5-2002

Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server

Wu-chang Feng
Oregon Graduate Institute of Science & Technology

Francis Chang
Oregon Graduate Institute of Science & Technology

Wu-chi Feng
Oregon Graduate Institute of Science & Technology

Jonathan Walpole
Oregon Graduate Institute of Science & Technology

Citation Details

"Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server," Wu-chang Feng, Francis Chang, Wu-Chi Feng, and Jonathan Walpole, OGI CSE Technical Report, CSE-02-005, May 15, 2002.

Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server

Wu-chang Feng Francis Chang Wu-chi Feng Jonathan Walpole

Department of Computer Science Engineering
OGI School of Science and Engineering at OHSU
Beaverton, OR 97006
{wuchang,francis,wuchi,walpole}@cse.ogi.edu

Abstract—**This paper describes the results of a 500 million packet trace of a popular on-line, multi-player, game server. The results show that the traffic behavior of this heavily loaded game server is highly predictable and can be attributed to the fact that current game designs target the saturation of the narrowest, last-mile link. Specifically, in order to maximize the interactivity of the game itself and to provide relatively uniform experiences between players playing over different network speeds, on-line games typically fix their usage requirements in such a way as to saturate the network link of their lowest speed players. While the traffic observed is highly predictable, the trace also indicates that these on-line games provide significant challenges to current network infrastructure. As a result of synchronous game logic requiring an extreme amount of interactivity, a close look at the trace reveals the presence of large, highly periodic, bursts of small packets. With such stringent demands on interactivity, routers must be designed with enough capacity to quickly route such bursts without delay. As current routers are designed for bulk data transfers with larger packets, a significant, concentrated deployment of on-line game servers will have the potential for overwhelming current networking equipment.**

This material is based upon work supported by the National Science Foundation under Grant EIA-0130344 and the generous donations of Intel Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or Intel.

I. Introduction

Due the confluence of advances in graphics hardware, in network bandwidth on the backbone and to the home, in client and server hardware, and in innovative software game design, on-line interactive games across all mediums has seen explosive growth. Forrester Research estimates that in 2001, there were 18 million on-line gamers in a 1.6 billion dollar industry. Recent measurement studies have shown that this is consistent with the observed growth in gaming traffic as Internet games now account for a growing fraction of aggregate load [1].

With the upcoming launches of Microsoft's Xbox on-line game network and Sony's Playstation 2 on-line game network, along with the development of emerging massively multi-player on-line games that allow for thousands of simultaneous players [2], it is clear that gaming traffic will soon grow to scales far beyond what is being observed today. While not indicative of all on-line games, the class of games known as "first-person shooters" has clearly dominated much of the observed gaming traffic. With a large list of popular games such as Doom, Quake, Descent, Duke Nukem 3D, Half-Life, Counter-Strike, Unreal Tournament, Tribes 2, Day of Defeat, Command & Conquer Renegade, etc., these games are representative of what on-line games of the future are moving towards: large-scale, highly interactive, virtual worlds [3], [4], [5].

With the recent global explosion of on-line multi-player gaming, it is becoming more important to understand its network behavior and usage in order to

provision and design future network infrastructure. By nature, traffic generated in support of this type of application will be much different than web or TCP-based traffic which has received most of the attention in the network research community [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. In particular, on-line gaming requires low-latency point-to-point communication as well as directed broadcast channels to facilitate its real-time game logic. In addition, such traffic tends to employ small, highly periodic, UDP packets. Packets are small since the application requires extremely low latencies which makes message aggregation impractical. Packets are highly periodic as a result of the game's dynamic requirement of frequent, predictable state updates amongst clients and servers. Finally, packets are typically sent via UDP since clients typically send packets at an interval that is much shorter than the time it would take to retransmit lost packets. In addition, the latency induced via socket buffers [16] and delayed acknowledgements is often too large to support meaningful interactivity.

In this paper, we take a closer look at a class of applications that look to become a major component in the traffic mix for the foreseeable future. Specifically, we study the behavior of an extremely busy, on-line game server over the course of a week. Section II describes the on-line game we have examined. Section III describes the results of a one-week trace of the game server. Section IV analyzes the implications that gaming traffic will have on network infrastructure of the future and Section V concludes.

II. Background

In order to understand the characteristics of Internet gaming, we examined the behavior of an extremely popular Counter-Strike server [5]. Counter-Strike is a modification to the popular Half-Life [4] game and has become one of the most popular and most network-intensive games played over the Internet as of this writing. Counter-Strike is a part of a large class of multi-player, on-line, first-person shooters that has dominated network gaming traffic over the last several years. As of May 2002, there were more than 20,000 Counter-Strike servers active [5]. The game is architected as a client-server application with multiple clients communicating and coordinating with a central server that keeps track of the global state of the game.

In the game itself, two teams continuously play back-to-back rounds of several minutes in duration. Each team attempts to complete their objectives and foil those of the other team during the round. The round ends when one team manages to complete their mission, when one team eliminates the other team entirely, or when time runs out. When players are eliminated from a round, they become spectators until the next round. During this time, the player can shadow another player that has not been eliminated. The game itself is played on a variety of maps which rotate based on how the server is configured. Typically, maps are rotated every 30 minutes, allowing for over 10 rounds to be played per map. Depending on the hardware and configuration, a Counter-Strike server can support up to 32 simultaneous players.

Traffic generated by the game can be attributed to a number of sources. The most dominant source is the real-time action and coordinate information sent back and forth between clients and server. This information is periodically sent from all of the clients to the server. The server then takes this information and performs a periodic broadcast to each client, effectively distributing the global state of the game. In addition to game physics datum, the game engine allows for broadcast text-messaging and broadcast voice communication amongst players all through the centralized server. The game server also supports the upload and download of customized logos that can be seen by everyone on a per-client basis. Each client is able to customize a texture map that may be placed on the surfaces of the current map. These images are uploaded and downloaded when users join the game and when a new map starts so that each client can properly display the custom decals of the other users. Finally, the game server supports downloads of entire maps, which may consist of sounds, texture libraries and a compiled Binary Space Partitioning tree [17]. In order to prevent the server from becoming overwhelmed by concurrent downloads, these downloads are rate-limited at the server.

