Weaknesses of today's Internet architecture

Felix Schindler
Betreuer: Nils Kammenhuber
Seminar Innovative Internettechnologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: schindlf@in.tum.de

ABSTRACT
This paper describes the main weaknesses of today's Internet architecture, and the problems that arise from these weaknesses. In addition to that it gives a short overview over the assumptions on which the original architectural decisions were based, and why they do not hold anymore. There are some serious design flaws that really keep the Internet from showing its full potential when it comes to performance, and also in terms of usability. These weaknesses pose a big threat to the future of the Internet. Although there is a lot of research on future Internet technologies going on at the moment, still a lot of effort will be needed to overcome these barriers.

Keywords
Internet, Architecture

1. INTRODUCTION
The history of the Internet is a story of success and growth, and at first sight it seems to make a good job. But if you take a closer look behind the scenes, you start to realize that this is not the case. This paper gives an overview over the design decisions on which today's Internet is based. It also tries to give some insight on why the architecture is struggling to meet today's demands and the demands of the future. All this is becoming more and more important, as it has a great influence on how the Internet will look like in the future.
The first part of this paper focuses on problems that are based on the simple core functionality of the Internet. In the second part we will show why the unexpected growth in itself is causing more and more trouble. After that there will be a section about the most common security issues and why their importance is growing. Later we show why the change from a relatively static to a completely dynamic network is causing ISPs some serious headache. Last, we will look at problems that are based on combinations of the points described before.

2. SMART EGDE, DUMB CORE
For a better understanding of this paper it should be clear that when we talk about the Internet, the core Internet or the core network in almost all cases we talk about IP and the network layer and not about transport layer protocols.
To get a better understanding of the architecture of today's Internet and the resulting problems, a few points about the basic design decisions [1] have to be made clear first. The core architecture of the Internet was planned and implemented to be simple and robust. It offers a best effort service trying to forward small packages of data (IP packets [3]) to a certain destination. Apart from that, next to no additional services are offered. If you want to send data to another host in the Internet the data is split into small packets, containing a source address and a destination address. Then it is sent into the network. Within the network the nodes (routers) that receive those packets try to determine the best route to the destination of the packets, and just forward them into this direction.
What is really characteristic about the Internet is that the source of a packet can never be sure that the packet it sent actually arrived at the destination. If the packet got lost on the way for whatever reason, the sender is not informed in most cases. In short terms, the Internet is a best effort packet switching network [3].
In general it was tried to keep the components that make up the core Internet (see Figure 1) as simple as possible. The result was a stateless network. Routers do not store or use any state when they forward packets through the network, except for the routing information. One reason for that certainly was the hardware performance during the first years of the Internet. Keeping state for every connection would have overwhelmed the capabilities of routers, or at least made them much more expensive. In addition to that, the stateless approach has a considerable advantage. If routers kept state and then failed for whatever reason, all state were lost and had to be reestablished after recovery. This would result in dropped connections and lost data, which really is not desirable.

As we have seen, the services offered by the core Internet are very limited on purpose. This lead to the fact that whenever additional functionality apart from simple packet forwarding was needed, it was implemented not at the network layer but at higher protocol layers. This course of action has been practiced up until today. It has become common sense that new functions should be introduced at higher levels in the network protocol stack, if they do not add considerable advantages when implemented at lower layers. Therefore, the core network architecture and protocols hardly changed over the years after their introduction [2]. Another reason for the reluctance to to introduce new functions is the size of the Internet. It is really difficult to change big networks, as it would require to apply changes to a huge number of systems all over the world. Especially for network equipment like routers, today this is nearly impossible without threatening

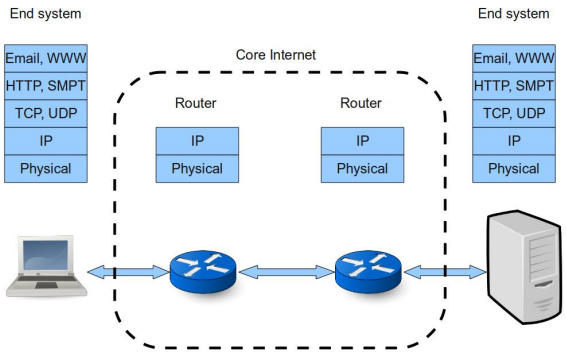

Figure 1: Protocols used in the Internet

the functionality of big parts of the Internet.

