2007-11

Internet Architecture: Lessons Learned and Looking Forward

Xie, Geoffrey

Internet Architecture: Lessons Learned and Looking Forward

Geoffrey G. Xie
Department of Computer Science
Naval Postgraduate School
April 2006
(Revised, December 2006)

Abstract

This chapter explores the architectural design of the Internet. The main objectives are: (i) highlight the design principles underlying the Internet architecture and explain their roles in the success of the network, and (ii) identify some of the limitations of the current Internet architecture and present a possible approach to addressing them.

1. Introduction

In this chapter, we explore the architectural design of the Internet. We believe that a historic perspective is essential for this exploration. Many important lessons about network design can be learned from the evolution of the Internet architecture. The Internet had a very modest start, borne out of an experimental network with a handful of nodes in the late 1960s. There was no comprehensive theory about packet network design in place at the time. It was not until a dozen or so years later, when the Internet had already become a network with about 100,000 nodes that the broad research community started to realize that several early design choices made for the Internet architecture, with an emphasis on *simplicity*, had played a crucial role in its growth and robustness. [Clark88] In other words, it is not by accident that the basic elements of the Internet architecture have withstood the test of time for over three decades, creating the one and only global data networking infrastructure in the process; several architecturally profound design principles were at work. In the first half of the chapter, which includes Sections 2, 3, and 4, we will try to expose as many key points of these design principles as possible while describing nuts and bolts of the Internet architecture.

Examining the Internet architecture from a historic perspective would not be complete without pondering the future of the Internet architecture. In the second half of the chapter, we pose and try to answer the following question: Has the current Internet architecture reached the end of its historical role? To put the question another way: Is a clean-slate design of Internet architecture necessary in order to meet all emerging requirements? We first provide a holistic view of the current Internet architecture based on its division of core functionality into three planes: data, control, and management. We then discuss why the Internet control and management planes have fundamental limitations in coping with several emerging service requirements and why a completely new approach to network control and management may be required. Finally, we describe the 4D network architecture [Greenberg05a,b], which is an instance of a clean slate design of Internet architecture.

For brevity, the discussion will be kept at a high level, with a focus on the fundamental trade-offs behind some of the most important network design choices embodied in the Internet architecture. No complete detail of an individual protocol or mechanism will be provided unless doing so is necessary for the discussion. Almost all important Internet protocols and mechanisms are specified in Internet Request for Comments (RFCs), a collection of documents that is maintained at the official Web site of the Internet Engineering Task Force (IETF). Interested readers are referred there for more information about a specific protocol or concept.

2. Origin of Internet Architecture

The Internet is easily the largest computer system ever built, with tens of millions of nodes running hundreds of protocols. Examining its architecture is foremost about looking beyond the low-level system components and protocols and identifying the set of *core functionalities* that make it tick. The Internet is essentially a network for transporting digital data (i.e., bit streams) between computer processes. In the most abstract form, a network simply consists of nodes connected by links. In the Internet setting, the nodes are computers and the links are connections between computers. To help illustrate the set of necessary functionalities for providing communication services over the Internet, consider a typical computer communication scenario where process A running on one computer wants to transmit a file to process B running on another computer. For this transmission to be successful, the following functions are required:

- *Data Formatting.* A and B must agree on a common data format so that B can extract and reassemble the content of the file from the bit streams received.
- *Addressing*. Process A must have a means to both uniquely identify B from other processes and supply this identification, called B's address, to the network.
- *Routing*. Methods must be in place for determining a feasible path for moving bits from A to B, based on the addresses of A and B.
- *Forwarding*. Methods must be in place for actually moving bits from A to B, through a predetermined sequence of nodes.
- *Error recovery*. Since no physical transmission medium is perfect and bits may be inverted or lost in transit, algorithms are required to detect and correct these errors.

Equally important is the *division of work*, regarding both the creation of distinct node types and the placement of the aforementioned key functions among the nodes. In one design, the network may consist of homogenous nodes, all of which implement one identical set of functions. While conceptually simple, this design may be inflexible and/or incur unnecessarily high cost. It is consideration of this kind of design trade-offs that has shaped the development of the Internet architecture.

In this section, we will introduce and briefly describe a set of design principles that have made the Internet architecture into what it is today. Since these design principles started

as practical solutions to specific network design problems [Clark88], we first look back at history and ground our discussion by laying out the key enabling technologies and the key requirements faced by the architects of the early Internet.
The purpose of introducing the design principles *before* describing the detail of the Internet architecture is twofold. First, we believe that one may appreciate many subtleties of the Internet architecture *better* after having a solid grasp of the big-picture design philosophy of the Internet architecture. Second, as mentioned in the Introduction, the focus of this chapter is on the fundamental design trade-offs and thus we would like to start the discussion at a 30,000 foot level.

2.1. Enabling Technologies

In the pre-Internet era, the communications technology was dominated by a circuit-based approach used by telephone networks. In circuit-switching the bandwidth of each communication link in the network is segmented into multiple smaller transmission channels called circuits, by utilizing either a frequency division multiplexing (FDM) or a time division multiplexing (TDM) technique. [Kurose05] Each circuit can only be allocated to one conversation at a time. Since the number of circuits in the network is finite, a call setup process is required before a conversation can start in order to ensure that there are adequate free circuits to form an end-to-end path between the calling parties.

The static allocation of circuits works well for telephony where the network traffic loads are well understood. However, circuit-switching would cause a significant waste of bandwidth when used to transport computer data traffic such as from a telnet session, which typically is very bursty with long periods of inactivity. This problem motivated people to seek an alternative approach to building networks for linking computer resources, which led to the invention of the *packet-switching* technique in the early sixties. [WebSite1]

In the packet-switching approach, computer data (i.e., bit streams) are transported in small chunks called packets. The capacity of a communication link is not segmented; packets of different users take turns to be transmitted at the full link rate. This form of dynamic sharing of the whole link capacity among different connections, termed *statistical multiplexing*, ensures no waste of link bandwidth as long as there are packets to be transmitted.

Statistical multiplexing requires the use of buffers to hold packets waiting for their turn to be transmitted. This type of buffering naturally led to the birth of a “store and forward” communication paradigm in which packets may be forwarded on a hop by hop basis toward their destinations. Under this paradigm, it is also very easy for intermediate nodes to independently adjust routes that packets take based on current network conditions. Such a dynamic routing capability was quickly recognized as a desirable function of a computer network for resisting link or node failures in the network even before the first packet network was ever built. [WebSite 1]

2.2. Driving Requirements

The Internet began as an experimental network called ARPAnet, which was sponsored by the U.S. Department of Defense (DoD) initially for testing the viability of packet switched computer networks and later for demonstrating ways of combining packet networks that use different link technologies (leased phone lines, satellite, radio, etc.) into one integrated data communications infrastructure for the military. [WebSite1] High on the requirement list for the Internet project were:

1. Robustness – Because of the military sponsorship, an emphasis is put on the ability of the network to continue to operate in the presence of link or node failures.