Start Time	Thu Apr 11 08:55:04 2002
Stop Time	Thu Apr 18 14:56:21 2002
Total Time of Trace	7 d, 6 h, 1 m, 17.03 s (626,477 sec)
Maps Played	339
Established Connections	16030
Unique Clients Establishing	5886
Attempted Connections	24004
Unique Clients Attempting	8207

TABLE I
GENERAL TRACE INFORMATION

III. EVALUATION

A. Trace summary

To properly evaluate the traffic generated by this representative application, we hosted the Counter-Strike (version 1.3) server of one of the most popular on-line gaming communities in the Northwest region: Olygamer.com [18]. Because of the large Olygamer following, our exceptional Internet connectivity (OC-3 to Internet2, dual T3s to non-Internet2 sites), the speed of the server used (Dell Dimension 4300, Pentium 4, 1.8GHz, 512MB), and the superiority of our game configuration (which includes modules that eliminate cheating and deter team killing [19], [20], [21], [22], [23]), this server quickly became heavily utilized with connections arriving from all parts of the world irrespective of the time of day. The server itself was configured with a maximum capacity of 22 players. After a brief warm-up period, we recorded the traffic to and from the server over the course of a week (April 11-18). The trace collected consisted of a half billion packets. Note that while we are able to effectively analyze this single server, the results in this study do not directly apply to overall aggregate load behavior of the entire collection of Counter-Strike servers. In particular, it is expected that active user populations will not, in general, exhibit the predictability of the server studied in this paper and that the global usage pattern itself may exhibit a high degree of self-similarity [1], [24], [25].

Table I summarizes the trace itself. The trace covers over a week of continuous operation of the game server. Over 300 maps were played during this time

Total Packets	500,000,000
Total Packets In	273,846,081
Total Packets Out	226,153,919
Total Bytes	64.42 GB
Total Bytes In	24.92 GB
Total Bytes Out	39.49 GB
Mean Packet Load	798.11 pkts/sec
Mean Packet Load In	437.12 pkts/sec
Mean Packet Load Out	360.99 pkts/sec
Mean Bandwidth	883 kbs
Mean Bandwidth In	341 kbs
Mean Bandwidth Out	542 kbs

TABLE II
NETWORK USAGE INFORMATION

Total Bytes	37.41 GB
Total Bytes In	10.13 GB
Total Bytes Out	27.28 GB
Mean Packet Size	80.33 bytes
Mean Packet Size In	39.72 bytes
Mean Packet Size Out	129.51 bytes

TABLE III
APPLICATION INFORMATION

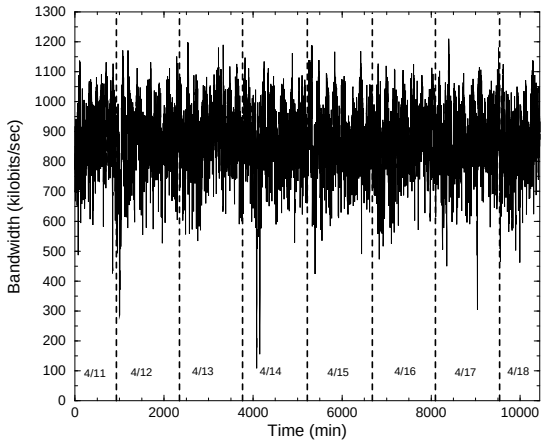


Fig. 1. Per-minute bandwidth of server for entire trace

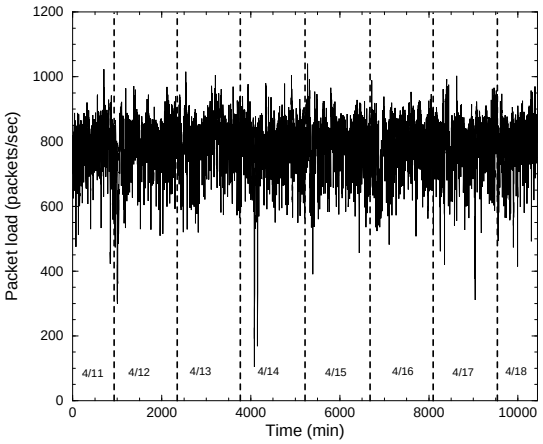


Fig. 2. Per-minute packet load of server for entire trace

frame and more than 16000 user sessions were established. Due to the popularity of the server, each user averaged almost 3 sessions for the week and more than 8000 connections were refused due to the lack of open slots on the server. In addition, each player was connected to the game an average of approximately 15 minutes. Table II summarizes the network load generated by the server. Over the 500 million packet trace, more than 60GB of data were sent including both network headers and application data. Overall, the bandwidth consumed approached $1Mbs$ for the duration of the trace. The table also shows that even though the application received more packets than it sent, its outgoing bandwidth exceeded its incoming bandwidth. Table III shows the reason why this was the case. The mean size of outgoing application data packets was more than three times the size of incoming application data packets.

To understand the dynamics of the trace, Figure 1 and Figure 2 plot the per-minute average bandwidth and packet load observed at the server. As the figures show, while there is a lot of short-term variation in the trace, the trace exhibits fairly predictable behavior over the long-term. Aggregate bandwidth consumed by the server hovers around 800-900 kilobits per second (kbs) while the server sees a packet rate of around 700-800 packets per second (pps). In addition, as Figure 3 shows, the number of active players also shows a lot of short-term variation with fairly predictable long-term behavior. The number

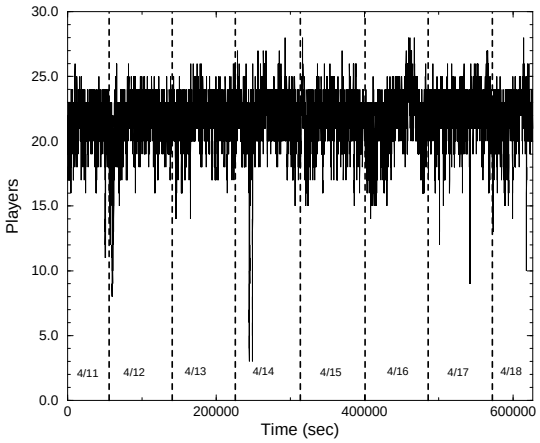


Fig. 3. Per-minute number of players for entire trace

of players sometimes exceeds the maximum number of slots of 22 as multiple clients can come and go during an interval. The trace itself also encompasses several brief network outages on April 12, April 14, and April 17. These outages caused minor disruptions in user numbers as the client and server disconnect after not hearing from each other over a period of several seconds. In all three cases, all of the players or a majority of players were disconnected from the server at identical points in time. While some of the players, having recorded the server's IP address, immediately reconnected, a significant number did not as they relied on dynamic server autodiscovery and auto-connecting to find this particu-