2.1 Lack of Quality of Service

The number of applications and protocols used in the Internet is growing, and with that the number of applications that rely on a certain performance provided by the underlying network connection. These applications and protocols require a certain Quality of Service (QoS) [16] in order to work properly. Applications like IP telephony or online gaming need considerable bandwidth or low latency in order to provide a good user experience. Therefore it would be desirable to notify the network about these demands, to reserve resources along the path of communication within the network to meet these demands. This also includes the separation of traffic into different classes which then can be given different priorities. For instance it would be preferable to give Voice over IP (VoIP) traffic a higher priority contrary to a simple file download. For the duration of a VoIP call, the network should have the ability to throttle the speed of the file download, so as to be able to offer enough resources to keep up the voice quality and the low latency.

The current architecture and the current protocols do not offer this kind of functionality, because it contradicts the principle of keeping state out of routers. For every connection or data flow, every router on the communication path would have to store some information about how to handle the incoming data. Not only is this an additional burden for the router in terms of memory consumption and processing time; it also introduces new problems that arise when a route suddenly changes. This may be the case when new routes are introduced or old ones are taken down. Another question is how the reservation of resources is propagated through the network along the communication path.

Of course there are protocols which offer resource reservation in the network [4], and even IP offers a header field for QoS, but hardly anybody uses them. Some providers use QoS mechanisms within their networks to guarantee certain services to single customers, but QoS across the borders of ISPs is just not available.

Instead of using QoS, IPSs try to provide the needed resources for certain protocols by using fast and expensive network equipment. This approach is called over provisioning [11]. Providers need to come to an agreement including a standard protocol, a proper billing mechanism and a way to guarantee QoS across the borders of single ISPs. Otherwise, over provisioning will stay the method of choice, despite being costly.

2.2 No congestion control in the network

One thing which is really remarkable is that the core Internet usually does not offer congestion control. Of course there are mechanisms which implement congestion control at the network layer, but they are not compatible to each other. The result is that they are used only locally and in most cases they are not used at all. The most widespread form of congestion control is implemented at the transport layer, in end systems in the Transmission Control Protocol (TCP) [6] [9] and similar transport layer protocols [7] [8]. And even there it was introduced later in the mid 1980s after the Internet collapsed a few times due to congestion [2]. It was a quick fix intended to face an imminent problem, and was thought to be a temporary answer to the congestion control problem [2]. An indicator for this is the fact that other transport layer protocols which are commonly used do not offer congestion control at all. This is a cause for concern, as more and more data intensive applications like audio streaming or video streaming refrain from using TCP. They prefer the User Datagram Protocol (UDP) [13] as transport protocol because of its high throughput, low overhead, and low latency. This leads to two problems. First the applications using UDP as transport protocol violate the principle of fairness. Fairness in this case means that all applications should obtain an equal share of the available bandwidth. TCP provides this kind of fairness while UDP does not [9] [12]. The second problem that may arise is even more severe. Due to the fact that there is no congestion control built into UDP, it is possible to overwhelm the network with traffic and therefore causing it to collapse [9].

Congestion control in end system at the transport layer might not be the best solution, because end systems usually have only little information about what is going on in the network [15]. The core of the network, which has all the information needed for proper congestion control, should be able to make much more sensible decisions when it comes to congestion control. This might also lead to a more effective usage of the bandwidth of links connecting parts of the network.

Instead of implementing congestion control at the network layer, the main strategy of ISPs to overcome this problem is again to use over provisioning. As mentioned before, this might not be the best solution although it worked reasonably well in the past. But in the long run it might prove as insufficient [11].

2.3 No traffic filtering in the network

In today's Internet there is no possibility to unsubscribe from unwanted traffic. All traffic you do not want to receive could be filtered within the network, before it reached your end systems. As we will see this approach has major advantages.

Unfortunately all traffic sent to you will reach you in the end. If you do not want it you have to filter it yourself. This is a serious drawback. It means that unwanted traffic has to cross the Internet before it can be discarded at the destination. This traffic thereby uses a lot of valuable resources like bandwidth and processing time in routers, just to be thrown

away in the end. Spam, for example, causes a considerable amount of today's Internet traffic, and it would be a big advancement to be able to filter this kind of traffic in the network as soon as possible. This course of action makes even more sense if you consider DoS attacks [14]. These attacks try to exhaust the victim's network and server resources by overwhelming them with huge amounts of traffic. This attack is really hard to protect against, unless you are able to intercept the traffic in the network, before it reaches the victim.
Being able to unsubscribe from unwanted traffic, and being able to block unwanted traffic in the network might even be sensible from a financial point of view. It would drastically reduce the amount of money which is spent by ISPs and companies to protect from DoS-attacks, spam and other unwanted traffic that poses a threat to their systems. It also might add to the security as there are several independent filtering points throughout the whole Internet and not just one at the end system of the receiver which poses at potential single point of failure.

