Internet Gaming in the Era of IPv6

Robert W. Kerbs
Computer Science Department
California State Polytechnic University, Pomona
rwkerbs@csupomona.edu

Abstract

We have not seen an Internet-based game that can be identified with the arrival of Internet gaming for the masses. Due to the foundational structure of the Internet, there probably will not be one anytime soon. Why? The problem is that the underlying Internet Protocol, IPv4, has not had a major update since its introduction in the early 1970's. Fortunately, in the mid-1990's the standards-based Internet Engineering Task Force (IETF) began addressing IPv4's deficiencies. The result of these efforts is a new version of IPv4 called IPv6. IPv6's entry will present the Internet gaming community with many opportunities as well as challenges. One of biggest challenges is that the rollout of IPv6 will take years for completion necessitating its coexistence with IPv4 in a number of forms. This paper focuses on issues associated with this coexistence, transition mechanisms that will be used, and what the gaming community can do today to be better positioned as IPv6 becomes the mainstream Internet protocol.

1. Introduction

We have not seen an Internet-based game that can be identified with the arrival of Internet gaming for the masses. Due to initial layering of Internet-related protocols, there probably will not be one anytime soon. Why? It is because the implementation of the primary Internet Protocol (IPv4) has not had a major update since its inception in the early 1970's. Fortunately, in the mid-1990's the standards-based Internet Engineering Task Force (IETF) began addressing IPv4's deficiencies. Results of these efforts is a new version of IP called IPv6 (version IPv5 has essentially been skipped), however, rollout of IPv6 will take years for completion. This necessitates IPv6's coexistence with IPv4. As IPv6 is introduced it will become clear that it includes many desirable features and opportunities for the Internet gaming community. This paper focuses on the technical issues associated with IPv6, its coexistence with IPv4, its eventual dominance on the Internet, and some suggestions relating to what the gaming community can do today to prepare for this future.

1.1. Residential Broadband

Through broadband delivery of data to the home, it would seem that there would be great opportunity for users to experience a wide range of entertainment and information-based services. While many homes now have high-speed Digital Subscriber Lines (DSLs) or cable modems, users are often surprised to experience Internet latency and bandwidth problems (for a look at latency in hosts that use modems see [1]). What could be the bottleneck?

1.2. Internet Protocols

Out of the 100+ protocols that comprise the Internet's protocol suite, IPv4 and TCP were the first to be implemented in 1973 [2]. While TCP has been updated many times, IPv4 has not had a major update. IPv4's primary problem, besides running out of global address space, is that it was designed to deliver packets in a "best-effort" manner. Consequently, TCP has typically been relied upon for explicit delivery of packets at the transport layer. This reliable transmission comes at a cost however. For this reason, most games that possess tight timing constraints use UDP as the transport layer of choice – it, like IPv4, also delivers packets in a best-effort manner. Ultimately, TCP results in latency and UDP results in unreliable transmission of data.

IPv4 has had a number of extensions added on to it to deal with these issues. The problem is that they are not widely deployed on an end-to-end basis. Further, these additions require IPv4 to operate in a number of ways it was not intended - although some of the problems have been solved, they are applied as add-ons to IPv4 resulting in higher processing overhead. All of these issues are multiplied when trying to deliver a game to a user over the Internet in a timely fashion.

1.3. Performance Requirements in Internet Games

The Internet has traditionally been used for applications that possess mostly static data. If data arrives in ½ a second or 2 seconds it usually doesn't matter. The Internet is in a transition phase to a medium that requires quasi-real-time transmission of data. Consider a video game that operates at a relatively slow frame rate of 30 frames per second. This game would require transmission of a new frame every 33 ms (1/30). Typical round trip times (RTT) across the globe can easily be ten-times this much. It is not hard to see that if a frame arrives corrupted, or not at all, it is not possible to receive a replacement frame without interfering with the delivery of subsequent frames. Complicating the situation is the fact that for video game data each frame might contain important control information that could affect the outcome of the game. Where can improvements be made to alleviate this problem? Before we can attempt to answer this question it is necessary to understand how end hosts and routers handle excessive data delivery.

1.4. End Hosts and the Network Core

When data is sent across the Internet there are many possible points-of-failure. Let's consider two host computers that want to communicate across the Internet. These hosts will communicate by sending data in the form of packets. These packets will most likely traverse a number of networks on their way to each other's computers. Each network consists of a number of routers (and probably switches also). These routers look at host packets and forward them along a path through the network (the network core). Eventually the packets will exit the network/s and be delivered. Now, let's consider two common failures that occur in this scenario.