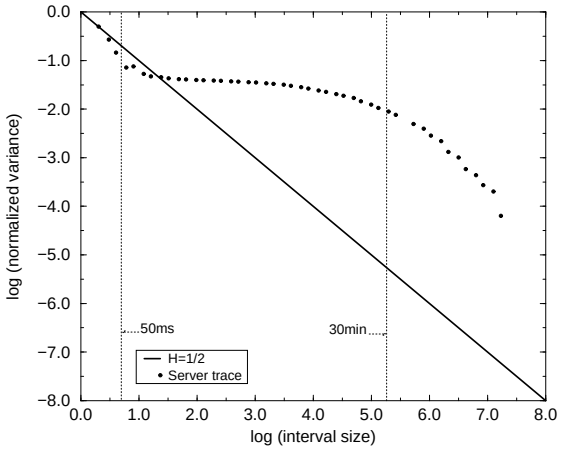


Fig. 5. Variance time plot for total server packet load

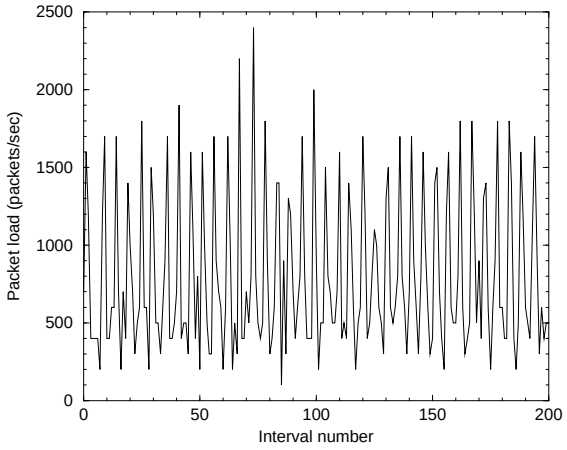


Fig. 6. Total packet load for the $m = 10ms$ interval size

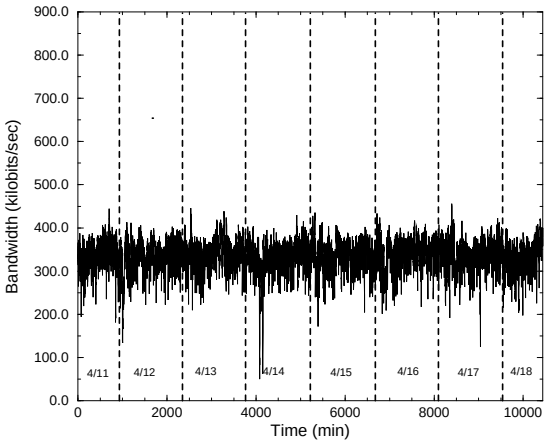
lar game server [26]. Thus, the user population and network traffic observed around these outages show significant dips on the order of minutes even though the actual outage was on the order of seconds. Due to the size of the trace, however, these brief outages did not have a significant impact on the traffic characterization.

Figure 4 shows a further breakdown of the trace between incoming traffic generated by clients and sent to the server and outgoing traffic generated by the server and sent to the clients. As the figure shows, the incoming packet load exceeds the outgoing packet load while the outgoing bandwidth exceeds the incoming bandwidth. Thus, it is clear that while the server sends less packets, the packets it sends are significantly larger than the incoming packets. This is not surprising as the game server itself is logically playing the role of a broadcaster: taking state information from each client and broadcasting it out to all other clients [27]. Section III-C gives a more detailed analysis of packet sizes in the trace.

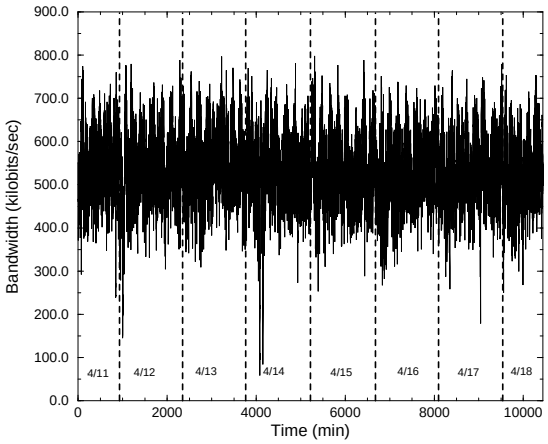
B. Periodicity and predictability

While visually it appears that the server's network load is relatively stable, the true measure of variability is its associated Hurst parameter [10], [28]. In order to measure variability across multiple time scales, we use the standard aggregated variance method to estimate the Hurst parameter (H) of the trace. In this method, the sequence is divided into multiple, consecutive, equally-sized, blocks. The values within the block are averaged and the variance of the sequence of averages is calculated. For a short-range dependent process, as the block size (m) is increased, the variance of the resulting sequence consistently decreases. In contrast, for long-range dependent sequences, the sequence maintains high variability across block sizes and time scales. To determine the degree of long-range dependence, the log of the normalized variance is plotted against the log of the block size. The normalized variance is calculated as the variance of the aggregated sequence divided by the variance of the initial, unaggregated sequence. The block size, in this case, is the number of frames per block. The Hurst parameter (H) can be estimated by taking the magnitude of the slope of the best-fit line through the data points (β) and calculating H via the relation $H = 1 - \frac{\beta}{2}$. The Hurst parameter thus ranges between $\frac{1}{2}$ and 1. A short-range or no dependence sequence will have a slope of -1 which corresponds to an H of $\frac{1}{2}$ while a long-range dependent sequence will have an H closer to 1.