2. Link heterogeneity – Also important to the military is the network's ability to rapidly assimilate different link technologies so the network can be quickly deployed and extended.

Other requirements for the Internet architecture included the support for multiple types of communications service and distributed management of network resources. [Clark88] Surprisingly, both network security and quality of services (i.e., performance guarantees) are not in the original list of requirements for the Internet. The reason is simple: ARPAnet was originally envisioned as a private data network for the U.S. military and, as such, security was considered more of a physical layer concern and quality of services deemed a non-issue with the assumption that traffic entering the network would be carefully planned resulting in a lightly-loaded network at all times.

2.3. Design Principles

To meet the overriding requirements of robustness and link heterogeneity, the original architects of the Internet made two important design decisions regarding how to organize the core computer networking functionalities. First they recognized that a monolithic network architecture where each switching node can cope with all link technologies will not scale. [Clark88] The concept of adding specialized packet switching nodes, called Internet Message Processors (IMPs), to the network architecture was developed to address that problem and to take advantage of the then new store-and-forward communication paradigm. Each IMP, which we call a gateway or router today, would be an intermediary linking two or more different packet networks. A three-part address format was defined: one for identifying a communicating process, another for identifying the process's host computer, and the last one for identifying the host network. A packet would carry both source and destination addresses in its header. A gateway would only need to inspect the network portion of the destination address when making packet forwarding decisions. Once a packet arrived at the destination network, that network would use the other parts of the destination address to deliver the packet to the receiving process. [Cerf74,WebSite2]

The use of packet-switching gateways not only greatly *simplified* the task of establishing connectivity between independently managed networks with heterogeneous link technologies, but also enabled the store-and-forward paradigm, under which dynamic routing of packets in transit could be done transparent to the communicating processes. However, this design had a distinct performance disadvantage compared to a circuit-switched network: no guarantee of quality of service. While aware of the disadvantage, the Internet architects made a conscious decision to choose simplicity over efficiency. This design choice since has become an overarching design philosophy for the Internet:

Simplicity over efficiency: *Whenever possible, trade efficiency for simplicity*.

The second far-reaching design choice made by early Internet architects can be thought of also as a practice of the "simplicity over efficiency" principle, though with a little twist. Initially, one "super" protocol that combines routing, packet forwarding, and end-to-end reliable delivery functionalities was developed to provide the communication service for all applications. [Cerf74] After closer examination, the Internet architects realized that reliable service may not be a good fit for some applications, e.g., one that exchanges real time voice traffic, to which timeliness is much more important and for which retransmissions are counterproductive by incurring extra latency. This observation inspired the concept of having the network core provide *minimal* packet level forwarding service, upon which different types of data communication services including reliable data transfer would be built at the end hosts of the network. Following this concept, the forwarding functionality was extracted out of the super protocol and made into an independent Internetworking Protocol (IP). Two types of end-to-end communication services were then defined over IP: (i) User Datagram Protocol (UDP) which provides just an end multiplexing point for IP packets of different application processes running on the same host; and (ii) Transmission Control Protocol (TCP) which has added functionality to support reliable data transfer.

By treating packets independently and forwarding them based on their destination network address, the IP protocol does not require the gateways to maintain any connection state about application processes. This flexibility has greatly *simplified* the design of gateways, the main piece of the technology puzzle for connecting new networks into the Internet, and therefore should be considered one of the crucial factors for the rapid growth of the Internet. Throughout the years, the Internet architects have upheld this design choice of a "thin" internetworking layer, resulting in the following design principle for division of functionalities required for building communication services over the Internet:

Datagram service: *Provide simple connectionless packet forwarding service in the network core*.

The minimalistic approach to gateway design was later justified and generalized into the so called end-to-end argument for placement of functions among modules of a distributed

computer system. [Saltzer84] The argument can be as stated succinctly as below [Saltzer84]:

End-to-end argument: *Functions placed at low levels of a system may be redundant or of little value when the cost of providing them at the low level is factored in.*

The Internet is a distributed system with two levels of functionality: the network subsystem at the lower level providing communication services to application clients at the upper level. Under the end-to-end argument or design principle, networking functions that deal with network anomalies such as bit errors, node crashes, packet duplications, buffer overflow, would be best implemented at the end hosts where application client processes reside, particularly when the occurrences of anomalies are too infrequent to make it cost effective to place corrective functions inside the network. [Saltzer84]

3. Current Internet Architecture

Today, the Internet has evolved from a U.S. military system prototype into an open, world-wide infrastructure over which a rich set of applications, including Web, E-business, voice over IP (VoIP), video broadcast, and on-line gaming, is deployed. These applications have imposed additional performance and security challenges on the network. New elements have been incorporated into the Internet architecture in an attempt to address these challenges. In this section, we delve into the major building blocks of the Internet architecture: describe their *current* functionality and trace their evolution path.

3.1. Data Formatting

In the Internet, all types of digital information are encapsulated in packets with a standard format defined by the IP protocol. [RFC 791] At a minimum, a host needs to be able to send and receive packets in that format in order to be connected to the Internet. As discussed in Section 2.3, two types of end-to-end communications service (or transport protocol), TCP and UDP, have been defined on top of IP. Moreover, each application has its own set of agreements on the message format and the method of exchanging these messages. For example, a Web browser uses the HyperText Transfer Protocol (HTTP) protocol [RFC 1945, RFC 2616] to communicate with a Web server. Therefore, the packaging of application data into IP packets at an end host involves several layers of encapsulation, as described below.

3.1.1. Packet encapsulation

Figure 1 illustrates the typical packet encapsulation process at an end host. The "Application data" box represents the sequence of bits for an application-specific message (e.g., an HTTP message requesting a Web page) that is to be sent from the host. In the first encapsulation step, the message is encapsulated in *one or more* transport-layer

segments with the same TCP or UDP header. (See RFC 793 and RFC 768, or a networking textbook such as [Kurose05], for details about TCP and UDP.) In the next step, each transport-layer segment is encapsulated in an IP packet by prepending an IP header. In the final step, a link-layer frame is created by prepending an additional header and possibly a trailer. The link-layer frame format may vary from network to network, specific to the link layer technology used in each network. For example, the Ethernet format would be used here if the host were part of a local area Ethernet network. It should be noted that the link-layer header and trailer will be removed before the packet is passed to router modules that make routing and forwarding decisions.

Also, each link-layer technology defines its own Maximum Transfer Unit (MTU) parameter: the maximum number of bytes that can be encapsulated in one frame. For example, the MTU size for the Ethernet protocol is 1500 bytes; therefore, the size of an IP packet cannot exceed 1500 bytes in an Ethernet environment. That's why the application message may have to be encapsulated in multiple transport layer segments. Since there is not a standard MTU size across all link-layer technologies, it may also happen that an IP packet is forwarded to a network with an MTU smaller than the packet's size. Should the packet be discarded or fragmented into multiple packets of an appropriate size? We defer this topic to Section 3.3 after we have a chance to inspect the IP header format.