In the first case data is sent across a network through a specific router. That router, as in most computing devices, will contain some type of memory to buffer data (in this case packets) as they arrive. The router determines which hop the packet should be forwarded to next in an effort to get the packet to its final destination. If the router's buffer should fill up because too many packets are arriving on its input, it will simply drop (not accept) any more packets until it has some buffer space again. This type of problem is called congestion. One solution to this problem is to assign packets to certain Traffic Classes that possess a higher priority than packets assigned to other Traffic Classes. When the router becomes congested the lower priority packets are dropped before the higher priority packets. This type of solution is supported directly in IPv6.

In the second case data is sent from a very fast host, through the network, and finally to its final destination. In this case we assume we have all the bandwidth we need through the network core. The problem here is that if the recipient computer is slow and/or has small buffer space, it will not be able to keep up with the arriving traffic. Consequently, it too will be forced to drop packets. This type of problem is referred to as Flow Control. Flow control is usually managed through the use of a transport protocol like TCP. For the rest of this paper we keep in mind the first, more common, type of problem, congestion.

1.5. Perceptual Limitations

By taking into account player perceptual limitations developers can create an environment that results in an enjoyable game experience for the user. We know that the spectrum of multiplayer games that can be created encompasses many different genres. We also know that scene complexity can drastically affect rendering time. For a real-time strategy game Bettner and Terrano [3] found that latency under 250ms did not offend players, however, it was determined that players would not accept wide variations in latency. The authors found that constant latency was preferred over a game that operates very quickly at certain times and much slower at others. Consequently, it would be preferred to provide constant latency frames to an end host. IPv6 can provide constant latency by assigning packets to specific Traffic Classes – in effect they act as reservation mechanisms through the network which can be used for constant latency transmission. The study of sensation and perception is very useful for people who make games. A good resource is [4].

note - Bettner and Terrano's [3] study also suggests that constant clock game engines might better serve online gamers when compared with those that take advantage of extra clock cycles to perform certain functions.

2. IPv6

The primary motivating factor behind IPv6's introduction was the depletion of IPv4's 32-bit global address space. This issue was first looked at by the Internet Activities Board (IAB) (now known as the Internet Architecture Board) in 1991. Initial IPv6 proposals occurred in 1994 and finalized in 1995 [5] [6] - [6] is the most recent version of the specification. While IPv6's 128-bit address space would certainly deal with the depletion of global address space it was also envisioned that IPv6 would incorporate a number capabilities that IPv4 lacked [7] or is only supported through extensions – the idea being if IP was going to be updated anyway why not add those features from the start [8] ultimately decreasing the processing overhead associated with IP special features.

2.1. IPv4 Limitations

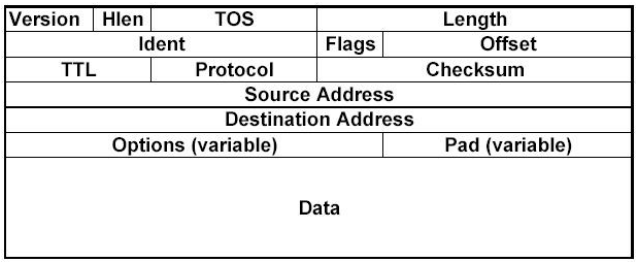
The original, un-extended, IPv4 suffered from a number of deficiencies. Nodes had to be manually configured in order to participate in a domain. There were no capabilities for multicast, security, and quality of service (QoS). Its header construction lacked support for future services. These issues have been dealt with through add-ons. Dynamic Host Control Protocol (DHCP) was introduced to automatically assign IP addresses to hosts. Multicasting techniques have evolved, security added with IPSec, and QoS is now supported with diffserv.

As a temporary fix for the depletion of global IP addresses, Network Address Translation (NAT) was introduced. NATs operate by assigning private addresses to nodes within an organization/home. When a node needs, or wants to, communicate with the outside world the node will obtain a global IPv4 address from the NAT and assign it to the request packet. This requires the source's header to be changed in real time. Consequently, an extra processing step is introduced into the communication process. In addition, using a NAT removes the end-to-end nature of communication that we would want between hosts (also destroying the end-to-end model of using IPSec security).