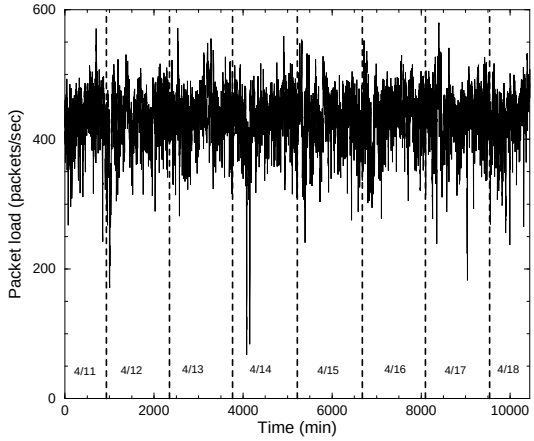
Figure 5 shows the variance-time plot of the trace. For this plot, a base interval size of $m = 10ms$ was used. The plot shows three distinct regions of behavior. For small m ($m < 50ms$), there is a high degree of increased smoothness as the interval size is increased, as shown by the slope of the



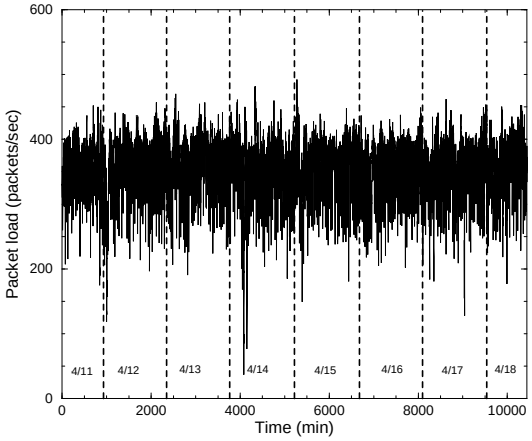
(a) Incoming bandwidth



(b) Outgoing bandwidth



(c) Incoming packet load



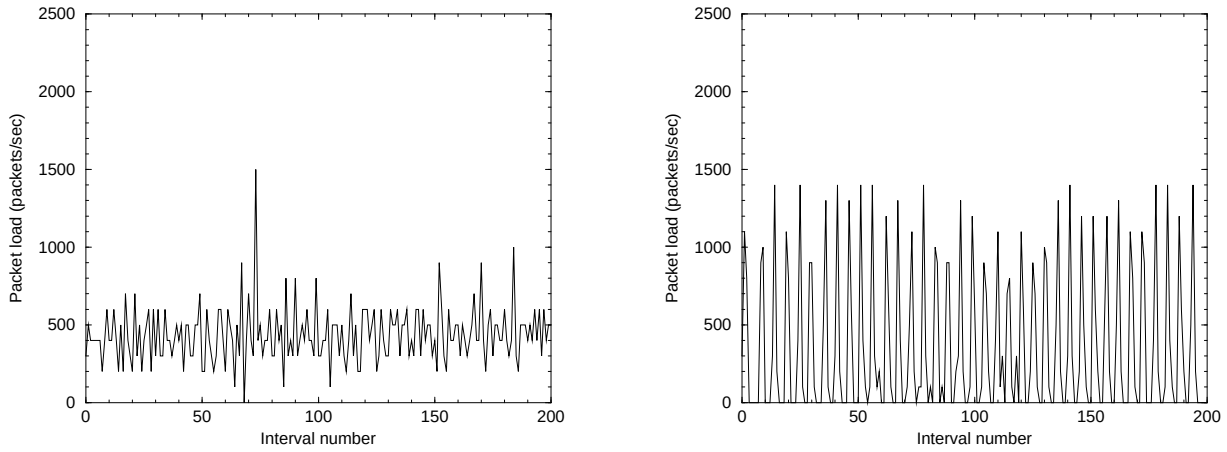
(d) Outgoing packet load

Fig. 4. Per-minute incoming and outgoing bandwidth and packet load of server for entire trace

variance plot as H drops below $\frac{1}{2}$. For larger interval sizes ($50ms < m < 30min$), significant variability and burstiness remains even as the interval size is increased. Finally, for large interval sizes ($m > 30min$), the plot shows typical behavior for a short-range dependent sequence as larger interval sizes consistently produce reduced variability with an estimated H of around $\frac{1}{2}$.

To fully understand the behavior at varying timescales, Figure 6 and Figure 7 plot the total packet load, the incoming packet load, and the outgoing packet load observed at the server for the first 200 $10ms$ intervals of the trace. The figure exhibits an extremely bursty, highly periodic pattern. When broken down between incoming and outgoing packet load, it is clear that the periodicity comes from the game server deterministically flooding its clients with state updates about every $50ms$. This synchronous behavior is due to the game server logic itself. Since clients connect via diverse network paths, the incoming packet load is not highly synchronized.

Given this behavior, Figure 8 shows the plot of the first 200 $50ms$ intervals of the trace. As expected, aggregating over this interval smooths out the packet load considerably. Between $m = 50ms$ and $m = 30min$, however, the variance-time plot



(a) Incoming packet load (b) Outgoing packet load

Fig. 7. Incoming and outgoing packet load for the $m = 10ms$ interval size

exhibits a high degree of variability. This variability can be attributed to the network disruptions caused by the $30min$ map time of the server. As the server loads a new map every half-hour, the network traffic dips significantly for a short period of time. Because most of the clients will have the maps stored locally, this down time is due completely to the server doing local tasks to perform the map change over. Figure 9 shows this behavior with a plot of the first 18000 $1sec$ intervals. Noticeable dips appear every 1800 ($30min$) intervals. Because map changing is a configuration-specific feature, this behavior is not a generic characteristic and can be affected by game administrators changing maps directly, players voting to change maps or extending the current map, or a different map time limit setting. For this trace and server configuration, increasing the interval size beyond the default map time of $30min$ removes the variability. Figure 10 plots the first 200 $30min$ intervals of the trace. As the figure shows, the variability has been eliminated.