2.4 No Multicast

Another technology that is still lacks widespread deployment in the Internet is multicast[17]. In contrast to unicast addressing, more than one host can be reached by sending to a single address. Host which are reachable with the same multicast address form a multicast group. When sending data to a multicast address, it is distributed to all members of this multicast group. This scenario offers several advantages over the common unicast approach. A server that distributes content does not have to be aware of all the recipients that actually want the data. The server just sends the content once to the multicast address, and the network takes care of distributing the data to all members of the multicast group. This introduces some big improvements for the server, because it does not have to handle connections to all clients that want content. In the best case scenario, the server has to send the data only once, and is still able to reach thousands of clients. The multicast approach also reduces bandwidth consumption within the network, as the data that is sent is only duplicated at routers that have more than one multicast group member attached to them (in network duplication, see figure 3). When using unicast data has to be sent n times for n clients (source duplication, see figure 2).

A good example for an application which could make use of this technology extensively is Internet Television. With thousands of people watching the same program at the same time, multicast would drastically reduce the load on the servers and the network. Another scenario which can be considered, would improve the propagation of updates. Instead of thousands of clients that periodically contact the server to ask for new data, they join a multicast group. Once there is an update at the server, it is distributed to the multicast address.
When looking at these advantages one might wonder why the use of multicast is not common in the Internet. Let us take a look at reasons. First there are a lot of different multicast routing protocols which complicate the task of building a distributed multicast architecture. In addition to that, some multicast routing protocols are limited to intra-AS use. Therefore it is hard, or even impossible to build

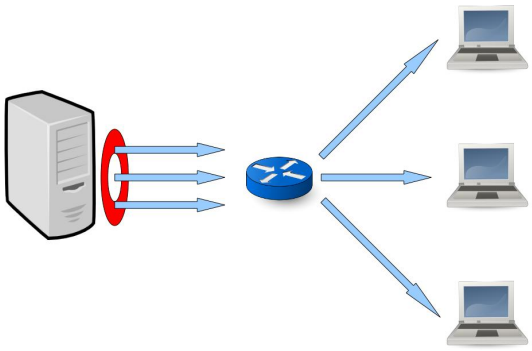

Figure 2: Source duplication

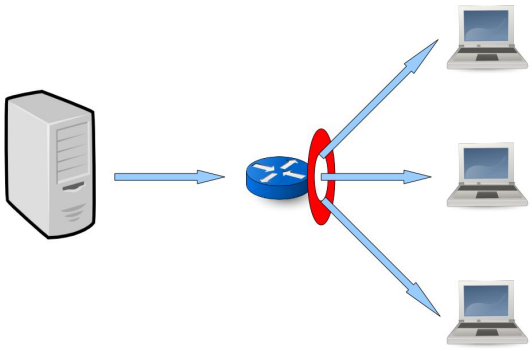

Figure 3: In network duplication

multicast structures which overcome the borders of single ASes [18].
Another reason for the slow deployment of muticast is that, at least at the moment, the management overhead is still too big. This means that it is more cost effective to stick to unicast, despite its higher bandwidth consumption, than to effectively manage multicast routing [18].
Similar to QoS, the lack of a working billing infrastructure keeps multicast from being deployed on a large scale.

3. INTERNET SIZE

Let us now take a look at the size and the growth of the Internet, which is becoming more and more of a problem. During the development phase it was considered to be a network of intermediate size, connecting a few hundred, maybe few thousand military, scientific and educational institutes all over the world. The most important decision that is based on this consideration is the size of the IP address space. IP addresses consists of 32 bits which leads to roughly 4 billion possible host connected to the Internet. As will we show later on, IPv6 (see section **??**) offers bigger addresses. For quite a while this seemed to be more than enough. But then the Internet started to grow exponentially (see figure 4), and

people realized that we would eventually run out of IP addresses.