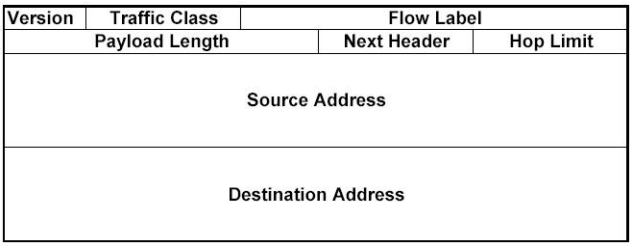
2.2. IPv6 Functionality and Construction

A number of enhancements over IPv4 have been included in IPv6. IPv4's 32-bit addresses have been replaced with 128-bit addresses. IPv6 provides automatic address assignment as well as embedded encryption and authentication capabilities to facilitate use of true end-to-end services. Support for tunneling, security, multicasting, and QoS have been built in from the start resulting in lower processing overhead. Further, IPv6 includes expansion abilities for future unforeseen requirements.

IPv6's header [6] is significantly different than IPv4's (Figure 1). Despite the fact that IPv4's header consists of 20B in its most basic form IPv6's header is 40B in its most basic form (extensions available) – which is amazing when consideration is given to the extra features IPv6 provides while simultaneously quadrupling its global address space. IPv6's header has removed the header Checksum utilized in IPv4 – offering better routing efficiency at little risk. A Flow Label field has been added for fast traversal of routing domains. IPv4's Time-to-Live (TTL) field has been replaced by IPv6's Hop Limit field. In addition IPv4's Type of Service (TOS) field has been replaced with IPv6's Traffic Class field – facilitating aggregation in multicasting. The IPv6 Length field refers to the number of bytes after its fixed length header. IPv6 extensions are added after the basic header. They efficiently accommodate services such as authentication, encryption,



(a) IPv4 header



(b) IPv6 header

Fig 1. Comparison of IPv4 and IPv6 headers. Notice the simplification of IPv6's header when compared with IPv4's.

and fragmentation. It is clear that IPv6's header construction has been carefully thought out and simplified.

3. Network Coexistence Strategies

The transition from IPv4 to IPv6 will be gradual – there will be no flag day were everyone will have to change over at once (unlike Sweden in 1967 when it changed from driving on the left-hand side of the road to the right-hand side of the road overnight). Not only will the two protocols have to coexist for a number of years they will need to interoperate. While there are many permutations of transition strategies [9], there are roughly three primary coexistence strategies including dual-stack, tunneling, and translators. In addition to these approaches, an important methodology that could be used in applications such as online multiplayer gaming could involve a technology called Multi-Protocol Label Switching (MPLS). It is hard to know which transition strategy will prevail; probably a mixture. Whichever technique is used it was necessary to require that IPv6 be backwards compatible to IPv4 – although we know that eventually IP will consist only of IPv6 nodes [10].

3.1. Dual-Stack

Hosts and networks will, and are, initially dealing with the transition by running dual protocol stacks on hosts and routers (see Figure 2). This approach allows for IPv6-only hosts to communicate with IPv4-only hosts and vice versa.

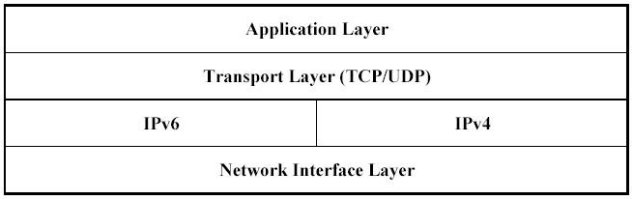


Fig 2. Dual-Stack hosts and routers possess complete IPv4 and IPv6 protocol stacks.

Dual-stack hardware uses the IPv4 stack when communicating with IPv4 hosts and routers and the IPv6 stack when communicating with IPv6 hosts and routers – the delineation of which stack to use is through the IPv4 and IPv6 header Version fields [11]. The dual-stack approach can also accommodate IPv6 addresses that are IPv4 compatible. Configuration of IPv4 addresses is either manually entered or through a standard DHCP server. Configuration of IPv6 addresses is either manually entered or through IPv6's auto-configuration mechanisms.

A major advantage of the dual-stack approach is that routers and hosts can handle both types of traffic. IPv4-mapped IPv6 addresses allow applications compiled with IPv6 system calls to communicate with IPv4-only applications. In addition, the dual-stack mechanism allows IPv4 hosts that haven't been updated to a dual-stack mode to operate over an IPv6 network core [11]. A downside to this approach is that routers must be configured to handle both protocols necessitating the use of additional resources.

3.2. Tunneling

Tunneling allows IPv6 end-hosts to communicate through an IPv4 network core (see Figure 3). The most common form of tunneling is IPv6 over IPv4 (6Over4). In this scheme IPv6 packets are encapsulated with IPv4 headers on one end of the network core, transmitted across IPv4 routers, and the final router (edge router), or end host, decodes the packet and forwards the IPv6 packet along its path to its final destination. This allows for non-IPv6-aware routers to transmit packets over an IPv4 network core, however, this does require the recipient router or host to be IPv6-capable [12]. Two major types of tunneling that will be used in the transition to IPv6 are configured tunneling and automatic tunneling.