The predictability of the aggregate leads us to examine how predictable the resource consumption of each individual flow is in the trace. Perhaps the most interesting observation is that when the mean bandwidth of the server is divided by the total number of players allowed by the game server itself (22), the bandwidth consumed per player is on average $40kbps$. This is no coincidence as the game is meant

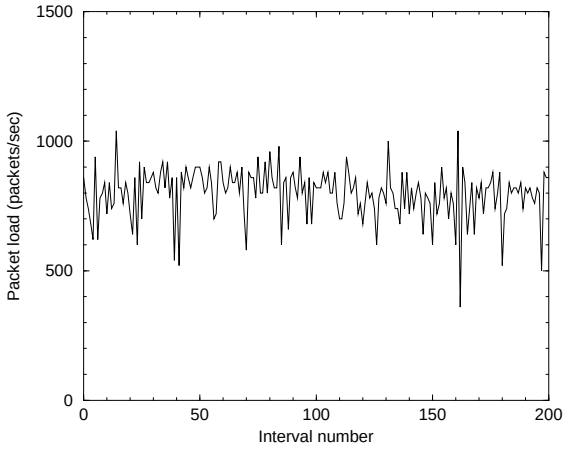


Fig. 8. Total packet load plot for $m = 50ms$

to be played uniformly across a wide range of network speeds, down to and including the ubiquitous $56kbps$ modem. As typical performance of $56kbps$ modems range from $40 - 50kbs$ [29], it is clear that this particular game was designed to *saturate the narrowest last-mile link*. Going back to the trace itself, we measured the mean bandwidth consumed by each flow at the server in order to get a picture of the bandwidth distribution across clients. Assuming minimal packet loss and a negligible difference in link-layer header overhead between the last-mile link and the server's link, the bandwidth measured at

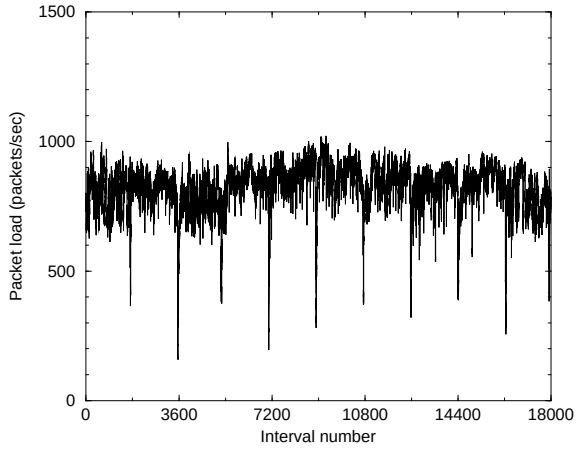


Fig. 9. Total packet load plot for $m = 1sec$

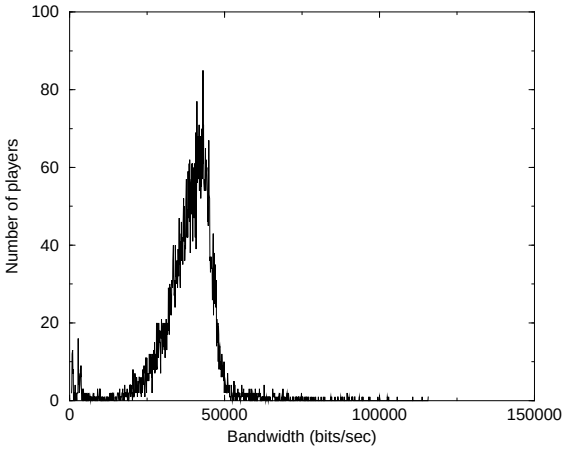


Fig. 11. Client bandwidth histogram

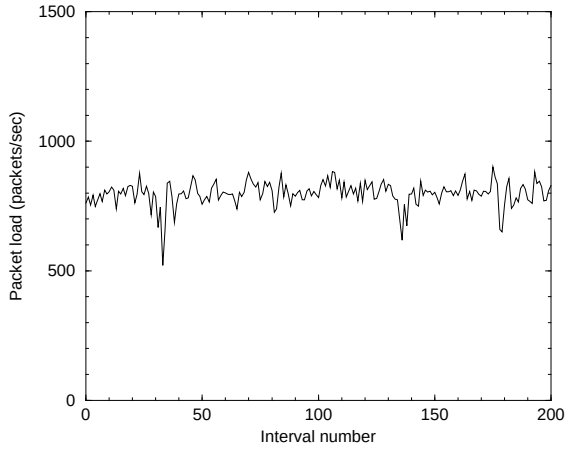


Fig. 10. Total packet load plot for $m = 30min$

the server will be quite close to what is sent across the last hop. Figure 11 shows a histogram of bandwidths across all sessions in the trace that lasted longer than $30sec$. The figure shows that the overwhelming majority of flows are pegged at modem rates or below even though connections arrived via diverse network mediums. The figure also shows that some flows do, in fact, exceed the $56kbps$ barrier. This is due to the fact that the client can be specially configured to crank up the update rate to and from the server in order to improve the interactivity of gameplay even futher. As shown by the histogram, only a handful of "*l337*" players connecting via high speed links have taken advantage of the setting.

C. Tiny packets

While the good news in the trace is that for a fixed set of players, the traffic generated is highly stable and predictable, the bad news is that the traffic itself is made up of large, periodic bursts of very small packets. Figure 12(a) plots the probability density function (PDF) of packet sizes across the 500 million packet trace. The graph is truncated at 500 bytes as only a negligible number of packets exceeded this. As the figure shows, almost all of the packets are under 200 bytes. Note that packet sizes include only the application data itself and not the UDP, IP, or Ethernet headers. Figure 12(b) shows the PDF of both incoming and outgoing packets. While incoming packets have an extremely narrow distribution centered around the mean size of 40 bytes, outgoing packets have a much wider distribution around a significantly larger mean. This causes the outgoing bandwidth to exceed the incoming bandwidth even though the rate of incoming packets exceeds that of outgoing packets. Figure 13 shows the cumulative distribution function (CDF) of the packet sizes. As the figure shows, almost all of the incoming packets are smaller than 60 bytes while a large fraction of outgoing packets have sizes spread between 0 and 300 bytes. This is significantly different than aggregate traffic seen within Internet exchange points [1] in which the mean packet size observed was above