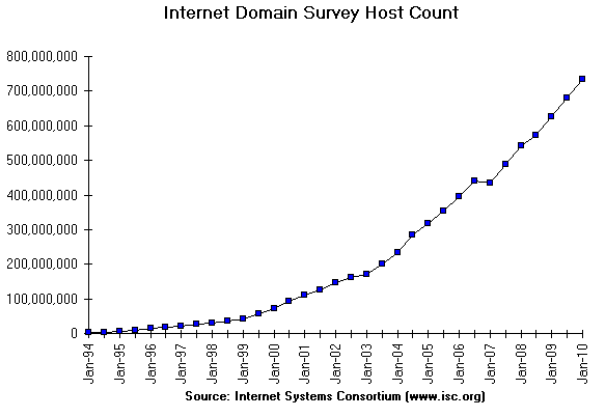

Figure 4: Exponential growth of the Internet [34]

Calculations have shown that the last unused IP addresses will be given to Regional Internet Registries (RIRs) in 2011, and that the RIRs themselves will give away their last unused addresses in 2012. Although there are possibilities to cope with that problem at least for a certain period of time, this is one of the biggest and most imminent threats to the Internet.

3.1 Network Address Translation

The first thing which has to be pointed at when talking about Network Address Translation (NAT)[20] is that it was introduced to provide a temporary solution for the address space depletion. But as time went by, NAT became more and more common and started to become a problem itself.
NAT tries to cope with the depletion of IP addresses by hiding whole networks behind one public address. This is done by mapping the local IP address and the port to the public IP address and a public port. At first sight this seems like a sensible solution, but if you take a closer look, NAT introduces a whole new set of problems.
Although devices behind a NAT are able to connect to hosts in the Internet, connections in the opposite direction usually fail. This breaks nearly all applications that rely on end-to-end connectivity. Famous examples are FTP and peer-to-peer applications. Apart from breaking end-to-end connectivity, it also introduces problems for programs that send their own address to other hosts. When someone tries to to establish a connection using SIP [21], this will not work because his own address which he sent is from the private network, and not reachable from the Internet.
Another big problem when it comes to Network Address Translation is that there is no common standard of how a NAT device should be implemented. This is really painful for developers of network applications, as they have to provide solutions for all different kinds of NATs. This has lead to several different frameworks [32] that try to help applications to determine the kind of NAT which is used, and offer functionality to overcome the NAT traversal problem.
Apart from not being standardized, NAT usually blocks all transport layer protocols that that are neither TCP nor UDP and therefore prevents new transport layer protocols [7] [8] from being propagated.

The biggest problem introduced by NAT is its extensive and widespread use. NAT has become so common that some people claim that the Internet has changed from a network connecting networks to network connecting NATworks.

3.2 IPv6 deployment

IPv6 [5] is an advanced version of the current network layer protocol IPv4. It was developed to take care of the problems of IPv4. First IPv6 offers 128 bit addresses that allow about 3.4×10^{38} hosts, which addresses the problem of IP-address depletion. It uses fixed length headers which reduces processing efforts within routers. Another feature that was introduced to reduce the processing load on routers is that there is no possibility to fragment IP packets within the network. IPv6 also introduces new features for a better support of QoS and multicast. Even network layer security is provided by IPv6.
As you can see IPv6 offers a lot of improvements over IPv4, and it officially is the successor of IPv4. The problem is that it has been like this for over 10 years. Since its definition in 1998, people thought that the switch to IPv6 would be just around the corner. Nowadays it is supported by protocols needed to run the Internet (DNS, routing protocols, etc.) and by all operating systems running on end systems. But still IPv6 traffic accounts for a very small percentage of the overall Internet traffic (see figure 5). If you take a look at the top one million websites of the world, only 0.15 percent offer an IPv6 website [22].

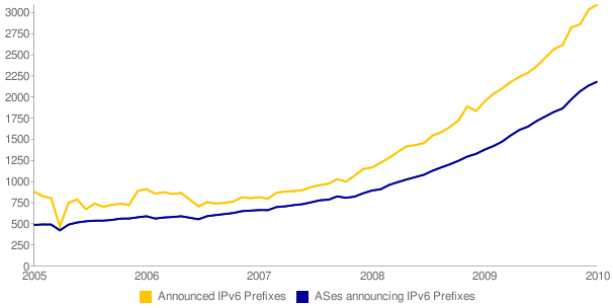

Figure 5: IPv6 Deployment in the Internet [26]