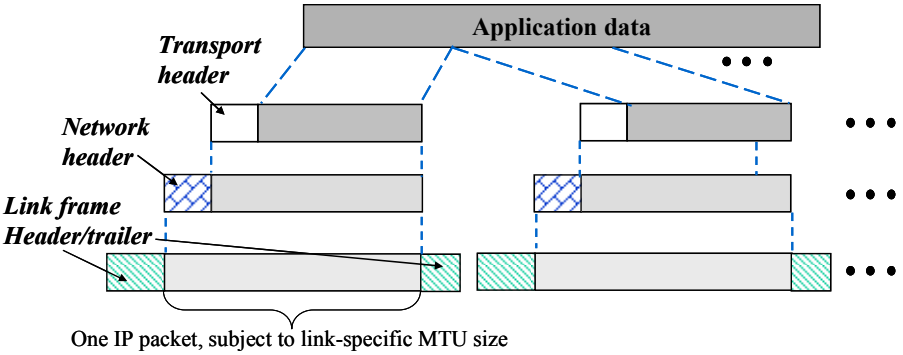

Figure 1: Packet encapsulation. From a router's perspective, all types of application data are encapsulated in IP packets. These packets are delivered using different link-layer technologies hop by hop.

3.1.2. IP header format

The IP header format is shown in Figure 2. The individual fields are defined as follows [RFC 791]:

- *Version*: a 4-bit value indicating the version of the IP protocol used. Currently, we are at version 4, abbreviated as IPv4. Some experimental networks are running IP version 6 (IPv6). We will briefly discuss the features of IPv6 in Section 3.2.3.

- *Internet Header Length (IHL)*: a 4-bit value indicating the length of the IP packet header (including all the option fields and padding), in 4-byte words. For example, a packet with a 24-byte long header would have this field set to 6.
- *Type of Service (ToS)*: an 8-bit value indicating the quality of service desired for the packet. This field had not been widely used until the differentiated service (DiffServ) model was introduced in the mid-1990s. We will briefly discuss DiffServ in Section 3.4.2.
- *Total Length:* a 16-bit value indicating the total length of the IP packet in bytes, including the header and the data payload encapsulated in the packet.
- *Identification, Flags, and Fragment Offset:* These fields are used by the IPv4 fragmentation and assembly algorithm, which we will describe in the next subsection.
- *Time to Live (TTL)*: an 8-bit value representing an upper bound on the number hops (routers) that packet can still traverse. Any router module that forwards packets must (i) decrement each packet's TTL field by at least one and (ii) discard a packet with a zero TTL value. The intention is to cause undeliverable packets to be discarded, and to place an upper bound on the maximum packet lifetime. Upon discarding a packet with a zero TTL, the router will notify the sender of that packet of this action via the Internet Control Message Protocol (ICMP). (ICMP is defined in RFC 792.)
- *Protocol:* an 8-bit value identifying the next level protocol that is encapsulated. The id's for various protocols, called protocol numbers, are specified in RFC 790. For example, the protocol number for TCP is 6.
- *Header Checksum:* 16-bit one's complement of the 1's complement sum of all 16-bit half-words in the header. For purposes of computing the checksum, the value of the checksum field is zero.
- *Source Address and Destination Address:* 32-bit addresses for the source and destination network interfaces, respectively. The specifics of the IP addressing scheme will be examined in the next subsection. It should be noted that an end host may have multiple network interfaces; in that case, this host is said to be "multi-homed" and any one of the interfaces can be used to reach an application running on this host.
- *Options:* The field may specify the presence of optional IP functionality. One of the existing options is source routing, whereby a pre-determined sequence of routers that must be traversed by the packet can be specified.

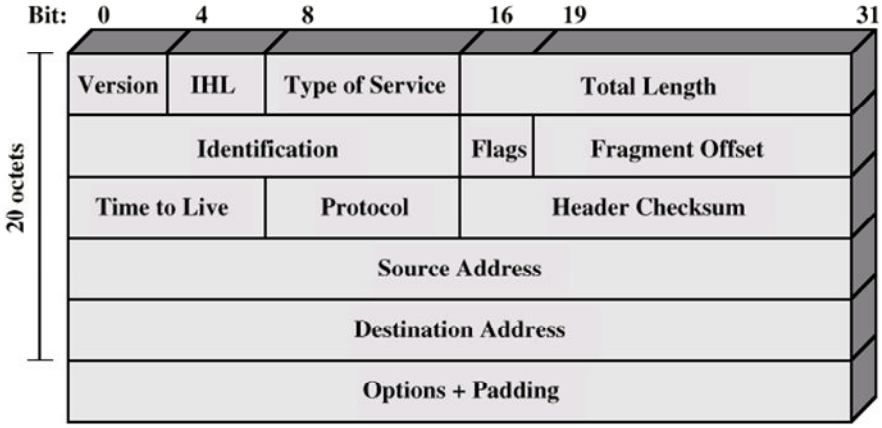

Figure 2: IP header format. [Figure 18.6 of Stalling04]
Each row represents a 4 byte word. The total length is always a multiple of 4 bytes after possible padding.

In summary, the IP header format is quite simple, reflecting the desire for imposing minimal requirements for connecting new hosts into the Internet. In accordance with the datagram service, no header fields are provided for recording session-specific state and each packet is required to carry destination address information.

3.1.3. Packet fragmentation and reassembly

The "Identification" field in the IP header is designed to give a unique identifier to each packet upon its generation at a host. Different host operating systems (Windows, Linux, etc.) may use different algorithms for setting this field. Most commonly, the value is derived from reading the host's system clock. When the packet encounters a network with an insufficient MTU size and thereby needs to be fragmented, i.e., broken down into several smaller packets, the same identifier is inherited by all the fragments. Additionally, the 13-bit "Fragment Offset" field of each fragment packet is calculated based on where the first byte of the fragment is situated, in length of 8-byte double-words, relative to the start of the original packet. One of the bits of the "Flags" field, the More Fragments (MF) bit, is set for all fragments except the last one. This information allows the receiving host to identify and assemble the fragments back into the original packet. It is possible that these fragment packets need further fragmentation at another network with an even smaller MTU. However, reassembly is done only once for the packet at the receiver.

The IP protocol allows the fragmentation/reassembly feature to be turned off by setting another bit of the "Flags" field, the so-called Don't Fragment (DF) bit. When that bit is set, the packet would be dropped under the scenario described in the previous paragraph. Additionally, the router that dropped the packet would send an error notification message to the sender of the packet via ICMP.

The design of the IP fragmentation/reassembly functionality can be viewed as an application of the "Simplicity over Performance" principle. It certainly has a negative performance impact on the routers that have to perform the checking and the fragmentation. But it avoids imposing a standard MTU size on every link technology and thus removes a potential barrier for connecting a new type of network into the Internet. Instead, a minimum MTU is instituted, which is 576 bytes for the current version of IP (version 4) and 1280 bytes for IP version 6.

