployed. In the area of quality of service, an entire framework known as Integrated Services was standardised [10]. When this failed to be widely deployed, an alternative framework known as Differentiated Services (DiffServ), [11] was standardised. This too is hardly ubiquitous as an end-to-end service model, although it is used within some ISPs.

Mobile systems are becoming increasingly common, and Mobile IP was devised to allow a computer to roam while still being accessible at a known IP address. But although Mobile IP has been an RFC for ten years now, very few networks support Mobile IP for visiting hosts, and no mainstream operating system ships by default with Mobile IP support.

Finally IP Multicast showed great promise in the early 1990s, as the solution to Internet-scale broadcasting. Again IP Multicast is widely used within ISPs and on LANs for local traffic, but end-to-end IP Multicast service is rare today. I believe that this may well be remedied, but only time will tell.

What do IP Multicast, Mobile IP, quality of service, and ECN have in common? They are all core network technologies that solve real problems that are not immediately pressing. They would most likely be described as *enhancements* rather than fixes to the architecture.

In a commercial network, new technologies essentially get deployed for reasons of fear or greed. Either an ISP can see a way to make money from the technology in the short term, or if the lack of deployment of a technology will cause a company to fail, then the technology will be deployed — and only then if it is incrementally deployable.

The main conclusion to draw from this is that technologies get deployed *in the core of the Internet* when they solve an immediate problem or when money

can be made. Money-making changes to the core of the network are rare indeed — in part this is because changes to the core need to be interoperable with other providers to make money, and changes that are interoperable will not differentiate an ISP from its competitors. Thus fear seems to dominate, and changes have historically been driven by the need to fix an immediate issue. Solutions that have actually been deployed in the Internet core seem to have been developed just in time, perhaps because only then is the incentive strong enough. In short, the Internet has at many stages in its evolution *only just worked*.

2.3 The future — stagnation?

Given that almost all extensions to the core of the Internet since 1993, despite having been standardised, have failed to be widely deployed, what does this say about the state of the Internet today? Two significant changes have occurred in the last decade, but both are below the IP layer. These are MPLS and VPNs [12]. Perhaps most importantly, link speeds have increased immensely, both in the core and in access networks. However, from the point of view of anything happening at the IP layer or above, the Internet has been essentially unchanged for more than a decade.

If we look at the IP layer, as we have already seen, the main extensions to IP such as multicast, QoS, and IP mobility have failed to be widely deployed. It is of course possible that this is simply that they were too early, and that the demand for these extensions has not yet become strong enough. Only time will tell, but to date the evidence is not good. Change at the IP layer is just really difficult. But surely change is easier higher up the stack?

From the end-to-end point of view, what is most noticeable is the decrease in transparency of the Internet. The main reason for this is security — the Internet has become a much more hostile environment, and firewalls are now an everyday part of the Internet connectivity story. Network firewalls are, by themselves, only a band-aid on security. It is all too easy for laptop computers to bring hostile code inside the firewall. Nevertheless, like a moat around a castle, they do form a useful part of defence in depth, and so are here to stay.

Network address translators (NATs) also inhibit transparency. NATs are interesting because they actively modify end-to-end traffic flows, while not being an explicit part of the Internet architecture. The primary reason for NATs is not, as is commonly assumed, a shortage of IPv4 addresses, although this may well become true at some point in the not-too-distant future. The primary reason for the existence of NATs is tiered pricing, whereby ISPs charge more for additional IP addresses, even though IP addresses do not in fact cost the ISP in any significant way. The number of IP addresses provided is thus used as a proxy for whether the customer is a business or a home user.

In businesses, NATs are frequently cited as a security solution. This is interesting because a NAT is in fact a very poor firewall. A NAT does, however, have one advantage over a traditional firewall, and this is that by default it fails closed. If the machines behind the firewall are not publicly addressable, then in the absence of the NAT they do not have connectivity. With a regular firewall, a misconfiguration can leave the internal network exposed. Thus, while a NAT is a poor security solution by itself, it does in fact complement a good firewall, and so NATs are unlikely to go away, even if IPv6 eventually sees widespread deployment.

Returning to the theme of change, it becomes clear that NATs and firewalls have a similar effect; it has become hard to change layer 4 too. NATs and firewalls understand TCP, but other transport protocols find them more problematic.