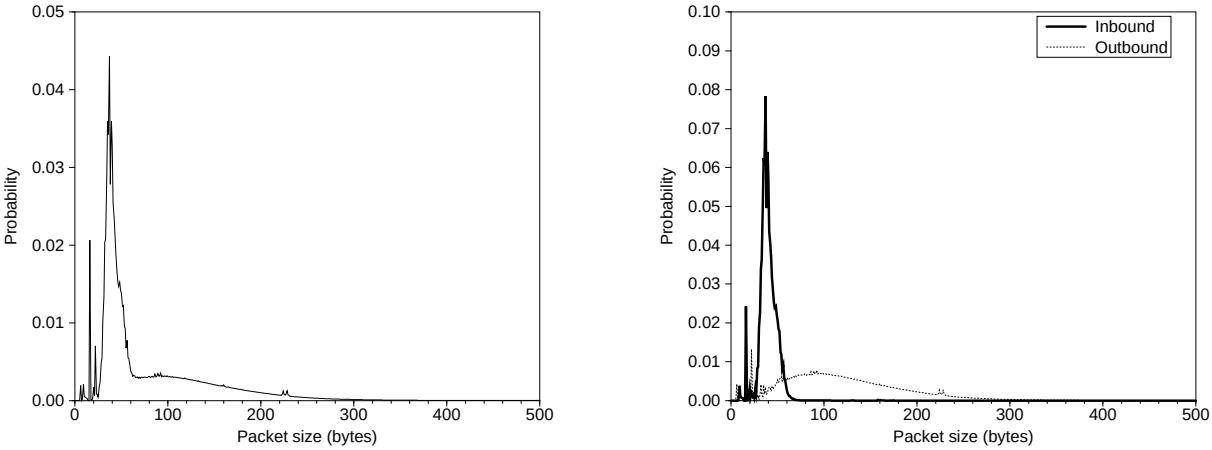


Fig. 12. Packet size probability density functions

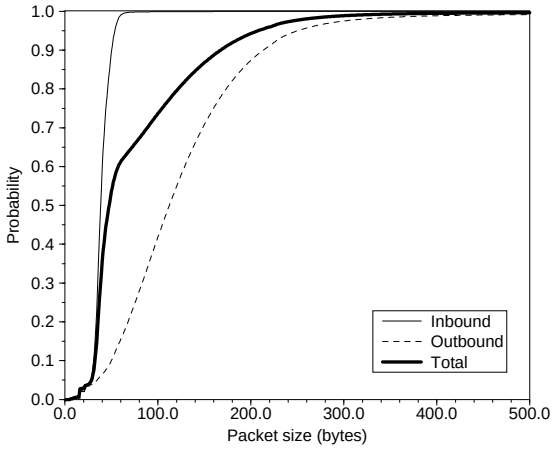


Fig. 13. Packet size cumulative density functions

Outgoing Traffic	
Total Packets From Server to NAT	677,278
Total Packets From NAT to Clients	674,157
Loss Rate	0.046%
Incoming Traffic	
Total Packets From Clients to NAT	853,035
Total Packets From NAT to Server	841,960
Loss Rate	1.3%

TABLE IV
NAT EXPERIMENT

400 bytes.

IV. Implications on Routing Infrastructure

A. The Bad News

Perhaps the most significant aspect of the trace is the observation that the game traffic itself consists of large, periodic bursts of short packets. While the trace is of only a single game over a single week, we believe that this is a characteristic that will be fundamental in all sufficiently loaded, highly interactive, on-line games due to the nature of the application and the underlying game logic. Short packets are required for low latency while highly periodic traffic allows the game to provide uniform interactivity amongst all of its clients. Some evidence of this exists in aggregate measures of other game applications [1].

Unfortunately for games, routers are not necessarily designed for this type of traffic. With the explosion of the web and peer-to-peer networks, the majority of traffic being carried in today's networks involve bulk data transfers using TCP. While TCP acknowledgements are typically small, data segments can be close to an order of magnitude larger than game traffic. With the addition of delayed acknowledgments and piggy-backed acknowledgments, the average packet sizes of most bi-directional TCP con-

nections will exceed those for games. Router designers and vendors often make packet size assumptions when building their gear, often expecting average sizes in between 1000 and 2000 bits (125-250 bytes) [30]. Thus, a significant shift in packet size from the deployment of on-line games will make the route lookup function the bottleneck versus the link speed [31]. Routing devices that are not designed to handle small packets will then see significant packet-loss or *even worse* consistent packet delay and delay jitter when handling game traffic.

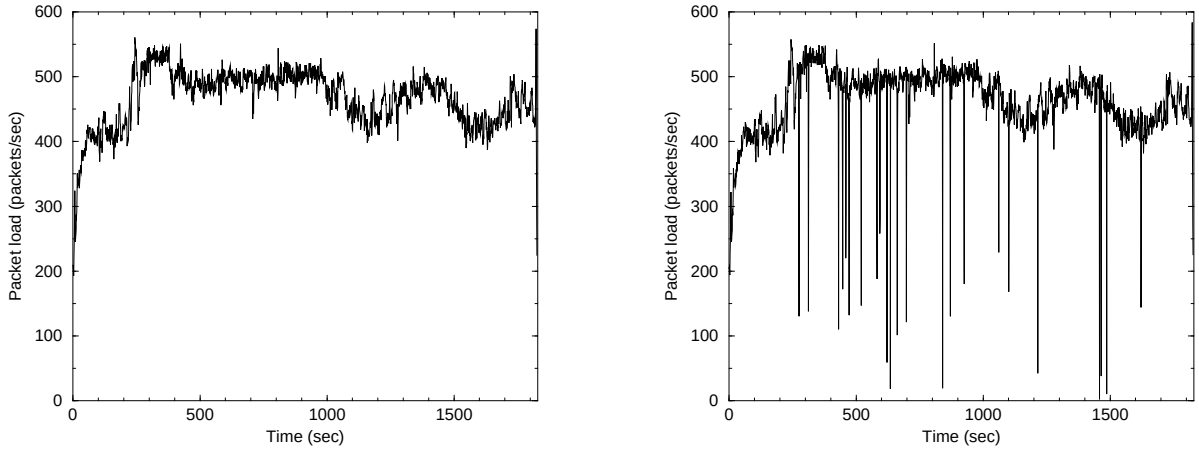
To demonstrate this problem, we inserted a commerical-off-the-shelf NAT device in between the server and the rest of the Internet. The specific device used was an SMC Barricade hardware NAT device (SMC7004AWBR). The SMC device has the capacity to switch at speeds of up to $100Mbs$ and has a listed packet routing capacity of 1000-1500 pps on average [32]. The wireless interface and all other physical ports were disabled so that the device only handled game traffic during the experiment.