3.2. Addressing

The basic approach of the three-level hierarchical addressing scheme as described in Section 2.3 remains unchanged through the years. More specifically, the process portion of the address definition, called *port*, has been standardized as part of both TCP and UDP header formats, and the network and host portions of the address definition have been combined into one 32-bit value, called *IP address*, which should be globally unique. For ease of writing, an IP address is usually represented in the so-called dotted-decimal form, in which the 32 bits are partitioned into four bytes written in their decimal form and separated by a period (dot). For example, "192.128.10.168" is the IP address assigned to the network interface installed on the laptop that I use to write this chapter.

3.2.1. Subnet and Subnet Mask

As shown in Figure 3, an IP address is composed of two continuous blocks of bits. The left block contains the "Network bits" and identifies the home network (or subnet) for the host. All network interfaces in one subnet should be assigned the same block of network bits, called the *subnet prefix*. The right block contains the "Host bits", which should be uniquely assigned to each network interface installed on a host in the subnet.

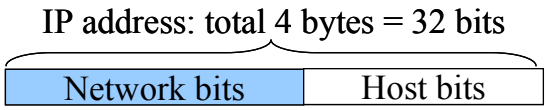

Figure 3: IP address layout

To reduce the size of the forwarding tables at routers, routing in the Internet is done based on matching subnet prefixes instead of matching the entire 32-bit addresses. Accordingly, one or more gateway routers should be attached to each subnet. The Internet routing and forwarding algorithms are only responsible for transporting packets to these gateways, which then deliver the packets to the receiving hosts (or more precisely, the network interfaces of these hosts) using a technology specific link protocol. Similarly, packets destined for a remote host (i.e., not in the same subnet as the sending host) also have to depart through the gateways.

Clearly, a method for determining the boundary between the network and host blocks of an IP address is required for extracting the right subnet prefix. For example, "192",

"192.128", "192.128.10", are just a few of the possible subnet prefixes for the IP address "192.128.10.168". The original solution was to define several classes of network prefixes, each with a unique starting bit pattern, and then assigning a fixed number of network bits per class. Specifically, class A network prefixes start with binary "0" and always have 8 network bits. Therefore, there are a total of 128 possible class A networks, each with a total of $2^{32-8} - 2 = 16{,}777{,}214$ unique addresses to assign to its network interfaces. Class B network prefixes start with binary "10" and always have 16 network bits. Finally, class C network prefixes start with binary "110" and always have 24 network bits.

Later, the class based network assignment was found to be inflexible and wasteful in terms of the percentage of addresses actually assigned per network. For example, a company with a dozen of employees would be unlikely to fully utilize even a class C network address space that contains $2^{32-24} - 2 = 254$ valid host addresses. The subtraction of 2 is needed because two special addresses are preconfigured for each subnet and not available as host address: one for broadcast, with all host bits set to 1, and the other being the subnet id, with all host bits set to zero. The current solution is to allow subnets of any size and to require each subnet to explicitly declare the exact number of network bits that it is assigned. For example, the subnet where my laptop is located has a prefix of "192.128.8.0/22". The "/22" notation indicates the prefix is made of 22 network bits. "192.128.8.0" is the *subnet id* and is obtained from zeroing out all the host bits (i.e., the rightmost 10 bits) in "192.128.10.168". The zeroing step can be easily accomplished by a bit-wise AND operation between "192.128.10.168" and a special address "255.255.252.0", which is obtained by setting all the network bits and clearing all the host bits. This special address is called the *subnet mask*. Under this scheme, a router can easily determine if a given destination address matches one of the network prefixes in its forwarding table: for every network prefix in the table, first calculate a subnet id value for the IP address using the prefix's subnet mask and then match it against the subnet id for that prefix.

3.2.2. DNS and DHCP

Even in the dotted decimal form, IP addresses are not easy for humans to remember. Also, *manually* configuring every network interface with the right combination of IP address, gateway address, and subnet mask can be a challenge for a large network with hundreds of hosts. Fortunately, two protocols have been developed to help us with these tasks.

The Domain Name Service (DNS) protocol allows an application to refer to a host by its fully qualified domain name (FQDN), which is also in a dotted format, albeit with words in a string format, not decimal numbers. For example, the FQDN of my laptop has been set to be "xielap.cs.nps.navy.mil", which is a lot easier to remember than "192.128.10.168". "xielap" is the *host name* chosen by myself, while "cs.nps.navy.mil" is the name for the *network domain* of my work place – Department of Computer Science of Naval Postgraduate School. The "." operator in an FQDN denotes the *part-of* relationship: "cs.nps.navy.mil" is part of the "nps.navy.mil" domain, "nps.navy.mil" is

part of "navy.mil", and so on so forth. All network domains can be thought of being part of a root domain. A network domain may contain multiple subnets and subnets of the same domain are typically administrated by one domain authority. Since the IP protocol does not recognize FQDNs, DNS provides applications a service that translates an FQDN into its IP address counterpart, and vice versa Each domain authority is responsible for setting up a DNS server that maintains a mapping between the FQDNs and IP addresses of all local hosts in its domain and answers queries about the mapping from either an application running on a local host or a remote DNS server on behalf of a remote application. A detailed specification of the DNS protocol is given in RFC 1035.

The Dynamic Host Configuration Protocol (DHCP) supports auto-configuration of a network interface. To enable this feature on a subnet, one must set up a DHCP server and assign to it a pool of free IP addresses for that subnet. When a host in the subnet is booted up, it will use broadcast to reach, maybe via a relay agent, the DHCP server and request an IP address and other information required for configuring its network interface. In response, the server will typically remove an address from the free address pool and allocate it to the host for a fixed duration. The host will contact the server to renew the allocation periodically when it stays up for a long time. The host may also contact the server to explicitly return its address to the pool of free addresses, e.g., when the host is being shut down. A detailed specification of DHCP can be found in RFC 951.

3.2.3. NAT and IPv6

In the mid 1990s, due to years of explosive growth of the Internet, a crisis of IP address shortage seemed looming. Theoretically, the 32-bit address space provides close to 4 billion unique addresses. However, even with classless network allocation, a significant percentage of allocated addresses are unused and thus wasted. Also, many addresses are reserved for special purposes: broadcast, multicast, private, etc., and not available for hosts. Therefore, when people started to talk about connecting everything, from cell phones to toasters, to the Internet, it was perceivable that the pool of unallocated IP addresses could soon run dry. In response to this crisis, two major solutions were proposed. One was supposedly a near-term fix which requires no change of the current IP address format and the other was touted as a long-term solution which changes the address format to make it 128-bit long.

The near-term solution is called Network Address Translation (NAT), specified in RFC 1631. In this scheme, a network's external connection must go through a NAT server. The server replaces the source address and source port pair of every outgoing packet with its own IP address and a "surrogate" source port that is unique to a specific source address and source port combination. In other words, there is a one-to-one mapping between a source address and source port combination to a "surrogate" port. The NAT server records such mappings in a hash table keyed by the value of the "surrogate" port. The server also inspects every incoming packet, uses the packet's destination port field as the key to find the right address mapping for the packet, overwrites the destination address and destination port fields with the ones obtained from the mapping, and finally

forwards the packet on inside the network. Thus, it is not necessary to assign globally unique addresses to hosts inside the network. A common practice is to use private address ranges for these hosts. Only the NAT server needs to have a globally unique address, making NAT an effective solution for mitigating the address shortage problem.