This is clearly an issue for UDP, as UDP-based applications, such as voice-over-IP, generally do their connection set-up using a separate signalling protocol, using UDP purely to transport audio data. This presents a problem — either the NAT needs to understand the signalling protocol and set up the appropriate forwarding state, or the application needs to understand that a NAT exists and reverse engineer [13, 14] how the NAT has jumbled the UDP ports. Both of these are ugly and error-prone solutions. If the signalling is encrypted (as we would normally wish it to be), then only the latter is possible, and even then we have to rely on heuristics and the existence of third party servers to figure out what happened.

For new transport protocols such as DCCP [15, 16] and SCTP [17], the problem is even worse. If a NAT or firewall does not understand a new protocol, there is no working around the problem. Communication will simply fail.

So, how then does a new protocol become widespread? There is a vicious circle — application developers will not use a new protocol (even if it is technically superior) if it will not work end-to-end; OS vendors will not implement a new protocol if application developers do not express a need for it; NAT and firewall vendors will not add support if the protocol is not in common operating systems; the new protocol will not work end-to-end because of lack of support in NATs and firewalls. The problem is exacerbated by the existence of NATs in the embedded firmware of devices such as DSL modems and IEEE802.11 base-stations. The

firmware of these devices is almost never upgraded by consumers, so the deployment of a new protocol depends on both the vendors implementing it in middle-boxes, followed by the previous generation of these middle-boxes expiring and being replaced. In short, a new transport protocol is not going to become widespread on a time-scale shorter than a decade, if ever.

Wasn't the transport layer supposed to be relatively easy to change, as layering ensures that IP routers don't care about the contents of packets? The recent tendency towards deep-packet-inspection can only make this problem worse.

The conclusion is that there has been no substantial change at layer 3 for a decade and no substantial change at layer 4 for nearly two decades. Clearly then the Internet is suffering from *ossification*. The original general-purpose Internet which could evolve easily to face new challenges has been lost, and replaced with one that can only satisfy applications that resemble those that are already successful. But the Internet is a great success nonetheless, so is this really a cause for concern? I firmly believe yes. The main issue is that while the Internet has not changed, the requirements placed upon it have.

3. Convergence

Digital convergence is a much hyped term, but it is finally happening. BT's 21C network architecture is one of the higher profile examples, but the transition from circuit-switched telephony to VoIP is well under way in many organisations. IPTV is also gaining momentum, and it appears likely that a large fraction of television will be distributed over IP networks within a decade.

It matters little whether we consider this convergence to be a good idea — the economic drivers are just too powerful. Faced with competition from a smaller, more agile competitor running a multi-use IP network, a traditional circuit-switched telco in a de-regulated market will have difficulty competing. A similar phenomenon is likely to occur for television distribution. It is unclear whether *all* pre-recorded television will be transmitted on-demand in a decade, as the social aspects of traditional time-synchronous broadcast cannot be ignored. However, it seems very likely that a large fraction of television will be on-demand over the Internet.

With the demise of circuit-switched telephony and traditional broadcast TV, all the main communications channels used by businesses and homes will use IP-based networks. It is likely that either these networks will use the Internet, or they will be provided as VPNs over the same networks that comprise the Internet.

This leads us to the obvious question of whether the Internet is up to the challenge? After all, technologies that were supposed to help with convergence such as QoS and IP Multicast are widely regarded as failures, and certainly not widely available as end-to-end services.

4. Architectural problems

As we have seen, the core protocols that comprise the Internet architecture have ossified, and today change is extremely difficult. At the same time, demands being placed on the Internet have continued to change, and so problems have started to accumulate. At the present time, none of these problems has become severe, but the concern is that they will become so in time. These are some of the ways in which the Internet currently only just works.

4.1 Short-term problems

4.1.1 Spam

Spam is everyone's favourite problem, and certainly the most obvious annoyance to most Internet users. Beyond traditional spam, we are also likely to see a rise in spam over Internet telephony (SPIT), which threatens to negate many of the advantages of VoIP.