After a brief warm-up period, we traced a single, $30min$ map of our server. Throughout the experiment, players complained about a significant degradation in performance with noticeable amounts of lag and packet loss that did not exist before the NAT device was in place. The results of the trace are shown in Table IV. The table shows that the NAT device does, in fact, induce packet loss with over 1% loss for incoming packets and almost 0.5% loss for outgoing packets. The losses occur at the most critical points during gameplay when a lot of events are occuring, magnifying the impact of the losses even futher. It is interesting that more incoming packets are lost versus outgoing packets as Figure 7 shows that the outgoing packet load is much more bursty. We believe that the loss on the incoming path is induced by and is a result of individual server packet bursts. Figure 14 and Figure 15 plot the packet load through the device. In this case, the incoming packet load from the clients to the NAT device is relatively stable while the packet load from the NAT device to the server sees frequent drop-outs. Interestingly enough, this disruption in service causes the game application itself to freeze as well with outgoing traffic from the server to the NAT device and outgoing traffic from the NAT device to the clients showing drop-outs directly correlated with lost incoming packets. While the loss rates overall were small, it was enough to substantially degrade the game quality. Due to the nature of the game applications, observed loss rates *self-tune* themselves at the worst tolerable level of performance. Any further degradation caused by additional players and/or background traffic will simply cause players to quit playing, reducing the load back to the tolerable level. Given the level of dissatisfaction caused by this experiment within the game, we believe the worst tolerable loss rate for this game is not far from 1-2%.

The NAT experiment shows that when placed behind a device like this, a single instance of the server running at about $800kbps$ is enough to overwhelm the resources of a device designed to route at significantly higher rates. For this application, adding buffers or combining packets does not necessarily help performance since delayed packets can be worse than dropped packets. Delayed packets degrade user experience [33] as well as consume additional processing resources at the server as it tries to keep its global state consistent. In this particular case, buffering the $50ms$ packet spikes will consume more than a quarter of the maximum tolerable latency for this type of interactive game. Without the routing capacity in place, hosting a successful game server behind such a device is simply not feasible. Extending this to on-line ventures from Microsoft and Sony and to massively multi-player games, it is apparent that even mid-range routers or firewalls within several hops of large hosted on-line game servers will need to be carefully provisioned to minimize both the loss and delay induced by routing extremely small packets. This will almost certainly require increasing the peak route lookup capacity of intermediate routers as adding buffers will add an unacceptable level of delay.

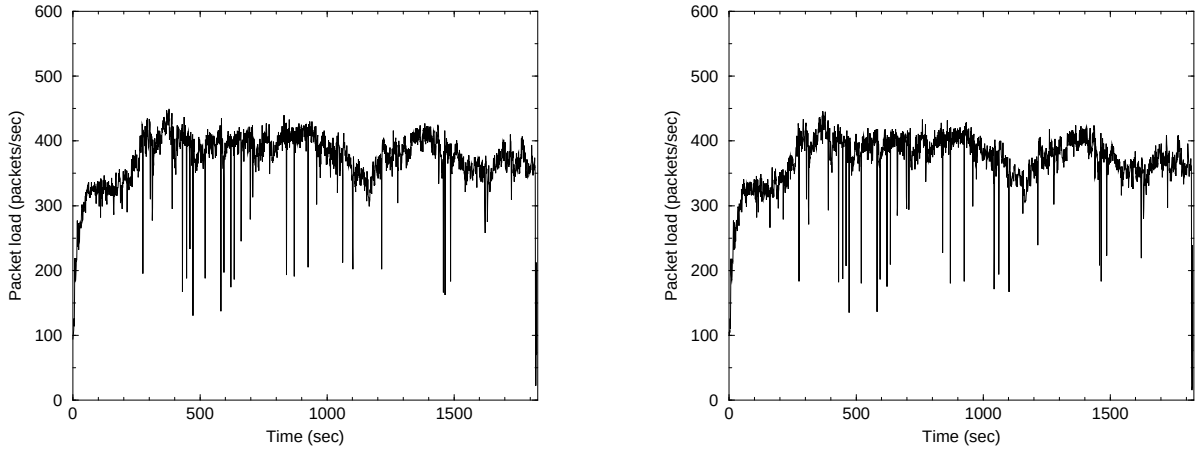
B. The Good News

The good news about the trace is that the predictability in resource requirements makes the modeling, simulation, and provisioning on-line gaming traffic a relatively simple task as the traffic does not exhibit fractal behavior when the number of active players is relatively fixed. As a result of this predictability, the traffic from an aggregation of all online Counter-Strike players is effectively linear to the number of active players. While this is the case, the



(a) Packet load from clients to NAT (b) Packet load from NAT to server

Fig. 14. Per-second incoming packet load for NAT experiment



(a) Packet load from server to NAT (b) Packet load from NAT to clients

Fig. 15. Per-second outgoing packet load for NAT experiment

actual number of players on-line over time may, in fact, exhibit a high degree of variability and self-similarity. Self-similarity in aggregate game traffic in this case will be directly dependent on the self-similarity of user populations [24], [25]. Since the trace itself can be used to more accurately develop source models for simulation [34], we hope to make the trace and associated game log file publicly available [35].

The other silver lining in this trace is that while the small packets of on-line games have the potential to wreak havoc on routing infrastructure, the periodicity and predictability of packet sizes allows for meaningful performance optimizations within routers. For example, preferential route caching strategies based on packet size or packet frequency may provide significant improvements in packet throughput. We hope to explore these issues further on a network processor testbed [36].

V. Conclusion

With the explosion in on-line, multi-player network games, it is becoming imperative to characterize this component of Internet traffic that will remain a sizable portion of overall usage. To this end, we have collected and analyzed a one-week, 500 million packet trace of a popular Internet game server. The resulting traffic consists of large, highly periodic bursts of small packets with predictable long-term rates and can be attributed to the synchronous nature of current game designs, the need to uniformly deliver a high degree of interactivity, and the phenomenon of *narrowest last-mile link saturation*. The bad news is that because of these factors, on-line game servers tend to generate heavy packet loads that may potentially overwhelm the route-lookup resources within the network. The good news is that, given a fixed number of players, these servers generate highly predictable traffic, thus simplifying the provisioning for and modeling of on-line games. In addition, the predictability and periodicity may be further leveraged to improve route caching strategies within the network.

