# COMP 535

End-to-End Communication System Design

Simon Foucher (260 223 197)

**McGill University
Monday, Sept 21st 2009**

The internet as we know it today was originally designed as the Department of Defense ARPANET project. Its original nature was to be a robust and versatile internetworking tool to be used by the DoD. Because of the possibility of usage in hostile and unstable war environments rather than for commercial purposes, the survivability of the internetwork was put as a primary goal, and accountability as last. As such, design specifications were drafted in order of importance and the subsequent construction and evolutions of the internet as we know it know reflects some of the fundamental early design decisions taken.

 The first design goal was that host to host communication, once establish and synchronized had to remain uninterrupted, regardless of any changes of state in the network’s infrastructure. As such, the network could for example withstand bombings of some of the gateways installations without interrupting networking activities as long as there remained a path between the hosts. In order to accomplish this primary goal, the state of the conversations taking place had to be stored within the end hosts, and not on the network. Another alternative would have been to use duplication of the data, but this would have cost an enormous amount of extra overhead, and would still not provide the same versatility as end-to-end communication.

Another argument that motivated the usage of end-to-end protocols is that no matter how good low level error checking algorithms work, higher level applications still have to perform some sort of data integrity check anyways, in case some data corruption occurred while the data was traveling back up the OSI layers. Because of this redundancy, early internet designers decided that low level checks and corrections were not a priority and rather opted for a datagram network design, where data was to be segmented in to packets and transported by the network as fast as possible via best effort, with minimal redundancy checks, and coherence checks were left for higher level applications.

The second most important design goal was that the internet had to support many types of communication services with different requirements in terms of speed, latency and reliability. This also pushed the design towards a simple datagram implementation. This simple datagram, the TCP packet, was to be used as a building block for more complex communication requirements, implemented at higher levels. By letting the end users of the network construct complex communication protocols via abstraction layering, the designers avoided the burden of trying to imagine every possible end user requirements in terms of communication and implementing solutions. Besides, even if they did so, it is almost inevitable that at some point, a user would have required a variation of the service provided and ended up implementing the algorithms himself, using simple building blocks.

Because of the size of the project at hand, the internet fathers did have to make some concessions, as a single building block, the TCP packet, was not sufficient to implement the versatility the network required. For example, in VoIP applications, a fast and timely packet delivery is much more crucial than a bit by bit valid data, which could stall a conversation. A few more datagram were therefore implemented, like the UDP packet, which provided application developers with more choice in building blocks. But even with the addition of these extra building blocks, the IP still remained end-to-end oriented.

The use of datagrams also enabled the internet to respect its 3rd most important goal, i.e. that the internet architecture must accommodate a variety of networks. By minimizing the functions the internet provided to the end users, the building blocks were kept simple. This enabled any type of networks capable of transporting datagrams to transport internet traffic.

Since emphasis was made on the primary requirements, compromise had to be made with regards to some of the requirements further down the list. For example, some of the largest issues with modern internet are the lack of tools for distributed management or resources between entities. Another issue found in modern internet is the lack of tools for accounting for packet flows. Since the internet had to be versatile and robust, once a packet is launched in the network, it is hard to determine the path it will take until its destination. Also, observing traffic at a single gateway reveals multiple incomplete data flows because packets are observed in isolation. They contain very little concrete information about the upper layer applications the communications taking place.

Even though the designers felt that providing good underlying performance had to be achieved, they were not able to formalize performance constraints in the architecture. This was mostly due to the fact that one of the primary goals of the internet was versatility and flexibility, and not performance. It is also due to the fact that they failed to implement good tools to measure performance within the network, other than the number of bytes/packets routed within a gateway. With little information on the path the packet is on as well as the upper layer communications taking place, it revealed little, if no information about the performance of the network.

 Besides the few design flaws that only became apparent in modern internet like the lack of tools to coordinate the distributed management that gave the internet its intended versatility, the constructing fathers did achieve most of their goals and manage to create a very robust network. The versatility of its basic building blocks allowed application developers to construct many layers of abstraction over the IP layer and by doing so, develop a multitude of applications we currently see on the internet.