There are no easy solutions to the spam problem, but at the very least the standardisation of a scheme for digitally signing the headers of e-mail messages would allow the creation of white lists of the computers of known acquaintances. This would reduce the false-positive rate of content-based spam filters, so that e-mail can once again become a high-reliability service. Spam seems to be a problem that will not go away, but which can be successfully contained, albeit at some cost.

4.1.2 Security

Security is probably the biggest imminent problem facing the Internet. At best, viruses, worms, phishing, and spyware between them risk reducing people's confidence in the network and therefore its usefulness. At worst, crime runs rampant, companies are bankrupted, and security lapses in critical systems cause widespread disruption, civil unrest, and perhaps deaths.

Security is again a problem for which there is no magic bullet. There is an arms race under way, as techniques used by attackers and defenders co-evolve. Consider for example, taint-tracking [18—21]. Recent developments permit an Internet server to be instrumented so as to track where data derived from untrusted network input has spread to in memory. If this data is ever executed, used as the value of an address in a jump or return instruction, or used directly as a

parameter to various system calls, then it is clear that the server is in the process of being compromised. Execution can be terminated, and alerts or filters produced to defend other servers against the same attack.

This is a defence technique that shows great promise, especially if it can be made to run fast enough to be performed on busy production servers. However, like many technologies, it can also be misused [22]. An attacker might run such a detector on a botnet of compromised desktop systems, with the aim of quickly learning about a new exploit that someone else has devised. That exploit can then be automatically turned into a fast worm, which can outrun the original exploit. The evolutionary pressure then forces an attacker to be very selective in his attacks, or to write the fastest worm possible so that his worm compromises more machines than that of his competitor.

Techniques such as safe languages, automated code analysis, stack protection, memory randomisation and taint-tracking each make the attacker's job harder. Still, it seems unlikely that end systems will ever be completely secure. The goal then should perhaps be to raise the bar sufficiently high that exploits are rare, and then to ensure that no one exploit can rapidly compromise a large population.

Although the situation is serious, it is also now receiving a great deal of attention from software and operating system vendors. The number of exploitable vulnerabilities discovered continues to increase, but the threat landscape appears to be changing. Several years ago, most exploits were stack overflows in C programs. Today these are rarer as programmers have become better at avoiding risky programming constructs such as `strcpy` and `sprintf`. Vulnerabilities such as cross-site scripting and code injection in scripting languages have taken over as the most common vulnerabilities. This is progress of a sort.

4.1.3 Denial-of-service attacks

The Internet was designed to move packets from A to B as fast as possible, irrespective of whether B actually wants those packets. This is at the heart of flooding denial-of-service (DoS) attacks, which have seen increasing prevalence in recent years. Distributed DoS (DDoS) attacks tilt the balance of power further towards the attacker. Few end systems can survive an onslaught of tens of thousands of bots flooding them with unwanted traffic, and such large-scale attacks have been observed in the wild in the last couple of years. There is currently no way for a victim to reach back into the network and request that traffic from a particular source be switched off. The matter is made worse still by the ability to spoof the source address on traffic at many sites that do not perform ingress filtering [23].

One attack that has been seen in the wild recently is a DNS reflection attack. The attacker compromises a set of end systems at sites which permit source address spoofing. These bots then launch their attack by performing a DNS look-up for some domain name such as 'www.example.com'. The DNS look-up is sent to one of a large list of DNS servers that the attacker has probed and discovered that they perform recursive DNS look-ups for any client that asks. The source address on the DNS request is spoofed to be that of the victim. The DNS server then looks up 'www.example.com', and receives an answer — in this case it contains a DNS response that contains a DNSsec certificate, so the response is up to 4 kBytes in size. The response is now relayed back to the victim. For the price of a 60 byte DNS request, 4 kB is returned to the victim from one of thousands of DNS servers around the world that the attacker chooses as a relay. Consider if the attacker controls only 150 bots on DSL lines that permit 1 Mbit/s upstream. The reflection provides sufficient amplification that a resulting attack can saturate a 10 Gbit/s network link.

4.1.4 Application deployment

Firewalls, NATs and other middle-boxes do not fit into the layered IP architecture, as they work at layer 4 or above but are neither the originator nor the intended recipient of the packets they process. Not being a part of the architecture is problematic, as there is no prescribed way for an application to take these boxes into account. Add to this the general firewall issue — everything that is unknown is a *potential* threat — and the result is that it has become difficult to deploy new applications.