As part of future work, we plan on using the results of this trace to explore the possibility of better route caching algorithms to support networks carrying this type of traffic. We also plan on examining a multitude of emerging on-line games. While we expect the trends to change very little as long as the last hop is dominated by slow modems, the gaming landscape tends to change much more quickly than any other class of Internet applications. Unlike web traffic whose aggregate application behavior is relatively well-characterized and changes slowly, we expect game traffic to be much more dynamic as games tend to zoom to popularity in a short period of time with new games replacing old games within very short time periods.

References

[1] S. McCreary and k. claffy, "Trends in Wide Area IP Traffic Patterns: A View from Ames Internet Exchange," in *Proceedings of 13th ITC Specialist Seminar on Measurement and Modeling of IP Traffic*, September 2000, pp. 1–11.

[2] GameSpy.com, "What's This World Coming To? The Future of Massively Multiplayer Games," `http://www.gamespy.com/gdc2002/mmog`.

[3] id Software, "id Software," `http://www.idsoftware.com/`.

[4] Half-Life, "The Official Half-Life Web Site," `http://half-life.sierra.com/`.

[5] Counter-Strike, "Counter-Strike: A Counter-Terrorism Half-Life Modification," `http://www.counter-strike.net/`.

[6] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," in *Proceedings of ACM SIGMETRICS*, July 1998, pp. 151–160.

[7] R. Caceres, N. Duffield, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B. Krishnamurthy, D. Lavelle, P. Mishra, K. Ramakrishnan, J. Rexford, F. True, and J. vanderMerwe, "Measurement and Analysis of IP Network Usage and Behavior," *IEEE Communication Magazine*, May 2000.

[8] M. Crovella and A. Bestavros, "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes," in *Proceedings of ACM SIGMETRICS*, May 1996.

[9] A. Feldmann, A. Gilbert, and W. Wilinger, "Data Networks as Cascades: Investigating the Multifractal Nature of Internet WAN Traffic," in *Proceedings of ACM SIGCOMM*, September 1998.

[10] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, February 1994.

[11] B. Mah, "An Empirical Model of HTTP Network Traffic," in *Proceedings of IEEE INFOCOM*, April 1997.

[12] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," in *Proceedings of ACM SIGCOMM*, August 1994, pp. 257–268.

[13] V. Paxson, "Empirically-derived Analytic Models of Wide-Area TCP Connections," *IEEE/ACM Transactions on Networking*, 1994.

[14] V. Paxson, "End-to-End Internet Packet Dynamics," in *Proceedings of ACM SIGCOMM*, September 1997.

[15] A. Veres and M. Boda, "The Chaotic Nature of TCP Congestion Control," in *Proceedings of IEEE INFOCOM*, April 2000.

[16] Ashvin Goel, Charles Krasic, Kang Li, and Jonathan Walpole, "Supporting Low Latency TCP-Based Media Streams," in *Proceedings of IWQoS*, May 2002.

[17] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On Visible Surface Generation by a Priori Tree Structures," *Computer Graphics(SIGGRAPH '80 Proceedings)*, vol. 14, pp. 124–133, July 1980.

[18] Olygamer.com, "Olygamer.com Gaming HQ," `http://www.olygamer.com/`.

[19] Half-Life Admin Mod Developers, "Half-Life Admin Mod Home," `http://www.adminmod.org/`.

[20] Cheating-Death Developers, "Cheating-Death Home," http://www.cheating-death.com/.

[21] CSCop Developers, "Project Info - CSCop for Counter-Strike," http://sourceforge.net/projects/cscop.

[22] CSGuard Developers, "CSGuard Home," http://www.olo.counter-strike.pl/.

[23] UDPSoft.com, "HLDS Ping Booster," http://www.udpsoft.com/.

[24] T. Henderson and S. Bhatti, "Modelling User Behavior in Networked Games," in *ACM Multimedia*, 2001, pp. 212–220.

[25] T. Henderson, "Latency and User Behaviour on a Multiplayer Game Server," in *Networked Group Communication*, 2001, pp. 1–13.

[26] T. Henderson, "Observations on Game Server Discovery Mechanisms," in *NetGames*, April 2002.

[27] D. LaPointe and J. Winslow, "Analyzing and Simulating Network Game Traffic," http://www.cs.wpi.edu/~claypool/mqp/net-game/game.pdf, December 2001.

[28] H.E. Hurst, "Long-Term Storage Capacity of Reservoirs," *Proc. American Society of Civil Engineers*, vol. 76, no. 11, 1950.

[29] J. Kristoff, "Understanding 56Kbps Modems," http://homepage.interaccess.com/~jkristof/56kmodem.html, 1997.

[30] C. Partridge, P. Carvey, E. Burgess, I. Castineyra, T. Clarke, L. Graham, M. Hathaway, P. Herman, A. King, S. Kohalmi, T. Ma, J. Mcallen, T. Mendez, W. Milliken, R. Pettyjohn, J. Rokosz, J. Seeger, M. Sollins, S. Storch, B. Tober, G. Troxel, D. Waitzman, and S. Winterble, "A 50-Gb/s IP Router," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, pp. 237–245, June 1998.

[31] Super Computer Gaming, "Super Computer Gaming: Jupiter Cluster," http://www.supercomputergaming.com/, 2002.

[32] SMC FAQ, "SMC Barricade FAQ: How fast is the Barricade? What is the throughput?," http://www.smc-europe.com/english/support/faq.html.

[33] I. S. MacKenzie and S. Ware, "Lag as a Determinant of Human Performance in Interactive Systems," *Proceedings of the ACM Conference on Human Factors in Computing Systems - INTERCHI*, pp. 488–493, 1993.

[34] M. Borella, "Source Models of Network Game Traffic," *Computer Communications*, vol. 23, no. 4, pp. 403–410, February 2000.

[35] W. Feng, "cs.mshmro.com Counter-Strike Server Traffic Trace 4/11-4/18," 40GB trace available upon request, April 2002.

[36] Intel, Inc., "Intel IXP 1200," http://developer.intel.com/design/network/products/npfamily/ixp1200.ht%m.