3.2.1. Configured Tunneling

Configured tunneling uses manual configuration of the network infrastructure. One advantage of using this mechanism is that end hosts can use native IPv6 addresses. However, it will mostly likely be automatic tunneling that will be used primarily for IPv6.

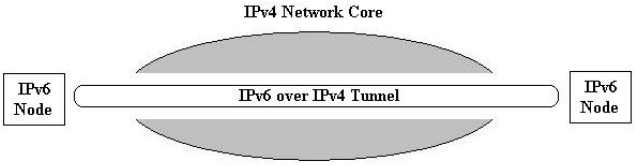


Fig 3. Tunneling allows IPv6 end hosts to communicate through an IPv4 network core.

3.2.2. Automatic Tunneling

Automatic tunneling requires that each endpoint possess an IPv4-compatible address. The advantage is that hosts are able to set up a tunnel without configuration. The IETF has identified three primary automatic tunneling approaches including 6to4, IPv4 compatible IPv6 addresses, and Intrasite Automatic Tunnel Addressing Protocol (IATAP).

6to4 automatic tunneling is the most common. It allows IPv6 domains to be mapped over IPv4 domains. IPv4-compatible IPv6 addresses can be used with IPv4 end hosts. In this situation the IPv4 address is embedded in the IPv6 address. The end host must be dual stack in this case. Finally ISATAP allows automatic tunneling to occur between IPv6 hosts operating in an IPv4 core network. Another approach that automatically sets up tunnels, and scales well, is Multi-Protocol Label Switching (MPLS).

3.4. Multi-Protocol Label Switching (MPLS)

In the scenarios outlined above packet routing is determined in real time. Consequently, these approaches take time to process requests and remove the possibility for pre-selected paths to be determined up front. This means a greater possibility for variance in latencies. On the other hand, MPLS utilizes an approach for quickly switching packets through the core of a network by carefully encoding packets to paths via the routers that sit on the edges of network cores running MPLS (see Figure 4).

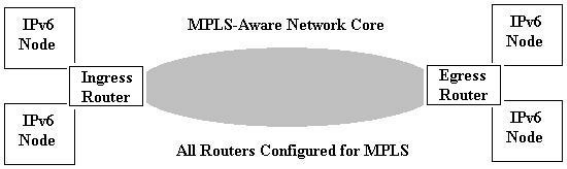


Fig 4. MPLS operates by encoding packets on the edge of a core network.

MPLS is a standards-based technology that came about in an effort to increase performance in the core of networks at the link-level over specific technologies like ATM or frame relay [13]. By allowing explicit paths (called Label Switched Paths (LSP)) to be set up through a network based upon specific entity numbers (called labels), next hop calculations do not need to be made in the core of the network. Packets arriving from a specific IP address, or range of addresses, are assigned specific labels - these are assigned by the ingress router. A forwarding table is consulted to determine what the next hop should be for the label and each packet is forwarded along its way without having to decode the IP header, read the destination IP address, and to forward the packet to the next hop. When the packet arrives at its next hop the arriving packet's label is analyzed, the current router's MPLS forwarding table is looked at, the corresponding label is selected and swapped with the packet's current label and the packet is sent along its way. This process continues until the egress router is reached where the label is stripped off, the packet's destination IP address is read and the packet is sent along its way using the real-time forwarding scheme used in standard packet forwarding.

The beauty of MPLS lies in its ability to distinguish different types of traffic arriving at the core of a network without having to read an IP header. Users that are identified as having a higher priority than other users are assigned labels giving them access to corresponding resources and traffic flow paths. This is important for the gaming community. One of the primary benefits of MPLS is that it is already implemented in many organizations that only have IPv4 network cores – the reason being is that a number of applications such as Virtual Private Networking (VPN) and QoS have been implemented with MPLS. IPv6 supports labels through its Flow Label field (see Figure 1).

3.3. Translators

Translators allow IPv6 and IPv4 domains to interoperate. More specifically, they allow IPv6 hosts to communicate with IPv4 hosts and vice versa. In this case a combination of dual-stack nodes and translation techniques must be used. In essence, translators use extensions to NAT techniques. The difference is that not only IP addresses are being translated but also address format (Figure 5).