Many new applications today are deliberately designed to look like HTTP or are actually tunnelled over HTTP. The reason is obvious — HTTP is much easier to get through a firewall. Now this is not a problem to the extent that the natural way to implement a new application is to implement it in an HTTP-like way. And HTTP's model is pretty good for a wide range of applications ranging from file transfer to remote procedure call. Issues arise though where the new application does not resemble HTTP in the slightest.

Consider Skype [24], which is currently one of the best Internet telephony implementations in common use. Skype generally uses UDP for audio data, transmitting directly end-to-end from one Skype client to another. This provides the best service, keeping delay to a minimum, which is essential for effective telephony.

When it encounters a NAT, Skype is faced with a problem. What is the IP address and UDP port seen by the remote site? The NAT will have rewritten them. To establish an audio connection, Skype has to reverse engineer what the mapping is, and it does this using standard techniques [13, 14]. This may involve talking to a remote server to discover the address mapping, and using heuristics to discover the port mapping. Such techniques are complex, ugly and error-prone, but they are a fact of life at the moment. They do, however, mostly work.

Where both Skype clients are behind NATs, the problem is worse still. NATs tend to allow only outgoing connections, which means that the techniques above may not work. Skype is based around a peer-to-peer architecture, so when direct connectivity fails, Skype resorts to relaying the call through another Skype client which is not behind a NAT. Such relay nodes add delay, reduce reliability (the relay node can simply quit), and may become traffic concentration points, leading to network congestion. Clearly if most clients were behind a NAT, then these relay nodes would be overloaded, and the whole model would not work.

Finally, it is possible that Skype may completely fail to establish connectivity using UDP, most likely because the NAT was not well behaved or because of firewall rules. Skype will then fall back to relaying audio data over TCP to the nearest Skype supernode, which can then relay onwards using UDP. TCP is a very poor solution for this, as its reliable nature and congestion control can combine to provide large and variable end-to-end delays. Skype does not use TCP because it is a good solution — it uses TCP as a last resort.

Skype is very successful. It is successful because users say that it *just works*. But consider all the complexity of mechanism that was needed to make it just work. When it comes to Internet Telephony, sending the audio data over the Internet really should be a very easy problem. After all, it was first done in the 1970s. Although Skype does *just work*, in a very real sense, it *only just works*.

It is hard to predict the degree to which healthy innovation of Internet applications is currently being harmed by these issues. Where the motivation is strong enough, as with Skype, people usually find a way to work around problems. But this comes at the expense of great complexity, resulting in a net that is already much more fragile and harder to debug that it needs to be.

4.2 Medium-term problems

Some of the problems faced by the Internet are not currently impeding stability or performance, but the trends make it clear that they will do in the not too distant future. The list of potentially serious issues is rather long; I identify three that illustrate different aspects.

4.2.1 Congestion control

Congestion control is the only piece of core network functionality that was developed in response to a serious failure. TCP's congestion control mechanism is quite minimal, basically probing the network repeatedly to see how much traffic can be in flight at a time, and then backing off when overload is detected via packet loss. The basic mechanism works well, but since 1988 the requirements have changed. The net continues to get more heterogeneous, with 40 Gbit/s links currently deployed in the backbone, but with slow wireless and dial-up connections also being used. Although there are quite a number of issues, especially when it comes to wireless links, we will discuss just one here — TCP's limited dynamic range.

As network link speeds continue to increase, it is inevitable that faster links will not only be used by more flows, but also the *per-flow* bandwidth will also increase. Consider a well-provisioned path, where the intent is to transfer 1 Gbit/s from California to London. The round-trip time (RTT) is about 150 ms (in fact the best case for speed of light in glass is about 100 ms, but the networks do not always take the shortest geographic path). To transmit 1 Gbit/s over a 150 ms RTT requires an average transmission window of about 19 MBytes, or 12 500 packets of 1500 bytes each. With TCP's congestion control algorithm, the window increases by one packet each RTT until a loss is detected, whereupon the window is halved, and the slow increase begins again. To maintain an average of 12 500 packets means that the window must reach about 16 700 packets before a loss occurs, when it is then halved to 8350 packets. To increase from 8350 packets to 16 700 packets takes 8350 round trip times, or about 20 minutes, during which over 150 GBytes have been transferred. So, no more than one packet every 20 minutes must be lost for any reason.