Why are people so reluctant to switch to something that is obviously better than the current system? There are several reasons for that. First is the question of how the transition should be made. There are two conflicting scenarios. The first one introduces a flag day, on which the whole Internet replaces IPv4 by IPv6. By looking at all the complex systems involved making the Internet work, you will realize that this approach just would not work. It would take days if not weeks to get the basic functions operating and even more important, interacting again. This would be a global disaster as whole economies are based on and rely on the Internet. Another strategy is a slow transition from IPv4 to IPv6 with both protocols coexisting for a certain period of time. This includes tunneling IPv6 traffic over IPv4 networks, and a so called dual stack [23]. This dual stack provides IPv4 and IPv6 capability to the TCP/IP protocol stack at the same time.
Another problem is that at the moment there is no financial gain in switching to IPv6. Actually, quite the opposite is true. The switch requires a lot of work and financial effort.

This poses a chicken and egg problem. In order to use IPv6 at end systems it has to be provided by content providers first. Content providers cannot switch to IPv6 if ISPs do not offer it. And the ISPs do not offer it because end users do not ask for it, because there is content distributed over IPv6.
As you can see, the switch to IPv6 is taking place slowly, and both protocol versions will coexist for quite a while and it is still not clear when IPv6 will be deployed on a large scale. But in the long run it will replace IPv4 as the problems it addresses are becoming more and more serious in the future.
The switch to IPv6 is a good example of the attitude towards fundamental changes in the architecture of the Internet: "Never touch a running system." At least if it is not absolutely necessary. This shows how inflexible the Internet has become over time in spite of its modular architecture. This inflexibility is sometimes called ossification [2]. It also shows how dependent the world has become on a working Internet.

4. DYNAMIC NETWORK

As we have seen before, the Internet has grown exponentially in the past and will keep on doing so. With the size and the widespread use a new problem arises. The Internet is changing constantly. New networks join, old networks leave or move somewhere else. New connections are introduced, old ones are discarded. This fact is putting more and more pressure on the design which was based on the assumption that the Internet would be a relatively static network.

4.1 Slow routing convergence

The Internet consists of more or less independent networks that are interconnected. These so called Autonomous Systems (ASes) share routing information using the Border Gateway Protocol (BGP) [24] in order to provide a globally consistent information basis on which routing decisions can be made. Apart from the routing tables which have become huge (> 300.000 entries, see figure 6) [25], BGP has difficulties coping with the constant change happening in the Internet. The problem is that it takes time to distribute updated routing information throughout the Internet to reach a new consistent state. Not only is the time the updates take a problem in itself, but during that time the global routing information is inconsistent. If some parts of the network make routing decisions based on new information, and some parts on old information, they might make incompatible decisions. This can lead to data being routed to dead ends, where it is dropped and therefore lost. Established connections break, and even whole parts of the Internet might be unreachable until a consistent state is reestablished. Due to the fact that the routing information basis is growing constantly, and therefore changing constantly this is a serious problem, which is hard to come by. Although these glitches are usually only temporary, they have to be approached nevertheless, as reliability is asked for by normal end users, and needed by most companies. In addition to the annoyance these glitches cause there is also the risk of substantial financial loss. If safety critical systems are involved failures due to slow routing convergence might even endanger people.

4.2 No locator/ID split

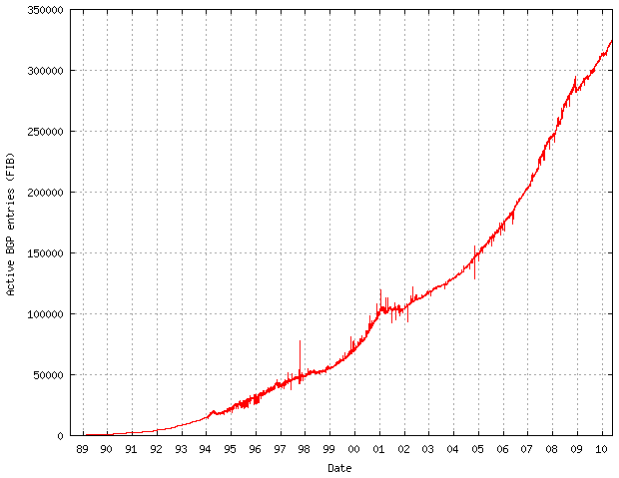

Figure 6: Growth of BGP routing tables [25]

