

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

Wu-chang Feng Francis Chang Wu-chi Feng Jonathan Walpole  
OGI School of Science and Engineering at OHSU

*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 and to provide relatively uniform experiences between all players, 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. Due to 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.

## I. INTRODUCTION

Due to 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. With the upcoming launches of Microsoft's Xbox and Sony's Playstation 2 on-line game networks and with the emergence of massively multi-player on-line games [1], it is clear that a large increase in gaming traffic is imminent. While not indicative of all on-line games, the class of games known as "first-person shooters" has dominated much of today's gaming traffic [2]. With a large list of popular games such as Doom, Quake, Half-Life, Counter-Strike, Unreal Tournament, Day of Defeat, Medal of Honor, Command & Conquer Renegade, etc., these games are representative

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

of what on-line games of the future are moving towards: large-scale, highly interactive, virtual worlds [3]. By nature, traffic generated in support of this type of application is completely different than web or TCP-based traffic which has received most of the attention in the network research community [4], [5], [6], [7]. 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 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 [8] 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 in the future.

## II. BACKGROUND

In order to understand the characteristics of Internet gaming, we examined the behavior of an extremely popular Counter-Strike server [3]. Counter-Strike is a modification to the popular Half-Life 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 [3]. 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 game state. 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

Start Time	Thu Apr 11 08:55:04 2002
Stop Time	Thu Apr 18 14:56:21 2002
Total Time	7 d, 6 h, 1 m, 17.03 s
Maps Played	339
Established Connections	16030
Unique Clients Established	5886
Attempted Connections	24004
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 pps
Mean Packet Load In	437.12 pps
Mean Packet Load Out	360.99 pps
Mean Bandwidth	883 kbs
Mean Bandwidth In	341 kbs
Mean Bandwidth Out	542 kbs
Mean Packet Size	80.33 bytes
Mean Packet Size In	39.72 bytes
Mean Packet Size Out	129.51 bytes

TABLE I  
TRACE INFORMATION

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 [9]. In order to prevent the server from becoming overwhelmed by concurrent downloads, these downloads are rate-limited at the server.

### III. EVALUATION

#### A. Trace summary

To properly evaluate the traffic generated by this representative application, we hosted a shared Counter-Strike (version 1.3) server for two of the most popular on-line gaming communities in the Northwest region: olygamer.com and mshmro.com [10], [11]. Because of the large followings of these communities, our excep-

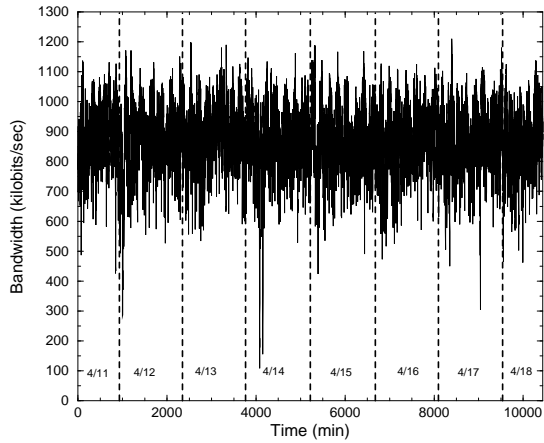
tional Internet connectivity, the speed of the server used, and the superiority of our game configuration (which includes modules that eliminate cheating and deter team killing [12]), this server quickly became heavily utilized with connections arriving from all parts of the world irrespective of the 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 [2], [13], [14].

Table III-A 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 frame and more than 16000 user sessions were established. Due to the popularity of the server, more than 8000 connections were refused due to the lack of open slots on the server and each player was connected to the game an average of approximately 15 minutes. In addition, over the 500 million packet trace, more than 60 GB 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. This was because the mean size of outgoing application data packets was more than three times the size of incoming application data packets.

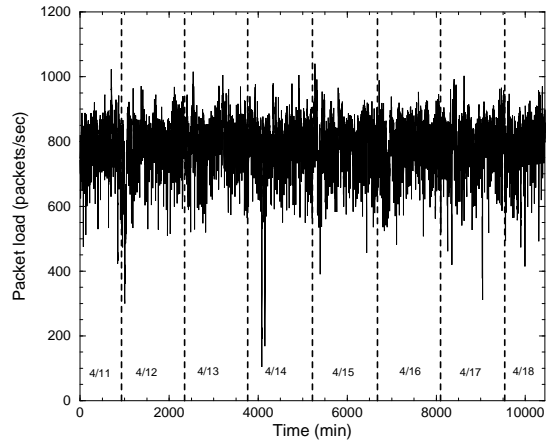
To understand the dynamics of the trace, Figure 1 plots the per-minute average bandwidth and packet load observed at the server. As the figure shows, while there is a lot of short-term variation in the trace, the trace exhibits fairly predictable behavior over the long-term. Aggregate bandwidth consumed by the server hovers around 900 kilobits per second (*kbs*) while the server sees a packet rate of around 800 packets per second (*pps*). In addition, throughout the experiment, the number of active players also remained at or near the capacity of 22. The trace also encompasses several brief network outages that did not significantly impact the analysis of the results [12].