From a transmission point of view, the uncorrected bit error rate needs to be no worse than 1 in 10^{12}, which is about the limit for most current equipment. From a TCP point of view, this means that no other TCP flow must slow-start in that time causing any queue to overflow. And from a fairness point of view, two flows will only converge to their fair share on a timescale of several hours. In short, this is pushing the bounds of TCP's algorithm. Now 1 Gbit/s is not exceptional these days — many research laboratories have networks available that can provide 1 Gbit/s wide area. A number of 'big science' experiments already need wide-area bit rates well in excess of 1 Gbit/s, and the commercial world is starting to use similar speeds for movie

distribution and similar applications. What these high-end users demand today will be commonplace in ten years, and TCP's current congestion control algorithm simply cannot cope.

Fortunately network researchers have been working on this problem for some time, and a number of alternative congestion control algorithms have been proposed. These range from relatively simple modifications to TCP's existing algorithm to give more dynamic range [25—27], to changes that involve modifications to many of the routers along a path [28, 29]. In general, schemes which involve the active participation of the routers are likely to perform better in a wider range of circumstances than schemes which only involve the end systems. However, they are also likely to be much harder to incrementally deploy. When it comes to congestion control, the problem is not a lack of solutions, but a lack of consensus on which solution to move towards.

Incremental deployment should be simple for the end-system-based solutions — simply use existing TCP dynamics at lower bit rates to be fair to legacy TCP implementations, and adaptively switch to the new behaviour at higher speeds. However, even here there is a problem — some of the schemes are already starting to be deployed. For example BIC TCP found its way into the Linux kernel, enabled by default, although there is no consensus that this is the right algorithm to deploy. Thus any new scheme must not only coexist with legacy TCPs, but also with competing newer TCPs.

4.2.2 Inter-domain routing

For over fifteen years BGP [30] has provided inter-domain routing for the Internet. BGP is conceptually very simple — routes to subnets are advertised together with attributes of the path to that subnet. These attributes include the path of routing domains (known as autonomous systems (ASs) that the route has taken through the network. Each time an AS passes on a route to another AS, it adds its own AS number to the AS path contained in the route. BGP's goal is to enable *policy routing*, whereby each routing domain can make its own choice about which routes to accept from its neighbours (based on the AS path and other attributes), and which routes to pass on to other neighbours.

Almost all the problems with BGP stem from the original design decisions that all ASs are by default equal, and that as policy is commercially sensitive, no policy information should be propagated to any other AS by the BGP protocol. These design decisions condemn BGP to explore many alternative possible paths while re-converging after a failure (as no AS knows which routes will be filtered by any other AS), and they condemn network operators to work in the dark as to the intended purpose of routes seen in the routing tables. BGP is therefore slow to converge, error-prone and easy to misconfigure, difficult to debug and insecure.

Somewhat ironically, the privacy of most BGP policy is fundamentally impossible to maintain in reality. A route needs to be propagated for packets to flow, and it is a simple task to infer many customer/provider relationships from the BGP path information in these routes. Thus BGP suffers from hiding information that cannot be a secret in the first place. This is not to say that all policies cannot be secret — just that the most common policies cannot be.

Are these issues significant enough to cause problems in the future Internet? I believe they will be, as much greater importance is placed on high availability. Slow routing convergence is a significant issue. In an ideal world routing protocols would reconverge fast enough that only a momentary glitch in communications is experienced, rather than an outage of several minutes.

Requiring explicit configuration of which routes are not allowed is a sure way for operator error to be a significant source of network outages. And the complete failure to deploy security enhancements [31, 32] to BGP is a serious cause for concern.

BGP is probably the most critical piece of Internet infrastructure — it holds the entire network together. The problem with BGP is that it mostly works. If it actually failed, then it would need to be replaced. Network operators actually think in terms of BGP — it is very hard for them to consider alternative ways of doing routing. If BGP is ever replaced, it seems at this point like it will not happen until BGP has been shown to fail. Such a moment would not be the ideal time to try and design alternatives.