The long-term solution requires upgrading the IP protocol (version number 4) to a new version, IP version 6 (IPv6). In addition to a drastically larger address space than IPv4, IPv6 also includes new features such as auto-configuration and a streamlined header structure. Interested readers are referred to RFC 2460 and [Davies02] for details about IPv6.

3.3. Dynamic Routing

Routing in the context of the Internet is about maintaining consistent forwarding tables at the routers, in accordance with the network's store and forward communication paradigm. In the early days of the Internet, routing was not a major issue because of the small number of networks connected to it. Routing within a network was often done with a private protocol and routes between networks were static and manually set up. [Clark88] Later, as the size of the Internet grew, dynamic routing became necessary as the topology of the network constantly changed.

Before we delve into routing, let's briefly look at how forwarding is done given consistent forwarding tables at all routers. As mentioned in Section 3.2.1, the forwarding is done by matching subnet prefixes. A typical forwarding table at a router, often referred to as the Forwarding Information Base (FIB) for the router, contains entries (i.e., routes) in the format of: <network prefix>, <next hop>, <metric>. <network prefix> is the subnet prefix of the destination network for this route, <next hop> the interface to use as part of this route to reach the destination, and <metric> a measure of goodness of this route. To forward a packet, the router first looks up its FIB with the packet's destination address and looks for subnet prefix matches using the method described in the end of Section 3.2.1. When the packet's destination address matches multiple routes in the FIB, the router chooses the route with the *longest prefix match*, i.e., with the most matching network bits. If there are more than one longest prefix matches, the router uses the <metric> field to break the tie. After determining a route, the router forwards the packet to the output port as defined by that route's <next hop>.

3.3.1. Hierarchical organization of networks

To scale to millions of networks, a two-level routing hierarchy has been defined for the Internet, analogous to the "first state, then city, and finally street" way of delivering mail by the post offices. At the top level, routing is done among network domains. Each domain is designated as an autonomous system (AS) and assigned a unique 16-bit *AS number* for routing purposes. For example, my school's domain "nps.navy.mil" is also AS 257. Currently, there are about 30,000 active ASes in the Internet. At the bottom level, within an AS, routing is done between routers inside that AS. This "divide-and-

conquer" approach ensures that each router only needs to maintain a relative small number of routes in its FIB. The largest FIB reported has about 200,000 entries, which is a lot smaller than the total number of subnets in the Internet.

Because ASes are independently administrated, different intra-domain routing protocols may be deployed in different ASes. This is not the case for inter-domain routing; all ASes must implement the same inter-domain routing protocol to achieve full interoperability.

3.3.2. Intra-domain routing

Typically, a single administrative authority has total control over all routers in an AS. Such control makes it possible to consider additional performance objectives when designing intra-domain routing protocols. The current generation of intra-domain routing protocols are designed to rank routes based on a distance metric defined as follows. First, each link connecting two subnets is given a cost metric. The distance of a route is the sum of the costs of all links traversed by that route. The simplest definition of link cost would be assigning a value of "1" to every link. In that case, the distance of a route would be its *hop count*, i.e., the total number of links it traverses. In summary, an intra-domain routing protocol is about computing shortest paths between all pairs of subnets within an AS.

Two classes of protocols have been developed for dynamically computing shortest paths between routers: *distant vector* and *link state*. A distant vector protocol is totally distributed, based on *iterative* computation of shortest paths. A router only communicates with a direct neighbor and exchanges updates on each other's distance vector, i.e., a table of current minimum distance to all known destinations. A neighbor's update message may trigger a new update at the router following the iterative Bellman-Ford algorithm. Much like human gossiping, the router will eventually know about every other router in the network, the correct minimum distance to it, and the neighboring router to use to achieve that minimum distance. Routing Information Protocol (RIP), one of the earliest dynamic routing protocols designed for the Internet and currently at version 2 [RFC 2453], is a primary example of distance vector routing protocols.

A link state protocol is centralized in the sense that each router will first obtain a *global* view of the network, including topology and link costs, through flooding of link state packets by all routers and then *independently* apply the Dijkstra's algorithm to compute the shortest paths. If all routers have the same global view, then the FIBs built by the routers will be consistent. Upon a change in the network topology or link cost, the affected router(s) will flood new link state packets to update the global view at each router and trigger new shortest path computation. Open Shortest Path First (OSPF) and Intermediate System-Intermediate System (IS-IS), specified in RFC 2328 and RFC 1142 respectively, are the two most prominent link state routing protocols.

3.3.3. Inter-domain routing

The AS-level topology is a mesh, with each AS having connectivity with one or more other ASes based on business agreements. Because ASes are independently managed, there is no uniform scale for the link cost metric across different ASes. So determining the distance between two ASes based on adding link costs is not very meaningful. Instead, policy is more important in inter-domain routing. For example, an Internet Service Provider (ISP) may choose to avoid a particular AS (belonging to a competitor) in all its routes. To assist policy based routing, the Internet uses a *path vector* protocol called Border Gateway Protocol (BGP) for inter-domain routing. [RFC 4271] Each AS sets up one or more BGP border routers for exchanging path vectors, each of which is a full sequence of ASes to use to reach a specific destination network, with border routers of neighboring ASes. In general, an AS will advertise a route learned from one neighbor to other neighbors after appending its own AS number to that route. Policy-based actions may be specified in three stages of BGP operation at a border gateway. First, import filters may be placed to reject certain routes received from a neighboring AS. Second, policy may be defined regarding how multiple imported routes for the same destination are ranked. Third, export filters may be placed to restrict the scope of route advertisements to neighboring ASes.

3.4. Resource Allocation

Resource allocation did not receive serious consideration in the original design of the Internet architecture because of the datagram service principle. However, as the reach of the Internet extends and the access speed increases, latency or loss sensitive applications such as video phone start to be deployed. These applications require the network to provide some minimum level of performance guarantee with respect to throughput, packet delay, packet loss rate, etc. A new catch phrase, quality of services (QoS), has since been coined by the networking community to refer to the level of performance guarantee a computer network provides.

While some people still view over-provisioning, i.e., making bandwidth so abundant that link congestion is unlikely, as a viable solution to all QoS problems, both the network research and operational communities have recently explored alternative solutions aimed at avoiding link congestion through elaborate resource allocation schemes. These efforts are described below.