#### 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 [6], [15]. In order to measure variability across multiple time scales, we use the stan-



(a) Total bandwidth



(b) Packet load

Fig. 1. Per-minute network load of server for entire trace

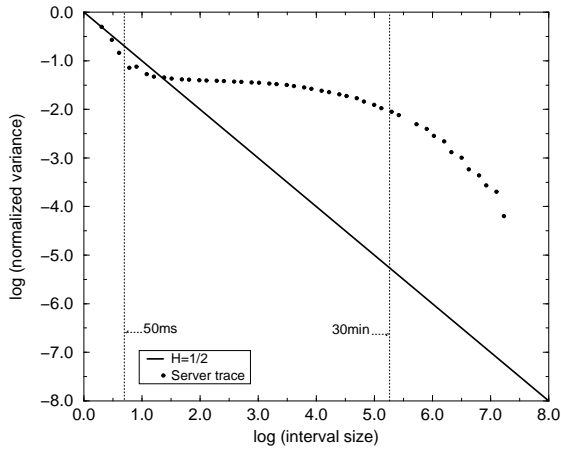


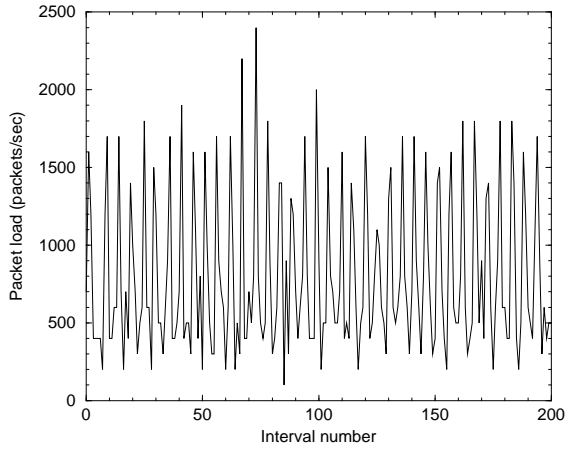
Fig. 2. Variance time plot for total server packet load

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

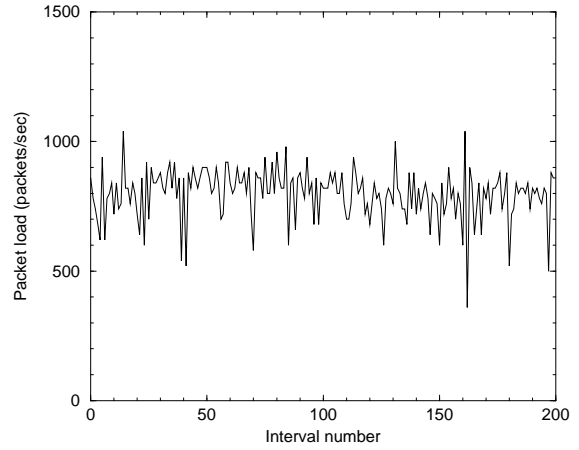
the number of frames per block. The Hurst parameter ( $H$ ) can be estimated by taking the magnitude of the slope of the best-fit line through the data points ( $\beta$ ) and calculating  $H$  via the relation  $H = 1 - \frac{\beta}{2}$ . The Hurst parameter thus ranges between  $\frac{1}{2}$  and 1. 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 2 shows the variance-time plot of the trace. For this plot, the interval size that is plotted is the normalized interval size using a base interval of  $m = 10ms$ . 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 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  $\frac{1}{2}$ .