4.2.3 Mobility

Support for mobile hosts has long been seen to be a future requirement, resulting in the development of Mobile IP [33]. To date, Mobile IP has been a near-complete failure when it comes to deployment. Why should this be when there is obvious agreement that many and perhaps most future Internet hosts will be mobile?

The first reason is that there are no devices that actually need Mobile IP. Mobile devices so far can be broadly classified as either mobile phones or laptops. Mobility for the former is largely solved at layer 2, so Mobile IP does not help these phone-like devices. Mobility for laptops is largely solved by DHCP. Although some technical users would like their *ssh* connections to

survive moving networks, there is no current compelling similar application for normal Internet users. DHCP serves these users well, with applications such as e-mail and instant messaging simply reconnecting each time the IP address changes.

It is possible that the emergence of phone-like devices that support both 3G and IEEE802.11 radios will change this story. However, there is still a question as to whether any applications actually care. I believe that VoIP may be the one application that needs Mobile IP, but it is far from clear that mobile telephony operators will provide the network support needed for Mobile IP to work, especially as this is in direct competition with their regular telephone service.

The second reason for a lack of deployment is the usual chicken-and-egg problem. Why deploy Mobile IP foreign agents on your network to support visiting hosts when those hosts do not support Mobile IP yet? And why integrate Mobile IP into the host stack, when there is no network support yet?

The likely story is that mobility will in fact be solved at the last minute at layer 4 or above, rather than layer 3. Mobility extensions to transport protocols are comparatively easy to deploy, requiring no support from foreign networks, and so do not suffer from the chicken-and-egg problem to the same extent. But the rush to add mobility support will likely come at the last minute, when applications that really need mobility finally emerge.

4.2.4 Multi-homing

Multi-homing is another issue that is growing in importance, but is not yet critical. As businesses come to depend more and more on Internet access, the consequences of a network outage become much more severe. It is natural then that any company that depends on the Internet for revenue will attempt to provide redundancy by connecting via more than one Internet provider. This is not technically difficult, and can be achieved by simply announcing the same address prefix via both providers. However, when this is done by a great many edge networks, the effect is to prevent address aggregation in routing announcements. The result is that the number of address prefixes that needs to be routed tends towards the number of end networks in the world, rather than towards the number of large ISPs. It is very unclear, at this moment in time, whether backbone routers or indeed the BGP routing protocol itself will cope with this massive increase in the number of advertised routes.

This growth due to multi-homing is already happening. Short of an architectural solution [34] to the problem that can utilise more than one address prefix for each edge-subnet, the growth seems certain to continue unabated. The question that remains is whether this will lead to an increase in routing problems, or whether more expensive routing hardware can keep ahead of the growth.

4.2.5 Architectural ossification

Perhaps the biggest problem is related to why all the other problems have not yet been addressed. The Internet architecture has not evolved significantly since the early 1990s, despite a revolution in the use of the Internet. NATs and firewalls have made it harder to deploy new transport protocols and even new applications. Deep packet inspection threatens to make this problem worse in future.

Even IP itself is affected. IP was supposed to be extensible through the use of IP options, but long ago the introduction of a hardware-assisted *fast path* through router forwarding engines meant that packets without IP options were forwarded much faster than packets with IP options. Today, using a new IP option would amount to a denial-of-service attack on the routing processor of many fast routers, and so such packets are highly likely to be filtered. Thus IPv4 effectively lost the use of its extension mechanism. IPv6 attempts to rectify this by separating end-to-end IP options from hop-by-hop IP options but deployment to date has been glacially slow.

In short, the Internet architecture has *ossified*. This would perhaps not be a problem in itself if it were not for the fact that the expectations and requirements placed on the Internet have not remained constant. There are significant problems that have been building for some time, but changing the Internet to address these problems has never been harder.

4.3 Longer-term problems

There will no doubt be many problems that need solving in the long term. However, for the most part it is hard to see what these problems will actually be. There is one exception to this — address space depletion.

4.3.1 Address space depletion

In the early 1990s it became clear that the Internet would run out of IP addresses. CIDR [35] was an interim solution to this, and has been quite successful. The rise of NATs, from being considered an ugly hack to being nearly ubiquitous, has also reduced the problem somewhat, although it has not actually been a shortage of addresses that has encouraged NAT proliferation. The long-term solution to the problem of address space depletion is supposed to be IPv6. Despite a great deal of effort, IPv6 deployment remains rare today. Quite simply, there is no significant incentive to deploy IPv6.