3.4.1. Traffic engineering

Traffic engineering involves adapting the flow of packets through the network based on a given set of performance objectives and the traffic matrix, i.e., the observed *typical* volume of traffic from each ingress point to each egress point. Often, a network designer needs to deal with conflicting performance objectives, such as minimizing the maximum link utilization and bounding the propagation delay between each pair of routers. Satisfying them simultaneously for a dynamic network environment is very challenging under the current Internet architecture. A good commentary on the existing traffic engineering approaches is given in [Rexford04], which we will quote below:

"Early attempts to engineer the flow of traffic involved extending the routing protocols to compute load-sensitive routes in a distributed fashion. In these protocols, the cost of each link is computed as some (possibly smoothed) function of delay or utilization, in order to steer packets away from heavily-loaded links. However, routing oscillations and packet loss proved difficult to avoid, since routers were computing routes based on out-of-date information that changed rapidly, and the effort was eventually abandoned. To improve stability, the distributed algorithms were extended to compute a path for groups of related packets called *flows*. These load-sensitive routing protocols can have stability problems as well, unless the dynamic routing decisions are limited to aggregated or long-lived flows. Perhaps more importantly, the protocols require underlying support for signaling and distributed algorithms for optimizing paths for multiple metrics.

Many existing IP networks have instead adopted a centralized approach for engineering the flow of traffic using traditional IP routing protocols (e.g., OSPF). In this scheme, the management plane collects measurement data to construct a network-wide view of the offered traffic and the network topology. Since the optimization of the OSPF weights is an NP-complete problem, the management plane conducts a local search through candidate settings of the link weights, looking for a solution that satisfies the various performance objectives. Considering additional performance metrics is as simple as changing the objective function used to evaluate the solutions. However, this approach has its limitations in satisfying different metrics for traffic to different destinations, and for avoiding disruptions during failures and planned maintenance. Ultimately, having a single integer weight on each link is not sufficiently expressive, though this approach has proven very useful in practice."

3.4.2. Integrated Services (IntServ) Model

In the early 1990s, the network research community made a serious attempt to extend the datagram service model and retrofit a QoS solution over the Internet. The effort was motivated by the seminal work of Parekh and Gallager which shows that the end to end delay of one application's packets can be upper bounded regardless of the behaviors of other applications if an appropriate packet scheduling algorithm, such as Packetized Generalized Processor Sharing (P-GPS) or weighted fair queuing (WFQ), is used at every output link that the packets traverse. The new service model was named Integrated Services (IntServ) after its lofty goal to meet the QoS requirements of all types of application data including interactive audio and video. [RFC 1633] The core of the service model is a new type of service called "guaranteed service", which provides a deterministic (i.e., for 100% of the packets) guarantee of performance, in terms of maximum end-to-end packet delay and minimum throughput, on a per application basis. All packets for an application that has subscribed to this service traverse the same set of links, and are referred to as a *flow*. The flow has a separate buffer at each output link and receives a guaranteed rate of service from the packet scheduling algorithm based on the flow's bandwidth requirement. Clearly, in order for the guaranteed service to work, the application must reserve network resources (link bandwidth, buffer, etc.) along a network

path ahead of time. A protocol called RSVP (resource ReSerVation Protocol) has been developed to facilitate this task. RSVP is specified in RFC 2205.

3.4.3. Differentiated Services (DiffServ) Model

IntServ is elegant in theory. However, the research community soon realized that IntServ might not be able to scale to the size of the Internet because it requires per-flow state to be maintained at all routers, including the backbone routers which may have to deal with millions of flows concurrently. An alternative solution was quickly developed. The solution centers on a Differentiated Service (DiffServ) model, in which only inter-class *performance differentiations* are guaranteed over a small number of service classes. Neither per-flow nor absolute, quantitative service guarantees are provided. Three DiffServ service classes have been well defined, in the order of increased performance: Default Forwarding which is the same as the default best-effort service offered by the Internet, Assured Forwarding which ensures a sustained throughput, and Expedited Forwarding characterized by low loss, low delay, and low jitter. It is up to each network provider (ISP) to choose particular packet scheduling and queue management algorithms at each of its routers to support the required Per-Hop forwarding Behaviors (PHBs) for the defined service classes.

DiffServ achieves scalability by implementing complex classification and conditioning functions (metering, marking, shaping, and policing) only at access routers at the edge of the Internet. These functions are carried out based on the service level agreements (SLAs) between network customers and providers. The core routers need to allocate buffer and bandwidth only on a per service class basis while applying PHBs to aggregates of traffic which have been appropriately conditioned and marked using the ToS field in the IP header by edge routers. The details of DiffServ can be found in RFC 2474 and RFC 2475.

3.4.4. Multi-Protocol Label Switching (MPLS)

MPLS is the latest attempt of the computer networking community to retrofit a connection-oriented forwarding service over the Internet. Such a forwarding service not only is conceptually appealing and but also streamlines resource allocation. Strictly speaking, MPLS is not a network layer protocol; it operates between the link and network layers and independently of the IP protocol. It is called “multi-protocol” because its 4-byte header format has been incorporated into the frame headers of different link technologies. For an Ethernet link, the header is simply appended to the front of an Ethernet frame. For an ATM link, the 32 bits of the ATM cell header fields, virtual path id (VPI) and virtual circuit id (VCI), are re-designated to carry the MPLS header fields. The MPLS header contains mainly a 20-bit “label” field which serves a connection id and a 3-bit “class of service” (CoS) field for support of QoS differentiation. A label switching router (LSR), one that is able to process MPLS packets, will use this label to make the forwarding decision and bypass the IP header. The LSR will also overwrite the label with a new value that is anticipated by the downstream LSR. That is why the protocol has “label switching” in its name. All the label values are determined ahead of

time as part of the MPLS connection (tunnel) set-up process performed by a Label Distribution Protocol (LDP).

MPLS is mostly used by an ISP as a local traffic engineering solution, often combined with DiffServ mechanisms. Typically, a set of MPLS tunnels is preconfigured within the ISP's network. The ingress routers of the ISP classify all arriving packets based on their header fields, and insert corresponding MPLS header fields at the output link for those classified to be transported by one of the MPLS tunnels. More information about MPLS can be found in RFC 3031.

3.5. Security

For the reason explained in Section 2.2, security was not high on the original list of goals for the Internet. Today, with the Internet becoming an open infrastructure for E-commerce and E-government, security is one of the most pressing issues faced by the Internet community. Several security mechanisms such as firewall, virtual private network, transport layer security, secure email, and public key infrastructure (PKI), have been added to the Internet architecture with some level of successes. Two of them are described below.

3.5.1. Firewall

A firewall is a combination of specialized hardware and/or software acting as a network's *security gate* that can restrict types of communication between the network and the public Internet, mainly to prevent unauthorized access to the network's resources. Typically, a firewall administrator has configured the firewall with a set of packet filtering rules based on security policy. The firewall inspects the header fields of all packets that come in and out of the network and drops those matching the filtering rules. For example, a firewall may only allow Web traffic to come in the network by filtering out all packets that don't carry an HTTP payload. A firewall can also be *stateful* in that it will try to enforce certain communication patterns involving several packet exchanges. For example, a stateful firewall will deny a TCP connection response (so-called TCP SYN-ACK message) from coming in if it has not seen a corresponding connection request going out.