4. IPv6 Testbed (6bone.net)

The IETF's Working Group on IPv6 Transition (NGTrans – Next GenerationTransition) oversees a worldwide IPv6 testbed. The 6bone was initially employed to test standards and implementations. Currently it is used for testing transition strategies and operational management issues. Originally designed as a

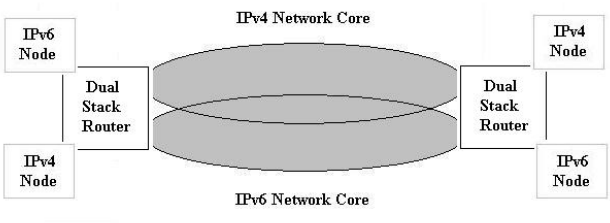


Fig 5. Translation requires the use of both dual-stack hosts and translation techniques.

virtual network (IPv6 over IPv4) to assist in the evolution and deployment of IPv6 [14], it has progressively been evolving to support native IPv6 transport links. It is interesting to note that the 6bone has developed into a worldwide collaboration of over 1200 sites – it will continue to be a major contributor to IPv6's successful rollout.

5. Analysis and Conclusion

As painful as the transition to IPv6 will be, it will usher in many opportunities for the gaming community. With time, end users will experience true IPv6 endpoint-to-endpoint communication at predictable, constant latencies. So, what can game developers do today? The question is a difficult one to answer for a number of reasons. For one, motivation for an Internet Service Provider (ISP) to upgrade to a full IPv6 backbone is small. IPv4 global addresses are still readily available and NAT can be used to stem off impending global address depletion. It might take market conditions to ultimately push certain ISPs into and through the transition.

The reality of the situation is that the core of the Internet is using IPv4 today. Even if an ISP is able to assure users that paths through their network core are indeed "IPv6-compatible," it is difficult to know how they are providing that compatibility and if it is even being provided in hardware or software – early router upgrades will certainly be through software and not hardware. Obviously, a software translation mechanism is going to affect frame-rate latency issues.

Most likely the transition will begin at the edge of an ISP's network core and work its way in from there. ISPs would simply make their edge routers dual-stack and in time convert their network core to dual-stack or construct a new IPv6 backbone.

Game companies hoping of making a successful online game that requires broadband bandwidth would be well advised to become intimately involved with the 6bone organization. By doing so, organizations will be tuned into IPv6 developments, transition status, as well as gain insight into what works and does not work in the IPv6 world. Through this education organizations can make

informed decisions and ultimately provide the end-user with a pleasant gaming experience.

References

[1] Jonathan Blow, “A Look at Latency in Networked Games”, *Game Developer*, vol. 5, no. 7, pp. 28-40, July, 1998.

[2] Edwin Diamond and Stephen Bates, "The Ancient History of the Internet," Wonders of Modern Technology (New York: American Heritage of Invention and Technology, 1997), pp. 12-19.

[3] Paul Bettner and Mark Terrano, “1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond”, *Proceedings of 2001 Game Developer Conference*, San Jose, CA, 2001.

[4] E. Bruce Goldstein, *Sensation and Perception*, Wadsworth, Pacific Grove, CA, 2002.

[5] David Morton, “Understanding IPv6”, *PC Network Advisor*, no. 83, pp. 17-22, 1997.

[6] Stephen Deering and Robert Hinden, "Internet Protocol, Version 6 (IPv6) Specification," Internet RFC 2460, December, 1998.

[7] IP Version 6 Working Group (ipv6), "IP Version 6 Working Group (ipv6) Charter," http://www.ietf.org/html.charters/ipv6-charter.html (current October 2002).

[8] Larry Peterson and Bruce Davie, *Computer Networks: A Systems Approach, 2nd Edition*. Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[9] Alain Baudot, Geir Egeland, Christian Hahn, Pasi Kyheroinen, and Andre’ Zehl, "Interaction of Transition Mechanisms," Internet-Draft, June, 2002.

[10] Erik Nordmark and Robert Gilligan, "Transition Mechanisms for IPv6 Hosts and Routers," Internet-Draft, July, 2002.

[11] Jim Bound, "Dual Stack Transition Mechanism (DSTM) Overview," Internet-Draft, June, 2002.

[12] Brian Carpenter and Cyndi Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," Internet RFC 2529, March, 1999.

[13] Multi-Protocol Label Switching Working Group (MPLS), "Multi-Protocol Label Switching Working Group (MPLS)," http://www.ietf.org/html.charters/mpls-charter.html (current September 2002).

[14] 6bone Testbed for Deployment of IPv6, "What is the 6bone?," http://6bone.net/about_6bone.html, (current November 2002).