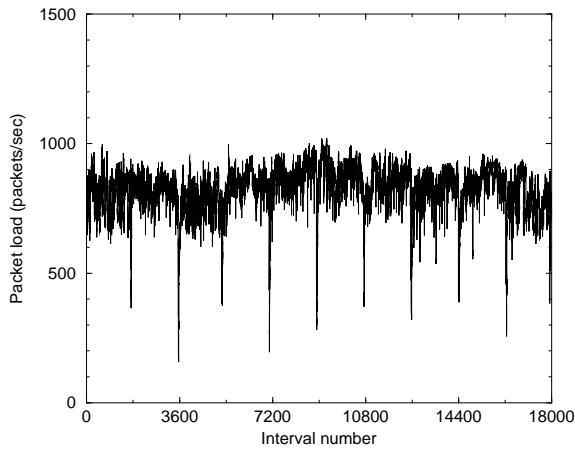
To fully understand the behavior at varying time-scales, Figure 3 plots the total packet load observed at the server averaged over a range of time intervals. Figure 3(a) plots the first 200  $10ms$  intervals of the trace. The figure exhibits an extremely bursty, highly periodic pattern which is the result of the synchronous operation of the game server logic itself which is written to deterministically flood its clients with state updates about every  $50ms$  [12]. Given this behavior, Figure 3(b) shows the plot of the first 200  $50ms$  intervals of the trace. As expected, aggregating over this interval smooths out the packet load con-



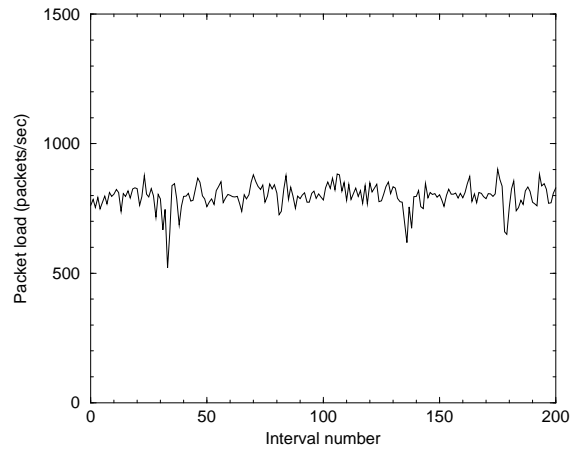
(a)  $m = 10ms$



(b)  $m = 50ms$



(c)  $m = 1sec$



(d)  $m = 30min$

Fig. 3. Total packet load over a range of interval sizes

siderably. Between  $m = 50ms$  and  $m = 30min$ , however, the variance-time plot 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 3(c) 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 3(d) 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 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  $40 - 50kbs$  [16], 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

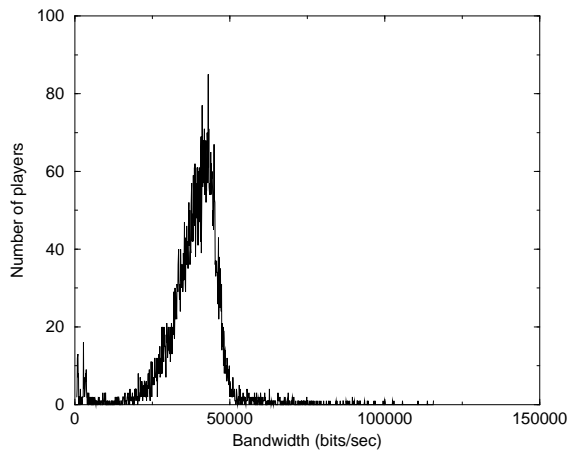


Fig. 4. Client bandwidth histogram

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 the server will be quite close to what is sent across the last hop. Figure 4 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 further. As shown by the histogram, only a handful of elite ("1337") players connecting via high speed links have taken advantage of the setting.

### C. Tiny packets

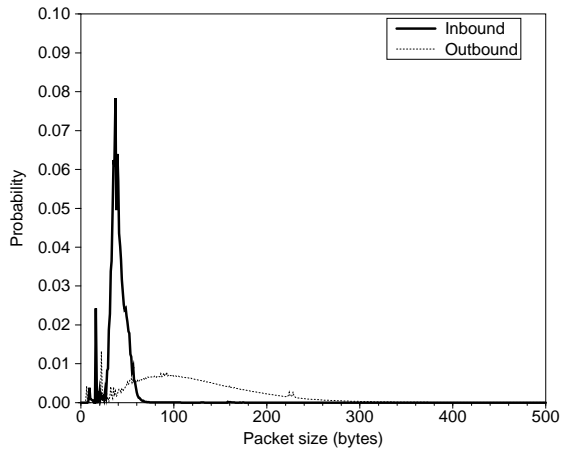
While the good news in the trace is that for a fixed set of players, the traffic generated is highly stable and predictable, the bad news is that the traffic itself is made up of large, periodic bursts of very small packets. Figure 5(a) shows the PDF of both incoming and outgoing packets. While incoming 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. 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 [17]. Figure 5(b) shows the cumulative distribution function (CDF) of the packet sizes. As the fig-

ure 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 [2] in which the mean packet size observed was above 400 bytes.