3.5.2. Virtual Private Network (VPN)

Often an organization spans multiple geographical locations. It's very expensive for this organization to build a private data network with leased lines to connect all its sites. An alternative approach is to use the public Internet for connectivity and rely on additional security protocols for data privacy, resulting in a virtual private network (VPN). Currently, most VPNs are built by setting up a VPN proxy between each site network and the Internet. The proxies run a tunneling protocol that allows packets for this VPN to be encrypted and encapsulated with additional headers at the proxy of the source site and then decrypted and de-capsulated at the proxy of the destination site. By treating the network as black box, VPN is a design based on the end-to-end argument. There are two

types of VPN tunneling protocols: some, like L2TP, run at layer 2 (link layer), and the others, like IPsec, run at the layer 3 (IP layer). L2TP and IPsec are defined in RFC 3931 and RFC 4301, respectively.

In summary, the current security techniques for the Internet focus on establishing a security perimeter around a network and preventing unwanted traffic from coming in. Very little can they do to defend against attacks originated inside the security perimeter. It is also very difficult to verify if the security perimeter has been properly configured or if the security perimeter will hold in the event of link or node failures. [Xie05]

4. Future of Internet Architecture

Starting from this section, we will look forward and examine the future of the Internet architecture. Several stirring proposals like [Clark03] and [Greenberg05b] have come out recently calling for a clean slate design of the Internet architecture. Regardless of whether they will stand the test of time, these proposals constitute serious efforts aimed at understanding the limitations of the current Internet architecture and seeking future directions in network design. Due to space constraints, the rest of the discussion will be based mainly on one of them, called the 4D architecture. [Greenberg05b]

The 4D architecture was conceived by a team of researchers from Carnegie Mellon, Princeton, AT&T Research, and Naval Postgraduate School, including the author of this chapter. The 4D architects argue that the current Internet architecture has reached the end of its historical role because it does not have intrinsic capacity to meet emerging QoS and security requirements and the bandage solutions such as presented in Sections 3.4-3.5 are creating an even bigger problem by inducing bewilderingly high network management complexity. [Greenberg05a,b]

To better understand this argument, let's introduce another abstraction of the Internet architecture based on the time scale of execution of its constituent functions. Specifically as described in [Rexford04], the current Internet architecture can be decomposed into three planes:

> **"Data plane:** The data plane is local to an individual router, or even a single interface card on the router, and operates at the speed of packet arrivals, *down to nanoseconds per packet*. For example, the data plane performs packet forwarding, including the longest-prefix match that identifies the outgoing link for each packet, as well as the access control lists (ACLs) that filter packets based on their header fields. The data plane also implements functions such as tunneling, queue management, and packet scheduling.
>
> **Control plane:** The control plane consists of the network-wide distributed algorithms that compute parts of the state in the data plane. The convergence times of these algorithms *vary from seconds to minutes*. For example, the control plane includes BGP update messages and the BGP decision process, as well as the Interior Gateway Protocol (such as OSPF), its link-state advertisements (LSAs), and the Dijkstra's

shortest-path algorithm. A primary job of the control plane is to compute routes between IP subnets, including combining information from each routing protocol's Routing Information Base (RIB) to construct a single Forwarding Information Base (FIB) that drives packet forwarding decisions. Currently, the control plane exhibits the classic symptom of an over-engineered yet *unstable* system: the decision logic (e.g., for controlling reachability) is spread across multiple independently configured protocols or mechanisms and a local configuration error may cause cascading network-wide failures.

Management plane: The management plane stores and analyzes measurement data from the network and generates the configuration state on the individual routers. For example, the management plane collects and combines Simple Network Management Protocol (SNMP) statistics, traffic flow records, OSPF LSAs, and information extracted from BGP update message streams. A tool that configures the OSPF link weights and BGP policies to satisfy traffic engineering goals would be part of the management plane. Similarly, a system that analyzes traffic measurements to detect denial-of-service attacks and configures ACLs to block offending traffic would be part of the management plane."

From this view of the Internet architecture, the 4D architects have identified the following problem: The management plane is currently the only place where decisions are made based on *network-wide* information to meet *network-level* performance objectives, but placing these control functionalities in the management plane suffers from two fundamental weaknesses. First, the time scale of their operation is too long for them to adapt to changing network conditions without causing noticeable periods of severe performance degradation. Second, the management plane does not have direct control over the data plane, or more precisely, the FIB entries at the routers. The decisions made at the management plane have to be carried out through setting specific protocol parameters (e.g., OSPF weights) in the control plane. However, determining the right protocol parameters is often an NP-hard problem, requiring complex modeling and inverting of the actions of the control plane. Furthermore, this type of indirect control creates a performance bottleneck since it is not conducive to an integrated view of different mechanisms and joint optimization of multiple metrics. [Rexford04] and [Greenberg05c] provide several detailed examples of this problem.

The 4D architects argue that network-level decision making at a faster time scale and direct control are necessary to meet stringent QoS and security requirements. Continuing on the current evolution path by tweaking the management and control planes will not fundamentally address either of the issues. A revolutionary change to the Internet architecture is inevitable.

5. The 4D Architecture

The design of the 4D architecture centers on streamlining network-level decision making and execution. Conceptually, network control and management functions are refactored into four planes: Decision, Dissemination, Discovery, and Data; thus the name 4D.

Before we delve into the role of each plane in the 4D architecture, we follow the main theme of this chapter by first presenting three design principles behind the refactoring of functions.

5.1. New Design Principles for Network Control

The 4D designers carefully researched the root causes of some of the major problems plaguing the current Internet architecture, to identify the desirable features for a new approach to network control. [Rexford04,Greenberg05a]. The problems examined included reachability control, traffic engineering, and planned maintenance. The effort has led to the formulation of the following design principles for crafting a more robust network architecture. [Greenberg05a,b]

Network-level objectives: *A network should be configured via specification of the requirements and goals for its performance, which should be expressed separately from the low-level network elements.*

Network-wide views: *Timely, accurate, network-wide views of topology, traffic, and events are crucial for running a robust network.*

Direct control: *Satisfying network-level objectives is much easier with direct control over the configuration of the data plane. The control and management system should have both the ability and the sole responsibility for setting all the state in the data plane that directs packet forwarding.*

5.2. Refactoring of Network Control and Management Functions

Guided by the design principles above, the 4D proposal refactors network control and management functions into four planes, as illustrated in Figure 4. Below is a brief description of the main functions of each plane. [Greenberg05a,b]

Decision plane: The decision plane makes *all* decisions driving network control, including reachability, load balancing, access control, security, and interface configuration. Replacing today's management plane, the decision plane operates in real time on a network-wide view of the topology (e.g., layer-3 topology, as well as layer-2 and layer-1 inventory), the traffic (e.g., the traffic matrix), and the capabilities and resource limitations of the routers. The decision plane directly configures the data plane based on network-level objectives, such as a reachability matrix, load-balancing goals, survivability requirements, and planned maintenance events. The algorithms in the decision plane may be customized based on knowledge of the network structure (e.g., simple path-computation algorithms for a ring topology). In one extreme design point, the decision plane may be a logically centralized component, with its functionality replicated by multiple Decision Elements (DEs) that connect directly to the network for fast, reliable communication with the routers and switches.