This may change, as Internet-connected mobile devices become the norm, but this in turn may simply serve to create walled gardens for mobile operators unless the rest of the Internet also deploys IPv6. The jury is still out on whether IPv6 will replace IPv4 in the long run. What is certain is that a shortage of IPv4 addresses is unlikely to force the issue for many years [36].

I hope that IPv6 will succeed, but I am concerned that it may not. If there is not significant application support by the time addresses do start to become short, then it will likely be too late — NATs will become required, and we are likely to see connections that traverse multiple NATs. This would become a deployment nightmare for new applications.

5. Perspective

The Internet was never designed to be optimal for any particular problem — its great strength is that it is a general-purpose network that can support a wide range of applications and a wide range of link technologies. The Internet is also a cost-effective network — it does not make great promises about the quality of service that it provides. It is *good enough* for a wide range of applications, but anyone considering telesurgery or remote-control of a nuclear power station might well be advised to look somewhere else. It basically provides 80% of the capability for 20% of the cost. If we wanted 100% of the functionality, so that telesurgery routinely could be performed over the Internet with very low risk, then it is highly likely that the network would be too expensive for the vast majority of users who wish to exchange e-mail, chat, or surf the Web.

However, the requirements are changing. Convergence progresses apace, in spite of the problems, and these changed requirements bring with them risks. The Internet is going to suffer growing pains as it progresses from providing 80% of the functionality to providing 90+% of the functionality, as called for by the new requirements. The track record is not at all good — the history of major changes that have been successful is one of changes implemented at the last minute. This should not be a surprise — there are always too many immediate issues to be concerned with to invest time and money on those that are not currently critical. And consensus for architectural change is very hard to reach unless faced with a specific and pressing problem.

To conclude, in many ways the Internet *only just works*. The number of ways in which it only just works seems to be increasing with time, as non-critical problems build. The main question is whether it will take failures to cause these problems to be addressed, or whether they can start to be addressed before they need to be fixed in an ill co-ordinated last-minute rush.

References

1 Crocker S: 'Protocol notes', RFC 36, Network Working Group, (Updated by RFC44, RFC 39) (March 1970).

2 Clark D D: 'The design philosophy of the DARPA internet protocols', in Proc ACM SIGCOMM, pp 106—114, Stanford, CA, (August 1988).

3 Kudlick M: 'Host names on-line', RFC 608, IETF (January 1974).

4 McQuillan J, Richer I and Rosen E: 'The New Routing Algorithm for the ARPAnet', IEEE Transactions on Communications (May 1980).

5 Rosen E: 'Exterior Gateway Protocol (EGP)', RFC 827 (October 1982).

6 Jacobson V: 'Congestion avoidance and control', in Proc ACM SIGCOMM, pp 314—329, Stanford, CA (1988).

7 Lougheed K and Rekhter Y: 'Border Gateway Protocol BGP', RFC 1105 (June 1989).

8 Rosen E, Viswanathan A and Callon R: 'Multiprotocol Label Switching Architecture', RFC3031, IETF (January 2001).

9 Ramakrishnan K K, Floyd S and Black D: 'The addition of Explicit Congestion Notification (ECN) to IP', RFC 3168, IETF (September 2001).

10 Braden R, Clark D and Shenker S: 'Integrated Services in the Internet architecture: an overview', RFC 1633, IETF (June 1994).

11 Carlson M, Weiss W, Blake S, Wang Z, Black D and Davies E: 'An architecture for differentiated services', RFC 2475, IETF (December 1998).

12 Rosen E and Rekhter Y: 'BGP/MPLS VPNs', RFC 2547, IETF (March 1999).

13 Rosenberg J, Weinberger J, Huitema C and Mahy R: 'STUN — simple traversal of user datagram protocol (UDP) through network address translators (NATs)', RFC 3489, IETF (March 2003).

14 Rosenberg J, Mahy R and Huitema C: 'Obtaining relay addresses from Simple Traversal of UDP Through NAT (STUN)', Internet-Draft (February 2006).