If we look at an IP address we realize that it fulfills two functions. First it points to a certain location within the network which is needed for proper routing. Second it provides a unique identifier for a device. When you think of stationary devices this does not seem like a problem. And in most cases it is not. But change also affects the location of devices. With the spreading of mobile devices like laptops, handhelds and even mobile phones, end systems tend to move to different locations in the network on a regular basis. This means that whenever they change their location they will get assigned a new IP address, and the result is that they cannot be reached anymore using the old address. So whenever a device moves somewhere else, it has to inform all communication partners about its new IP address. Otherwise they will not be able to contact that device any more. In addition to that, established TCP connection are dropped when the IP-address of one of the communication partners changes.
Apart from that, even a lot of stationary devices change their IP address on a regular basis. They either get IP addresses assigned dynamically by their home network (e.g. DHCP), or their access providers give them a new address whenever they connect to the Internet (most dial up connections work that way).
With the rise of mobile systems, a separation of the location and the device identifier would be preferable. This is called the locator/ID split. It basically decouples the location within the network from its globally unique identifier. When the device moves only the locator changes, but it still can be reached using its unique identifier.
This approach offers advantages for different scenarios [27]. First it becomes easier to migrate whole networks from one location to another, without having to renumber all hosts within this network. The second point is that it is less complicated for networks to be attached to the Internet at two different points. Company networks are often connected via two different ISPs to have redundancy in case of one ISP having problems. Third, the locator/ID split can reduce the size of routing tables, because the network locators could be aggregated much more efficient than today's IP addresses [28]. Systems that address this problem are the Domain Name System (DNS) which basically provides a service to resolve

names into IP addresses. The problem with DNS is that changes are propagated too slowly to provide a reliable solution for the locator/ID split. Another discussed solution divides IP addresses into a dynamic locator and a static identifier. The question is whether to include this functionality into the core network, or to build another system on top to provide it.

5. LACK OF SECURITY

During the first years of its existence, the Internet was a network connecting mostly academic, military and scientific institutions. People who used it were scientists and technicians. It was a situation where people using the Internet could be trusted, because at that time nobody would try to harm other users or the infrastructure itself [29]. First, everybody was knowledgeable enough not to break something by accident. Second, the Internet was an environment where people cooperated to achieve certain goals. Most important is the fact that at that time, no financial profit could be made by attacking someone or something.
This situation of trust lead to the fact that until today the core of the Internet is lacking basic security features. Security was thought to be an additional "nice to have" specialty. Whenever security was needed, it had to be implemented at the edge of the Internet, in higher layers of the protocol stack.

5.1 Lack of encryption

When you look at the most common applications and protocols used today, like email, WWW or even IP itself, you realize that they offer no encryption whatsoever. For a lot of applications this does not matter, because they actually do not require encrypted traffic. When surfing the Web, in most cases it just does not matter if others are able to know what you are looking at. But more and more people use websites where a login is required to access a certain kind of service. This is problematic, as unencrypted traffic can be read by potential attackers, who are able to gain knowledge of passwords and other login data. When you look at email this is even more severe, because emails often contain personal information or even passwords which are not intended for others.
Another problem when it comes to unencrypted traffic in the Internet is that most users are unaware of the risks or simply ignore them. Sentences like "I have nothing to hide", or "Who would want to attack me¿' are quite common. Although this state of mind is receding more and more, security features are often left unused by end users. The reason for this is that using security functions requires additional knowledge and effort. They often interfere with simplicity and the ease of use of applications.
Of course there are solutions that provide encryption for Internet traffic like SSL/TLS [30] which offers transport layer security or IPSec [31] which actually works on the network layer. But they all have to be used explicitly. As IP is the protocol all traffic uses to traverse the Internet, it is the place where encryption should be introduced and used by default. The network layer is the layer where a common security policy can be enforced, transparent to all protocols and application which are used in the Internet. With the growing need for security, the time has come to introduce encryption as a core feature and not just as a "nice to have" addition. As said before, IPv6 offers network layer security but still lacks a global deployment.

5.2 Lack of cryptographically signed protocols

Similar to the lack of encryption is the lack of digitally signed protocols. They provide two things which encryption alone cannot do. Data integrity is the first one. It protects data from being altered on the way from sender to the receiver without being detected. The second is authentication which identifies the communication partner on a reliable basis. Combined with encryption, these two features make the most important building block of secure communication over the Internet.
Especially when it comes to financial transactions made over the Internet, encryption alone is not enough. In fact, encryption is often pointless if you cannot tell whether your communication partner is who he claims to be. The same goes for data integrity.
Similar to encryption there are protocols that provide these features. And like encryption authentication and data integrity have become important enough to be added as a core functionality to the Internet.