Dissemination plane: The dissemination plane provides a robust and efficient

communication substrate that connects routers/switches with decision elements. While control information may traverse the same set of physical links as the data packets, the dissemination paths are maintained separately from the data paths so they can be operational without requiring configuration or successful establishment of paths in the data plane. In contrast, in today's networks, control and management data are carried over the data paths, which need to be established by routing protocols before use. The dissemination plane moves management information created by the decision plane to the data plane and state identified by the discovery plane to the decision plane, but does not create state itself.

Discovery plane: The discovery plane is responsible for discovering the physical components in the network and creating logical identifiers to represent them. The discovery plane defines the scope and persistence of the identifiers, and carries out the automatic discovery and management of the relationships between them. This includes box-level discovery (e.g., what interfaces are on this router? How many FIB entries can it hold?), neighbor discovery (e.g., what other routers does this interface connect to?), and discovery of lower-layer link characteristics (e.g., what is the capacity of the interface?). The decision plane uses the information learned from the discovery plane to construct network-wide views. In contrast, in today's IP networks, the only automatic mechanism is neighbor discovery between two preconfigured and adjacent IP interfaces; physical device discovery and associations between entities are driven by configuration commands and external inventory databases.

Data plane: The data plane handles individual packets based on the state that is *output* by the decision plane. This state includes the forwarding table, packet filters, link-scheduling weights, and queue-management parameters, as well as tunnels and network address translation mappings. The data plane may also have fine-grain support for collecting measurements on behalf of the discovery plane.

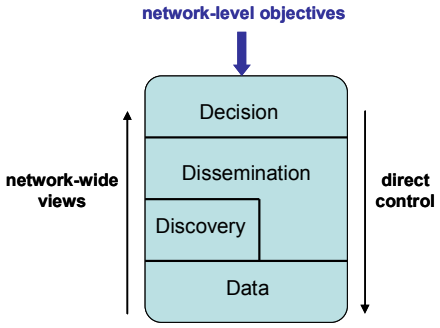

Figure 4: Four Planes of 4D Architecture with network level-objectives, network-wide views and direct control [Greenberg05a]

The results from an initial experimental study of the 4D architecture have confirmed the potential of the 4D design to achieve more robust and more efficient network-level decision making than currently possible. [Greenberg05c] The same study also shows that

a logically centralized decision plane, with decision functions and state replicated cross multiple DEs, is resilient in the event of link/router/DE failures or network partitions.

6. Conclusions

We have examined the Internet architecture from a historic perspective. Now it should be clear that the Internet is a success not purely because of economic forces; the network was built with a solid technology foundation which allows it to grow rapidly, foster innovations, and adapt to new application requirements. I hope you are also convinced that new thinking at the architecture level may be required in order to move forward and make the Internet become the integrated communications infrastructure of the future. This area of networking boasts a rich set of exciting research topics. For example, how to turn network configuration from an art, prone to human errors and cascading failures, into a science with precise abstractions and sound reasoning/checking frameworks? [Maltz04,Xie05] A research agenda specific to the 4D architecture is outlined in [Greenberg05b].

Acknowledgement

I'd like to thank my colleagues Peter Denning and John Gibson for vetting early drafts of this writing and providing helpful feedback. I'd also like to thank five anonymous reviewers whose insights have led to significant improvements in both content and presentation.

7. References

[Cerf74] V. Cerf, and R. Kahn, "A Protocol for Packet Network Intercommunication," *IEEE Trans. Communications,* Vol 22, No 5, May 1974.

[Clark88] D. Clark, "The Design Philosophy of the DARPA Internet Protocol," in *Proc. ACM SIGCOMM'88*, August 1988.

[Clark03] D. Clark, et al, "New Arch: Future Generation Internet Architecture," White Paper, 2003. Available from http://www.isi.edu/newarch/.

[Davies02] Davies, J., *Understanding IPv6*, Microsoft Press, 2002.

[Greenberg05a] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, J. Zhan, H. Zhang, "A Revolutionary 4D Approach to Network-Wide Control and Management." Research Proposal Submitted to NSF, January 2005.

[Greenberg05b] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, H. Zhang, "A Clean Slate 4D Approach to Network Control and

Management," *ACM Computer Communications Review*, vol. 35, num. 5, pp. 41-54, October 2005.

[Greenberg05c] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, H. Zhang, "Refactoring Network Control and Management: A Case for the 4D Architecture." Technical Report CMU-CS-05-177, Carnegie Mellon University, September 2005.

[Kurose05] Kurose, J.F., and Ross, K.W., *Computer Networking: A Top-down Approach Featuring the Internet,* 3rd ed., Addison-Wesley, 2005.

[Maltz04] D. Maltz, G. Xie, J. Zhan, H. Zhang, A. Greenberg, G. Hjalmtysson,"Routing Design in Operational Networks:A Look from the Inside", in *Proc. ACM SIGCOMM'04*, Portland, OR, August 2004.

[Rexford04] J. Rexford, A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, G. Xie, J. Zhan, H. Zhang, "Network-Wide Decision Making: Toward A Wafer-Thin Control Plane", in *Proc. ACM SIGCOMM HotNets'04*, San Diego, CA, November 2004.

[Saltzer84] J.H. Saltzer, D.P. Reed, and D.D. Clark, "End-to-End Arguments in System Design," *ACM Trans. Computer Systems*, vol. 2, num. 4, pp. 277-288, November 1984.

[Stallings04] Stallings, W., *Data and Computer Communications,*7th ed., Pearson Prentice Hall, 2004.

[WebSite1] *http://www.isoc.org/internet/history/*, "Internet Histories," last visited on March 28, 2006.

[WebSite2] *http://www.livinginternet.com*, "Living Internet," last updated on January 22, 2006.

[Xiao99] X. Xiao and L.M. Ni, "Internet QoS: A Big Picture," *IEEE Network*, pp 8-18, March/April 1999.

[Xie05] G. Xie, *et al*, "On Static Reachability Analysis of IP Networks," in *Proc. IEEE INFOCOM'2005 Conference*, Miami, FL, March 2005.

Glossary

Addressing – Means to uniquely identify source and destination of packets

Data Formatting – Means to convert application data into packets

Firewall – Hardware device or software program which establishes a security perimeter around a network

Forwarding – Means to move packets from input ports to output ports of a router

Internet Architecture – Communication services provided by the network, core building blocks for these services, and division of functionality among network elements

IP – Internetworking Protocol

Network Control and Management – All functionality deployed for managing resources in the data plane

Packet Switching – Communication paradigm in which large messages are partitioned into multiple, self-contained packets

Quality of Service – Level of performance guarantee provided by a network, measured by packet delay, loss rate, throughput, etc.

Resource Allocation – Techniques for dividing available network resources among traffic flows of different users

Routing – Means to determine a suitable path from a given a source to a given destination

Service Model – Abstraction of communication services provided by a network

TCP – Transmission Control Protocol, which provides reliable service

Traffic Engineering – Optimization of resource allocation for a given traffic matrix

Tunneling – Encapsulating a packet within another packet

UDP – User Datagram Protocol

Virtual Private Network – A private network where nodes are connected via encrypted tunnels built over the public Internet