15 Kohler E, Handley M and Floyd S: 'Datagram Congestion Control Protocol (DCCP)', RFC 4340, IETF (March 2006).

16 Kohler E, Handley M and Floyd S: 'Designing DCCP: Congestion Control without Reliability', to appear in Proc ACM SIGCOMM (September 2006).

17 Stewart R, Xie Q, Morneault K, Sharp C, Schwarzbauer H, Taylor T, Rytina I, Kalla M, Zhang L and Paxson V: 'Stream Control Transmission Protocol', RFC 2960, IETF (October 2000).

18 Smirnov A and Chiueh T: 'Dira: Automatic detection, identification and repair of control-hijacking attacks', in NDSS (2005).

19 Kiriansky V, Bruening D and Amarasinghe S P: 'Secure execution via program shepherding', in USENIX Security Symposium (2002).

20 Dunlap G W, King S T, Cinar S, Basrai M A and Chen P M: 'Revirt: Enabling intrusion analysis through virtual-machine logging and replay', SIGOPS Oper Syst Rev, 36, (2002).

21 Newsome J and Song D X: 'Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software', in NDSS (2005).

22 Raiciu C, Handley M and Rosenblum D: 'Exploit Hijacking: Side Effects of Smart Defenses', Proc ACM SIGCOMM Workshop on Large Scale Attack Defence (September 2006).

23 Ferguson P and Senie D: 'Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing', RFC 2267 (January 1998).

24 Skype — http://www.skype.com/

25 Floyd S: 'HighSpeed TCP for Large Congestion Windows', RFC 3649, IETF (December 2003).

26 Shorten R and Leith D: 'H-TCP: TCP for high-speed and long-distance networks', Proc PFLDnet, Argonne (2004).

27 Xu L, Harfoush K and Rhee I: 'Binary increase congestion control for fast long-distance networks', Proc INFOCOM (2004).

28 Demers A, Keshav S and Shenker S: 'Analysis and simulation of a fair queueing algorithm', Internetworking: Research and Experience, 1, pp 3—26 (1990).

29 Katabi D, Handley M and Rohrs C: 'Internet Congestion Control for High Bandwidth-Delay Product Environments', Proc ACM SIGCOMM (2002).

30 Cerf V: 'Report of the Second Ad Hoc Network Management Review Group', RFC 1109 (August 1989).

31 Kent S, Lynn C and Seo K: 'Secure Border Gateway Protocol (S-BGP)', IEEE JSAC Special Issue on Network Security (April 2000).

32 White R: 'Architecture and deployment considerations for Secure Origin BGP (soBGP)', Internet-Draft (June 2006).

33 Perkins C: 'IP Mobility Support', RFC 2002, IETF (October 1996).

34 Huston G: 'IETF64 Review: SHIM6', IETF Journal, 1, Issue 2 (Winter 2005/2006).

35 Rekhter Y and Li T: 'An architecture for IP address allocation with CIDR', RFC 1518, IETF (September 1993).

36 Huston G: 'IPv4: How long do we have?', The Internet Protocol Journal, 6, No 4 (December 2003).

Mark Handley joined the Computer Science department at UCL as Professor of Networked Systems in 2003, and holds a Royal Society-Wolfson Research Merit Award.

He leads the Networks Research Group, which has a long history dating back to 1973 when UCL became the first site outside the United States to join the ARPAnet, which was the precursor to today's Internet.

Prior to joining UCL, he was based at the International Computer Science Institute in Berkeley, California, where he co-founded the AT&T Center for Internet Research at ICSI (ACIRI). He has been very active in the area of Internet Standards, and has served on the Internet Architecture Board, which oversees much of the Internet standardisation process. He is the author of 22 Internet standards documents (RFCs), including the Session Initiation Protocol (SIP), which is the principal way telephony signalling will be performed in future Internet-based telephone networks.

His research interests include the Internet architecture (how the components fit together to produce a coherent whole), congestion control (how to match the load offered to a network to the changing available capacity of the network), Internet routing (how to satisfy competing network providers' requirements, while ensuring that traffic takes a good path through the network), and defending networks against denial-of-service attacks. He also founded the XORP project to build a complete open-source Internet routing software stack.