6. LOWEST COMMON DENOMINATOR = HTTP

As we have seen before, the Internet has become a more and more hostile environment in which nobody can be trusted. This lead to the widespread use of firewalls which try to block unwanted and malicious traffic either entering or leaving a network. As described before, NATs also introduce a certain kind of traffic filtering entities. For applications that do not belong to the group of standard Internet applications (Email, WWW, etc.) it is becoming more and more difficult not to get filtered by these devices. Whatever firewall or NAT is used, HTTP traffic is the most likely not to get filtered. This has lead to more and more applications using HTTP to exchange data over the Internet. At first sight, this may not seem too problematic, as HTTP is suitable for a broad range of different applications.

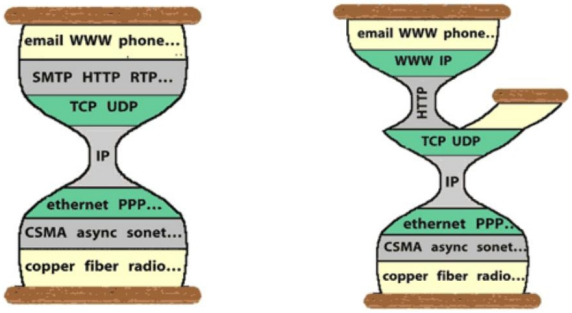

Figure 7: The change of the hourglass model of the Internet [33]

However, this makes it more and more difficult to distinguish between real HTTP traffic and traffic from other applications which can compromise the security provided by firewalls. A good example here is Skype (a widely use VoIP application) as it waits for connections on port 80 and port

443 which are usually used for HTTP and HTTPS connections. In addition to that HTTP uses TCP as transport layer protocol which makes it unsuitable for traffic that requires a constant bit rate like audio or video streams. In figure 7 you can see how the hourglass model of the Internet has changed and HTTP has become sort of the "new" transport protocol.

7. CONCLUSION

Although the Internet seems to be working quite well, we have shown that it is facing problems. Especially the simple core design is starting to show its age, and probably has to be changed in the future, to be able to keep up with the ever growing demand for better performance and functionality. For almost all problems shown in this paper, there is at least one proposed solution. The main problem is that it becomes more and more difficult to introduce these new ideas on a global scale. The Internet has become too important to be used as a test setup where you can break things. Another point is that there is no common sense about which direction the development should take, as there are different approaches competing against each other. Therefore it is important to switch to IPv6 as soon as possible, as it provides a foundation which deals with at least some of the weaknesses described here.
Finally the time has come where the simple architecture of the core Internet despite its success is starting to show its age. It is necessary to break with the old paradigm of keeping the core simple and introduce new functionality to meet the growing demands for new services.

8. REFERENCES

[1] David D. Clark: *The design philosophy of the DARPA Internet protocols*, Symposium proceedings on Communications architectures and protocols, Stanford, California, United States,Pages: 106-114, 1988

[2] M. Handley: *Why the Internet only just works*, BT Technology Journal 24, 3, July 2006

[3] J. Postel: *Internet Protocol*, RFC 791, IETF, September 1981, `http://tools.ietf.org/html/rfc791`

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin: *Resource ReSerVation Protocol (RSVP)*, RFC 2205, IETF Network Working Group, September 1997 `http://tools.ietf.org/html/rfc2205`

[5] S. Deering, R. Hinden: *Internet Protocol, Version 6 (IPv6)*, RFC 2460, IETF Network Working Group, December 1998, `http://tools.ietf.org/html/rfc2460`

[6] J. Postel: *Transmission Control Protocol*, RFC 793, IETF, September 1981, `http://tools.ietf.org/html/rfc793`

[7] L. Ong, J. Yoakum: *An Introduction to the Stream Control Transmission Protocol (SCTP)*, RFC 3286, IETF Network Working Group, May 2002, `http://tools.ietf.org/html/rfc3286`

[8] E. Kohler, M. Handley, S. Floyd: *Datagram Congestion Control Protocol (DCCP)*, RFC 4340, IETF Network Working Group, March 2006, `http://tools.ietf.org/html/rfc4340`

[9] Sally Floyd, Kevin Fall: *Promoting the Use of End-to-End Congestion Control in the Internet*, IEEE/ACM Transactions on Networking, Volume 7, August 1999