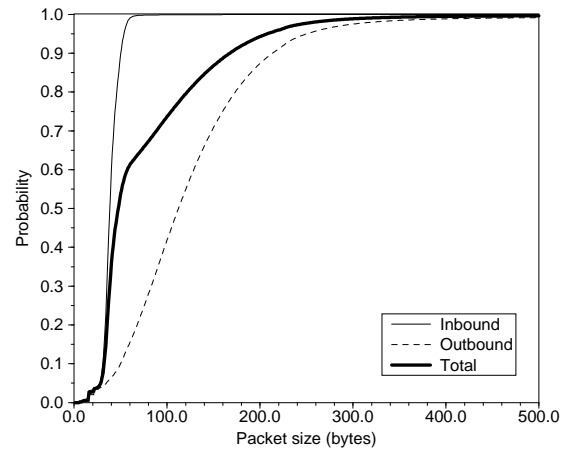
## IV. IMPLICATIONS ON ROUTING INFRASTRUCTURE

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 [2]. 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 larger-sized TCP segments. 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) [18]. 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 [19]. 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 [20]. Anecdotal evidence using a commercial-off-the-shelf NAT device corroborates this [12]. 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.

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 on-line Counter-Strike players is effec-



(a) PDF



(b) CDF

Fig. 5. Packet size distributions of trace

tively linear to the number of active players. While this is the case, the 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 [13], [14]. Since the trace itself can be used to more accurately develop source models for simulation [21], we hope to make the trace and associated game log file publicly available [22]. 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 [23].

#### REFERENCES

- [1] GameSpy.com, "What's This World Coming To? The Future of Massively Multiplayer Games," <http://www.gamespy.com/gdc2002/mmog>.
- [2] 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.
- [3] Counter-Strike, "Counter-Strike: A Counter-Terrorism Half-Life Modification," <http://www.counter-strike.net/>.
- [4] 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.
- [5] A. Feldmann, A. Gilbert, and W. Willinger, "Data Networks as Cascades: Investigating the Multifractal Nature of Internet WAN Traffic," in *Proceedings of ACM SIGCOMM*, September 1998.
- [6] 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.
- [7] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," in *Proceedings of ACM SIGCOMM*, August 1994, pp. 257–268.
- [8] A. Goel, C. Krasic, K. Li, and J. Walpole, "Supporting Low Latency TCP-Based Media Streams," in *Proceedings of IWQoS*, May 2002.
- [9] 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.
- [10] olygamer.com, "http://www.olygamer.com/".
- [11] mshmro.com, "http://www.mshmro.com/".
- [12] W. Feng, F. Chang, W. Feng, and J. Walpole, "Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server," in *OHSU TR CSE-02-005*, May 2002.
- [13] T. Henderson and S. Bhatti, "Modelling User Behavior in Networked Games," in *ACM Multimedia*, 2001, pp. 212–220.
- [14] T. Henderson, "Latency and User Behaviour on a Multiplayer Game Server," in *Networked Group Communication*, 2001, pp. 1–13.
- [15] H.E. Hurst, "Long-Term Storage Capacity of Reservoirs," *Proc. American Society of Civil Engineers*, vol. 76, no. 11, 1950.
- [16] J. Kristoff, "Understanding 56Kbps Modems," <http://homepage.interaccess.com/~jkristof/56kmodem.html>, 1997.
- [17] D. LaPointe and J. Winslow, "Analyzing and Simulating Network Game Traffic," <http://www.cs.wpi.edu/~claypool/mqp/net-game/game.pdf>, December 2001.
- [18] C. Partridge et. al, "A 50-Gb/s IP Router," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, pp. 237–245, June 1998.
- [19] Super Computer Gaming, "Super Computer Gaming: Jupiter Cluster," <http://www.supercomputergaming.com/>, 2002.
- [20] 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.
- [21] M. Borella, "Source Models of Network Game Traffic," *Computer Communications*, vol. 23, no. 4, pp. 403–410, February 2000.
- [22] W. Feng, "cs.mshmro.com Counter-Strike Server Traffic Trace 4/11-4/18," 40GB trace available upon request, April 2002.
- [23] E. Johnson and A. Kunze, "IXP1200 Programming," 2002.