[10] Murat Yuksel, K. K. Ramakrishnan, Shivkumar Kalyanaraman, Joseph D. Houle, Rita Sadhvani: *Value of Supporting Class-of-Service in IP Backbones*,

[11] Yaqing Huang, Roch Guérin: *Does Over-Provisioning Become More or Less Efficient as Networks Grow Larger?*, Dept. of Electrical and Systems Engineering, University of Pennsylvania, 2005

[12] Mohammad A. Talaat, Magdi A. Koutb, Hoda S. Sorour: *A Survey on Unicast Congestion Control Protocols for Media Traffic*, IJCSNS International Journal of Computer Science and Network Security, Vol. 9, No. 3, March 2009

[13] J. Postel: *User Datagram Protocol*, RFC 768, IETF, August 1980, `http://www.faqs.org/rfcs/rfc768.html`

[14] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, Stefan Savage: *Inferring Internet denial-of-service activity*, ACM Transactions on Computer Systems (TOCS), Volume 24, Issue 2, May 2006

[15] Lefteris Mamatas, Tobias Harks, Vassilis Tsaoussidis: *Approaches to Congestion Control in Packet Networks*, Journal of Internet Engineering, Vol. 1, No. 1, January 2007

[16] R. Braden, D. Clark: *Integrated Services in the Internet Architecture: an Overview*, RFC 1633, IETF Network Working Group, June 1994, `http://tools.ietf.org/html/rfc1633`

[17] Stephen E. Deering, David R. Cheriton: *Multicast routing in datagram internetworks and extended LANs*, ACM Transactions on Computer Systems (TOCS), Volume 8, Issue 2, May 1990

[18] C Diot, BN Levine, B Lyles, H Kassem, D Balensiefen: *Deployment issues for the IP multicast service and architecture*, IEEE Network, 2000

[19] Ian Brown, Jon Crowcroft, Mark Handley, Brad Cain: *Internet Multicast Tomorrow*, The Internet Protocol Journal, Vol. 5, No. 4, December 2002

[20] Pranitha Anand: *Network Address Translation* University of Alabama, Birmingham

[21] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler: *SIP: Session Initiation Protocol*, RFC 3261, IETF, August 2002, `http://www.ietf.org/rfc/rfc3261.txt`

[22] *Internet addressing: Measuring Deployment of IPv6*, OECD, April 2010

[23] E. Nordmark, R. Gilligan: *Basic Transition Mechanisms for IPv6 Hosts and Routers*, RFC 4213, IETF Network Working Group, October 2005, `http://tools.ietf.org/html/rfc4213`

[24] Y. Rekhter, T. Li, S. Hares: *A Border Gateway Protocol 4 (BGP-4)*, RFC 4271, IETF, January 2006, `http://www.ietf.org/rfc/rfc4271`

[25] *CIDR REPORT for 31 May 2010*, `http://www.cidr-report.org`

[26] `http://www.ipv6actnow.org/info/statistics/`

[27] D. Farinacci, V. Fuller, D. Meyer, D. Lewis:

Locator/ID Separation Protocol (LISP), IETF Network Working Group Internet-Draft, March 2009, http://tools.ietf.org/html/draft-farinacci-lisp-12

[28] Bruno Quoitin, Luigi Iannone, Cédric de Launois, Olivier Bonaventure: *Evaluating the Benefits of the Locator/Identifier Separation*, Proceedings of 2nd ACM/IEEE international workshop on mobility in the evolving Internet architecture, 2007

[29] Randall J. Atkinson: *Security for the Internet Protocol*, Naval Research Laboratory, Wasington, November 1995

[30] T. Dierks, E. Rescorla: *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, IETF, August 2008, http://tools.ietf.org/html/rfc5246

[31] S. Kent, R. Atkinson: *Security Architecture for the Internet Protocol*, RFC 2401, IETF, November 1998, http://www.ietf.org/rfc/rfc2401.txt

[32] J. Rosenberg, R. Mahy, P. Matthews, D. Wing: *Session Traversal Utilities for NAT (STUN)*, RFC 5389, IETF, October 2008, http://tools.ietf.org/html/rfc5389

[33] Dave Thaler: *Evolution of the IP Model*, IETF Journal, Volume 4 Issue 3, February 2009, http://www.isoc.org/tools/blogs/ietfjournal/?p=454

[34] Internet Systems Consortium: *The ISC Domain Survey*, http://www.isc.org/solutions/survey