




Centrum voor Wiskunde en Informatica

View metadata, citation and similar papers at core.ac.uk

brought to you by  CORE

provided by CWI's Instituut

REPORTRAPPORT

PNA

Probability, Networks and Algorithms



Probability, Networks and Algorithms

Modeling Ping times in First Person Shooter games

N. Degrande, D. De Vleeschauwer, R.E. Kooij,
M.R.H. Mandjes

REPORT PNA-R0608 JUNE 2006

Centrum voor Wiskunde en Informatica (CWI) is the national research institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organisation for Scientific Research (NWO). CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

Copyright © 2006, Stichting Centrum voor Wiskunde en Informatica
P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3711

Modeling Ping times in First Person Shooter games

ABSTRACT

In First Person Shooter (FPS) games the Round Trip Time (RTT), i.e., the sum of the network delay from client to server and the network delay from server to client, impacts the gamer's performance considerably. Game client software usually has a built-in process to measure this RTT (also referred to as Ping time), and as such gamers do not want to connect to servers with a large Ping time. This paper develops a methodology to evaluate the Ping time in a scenario where gamers access a common gaming server over an access network, consisting of a link per user that connects this user to a shared aggregation node that in turn is connected to the gaming server via a bottleneck link. First, a model for the traffic the users and the server generate, is proposed based on experimental results of previous papers. It turns out that the characteristics of the (downstream) traffic from server to clients differ substantially from the characteristics of the client-to-server (upstream) traffic. Then, two queuing models are developed (one for the upstream and one for the downstream direction) and combined such that a quantile of the RTT can be calculated given all traffic and network parameters (packet sizes, packet inter-arrival times, link rate, network load, ...). This methodology is subsequently used to assess the (quantile of the) RTT in a typical Digital Subscriber Line (DSL) access scenario. In particular, given the capacity dedicated to gaming traffic on the bottleneck link (between the aggregation node and gaming server), the number of gamers (or equivalently the gaming load the bottleneck link can support) is determined under the restriction that the quantile of the RTT should not exceed a predefined bound. It turns out that this tolerable load is surprisingly low in most circumstances. Finally, it is remarked that this conclusion depends to some extent on the details of the downstream traffic characteristics and that measurements reported in literature do not give conclusive evidence on the exact value of all parameters, such that, although the qualitative conclusions still remain valid, additional experiments could refine the detailed quantitative results.

2000 Mathematics Subject Classification: 60K25, 90B18

Keywords and Phrases: gaming -- packet delay -- quantiles

Modeling Ping times in First Person Shooter games

N. Degrande¹, D. De Vleeschauwer^{1,2}, R.E. Kooij^{3,4}, M.R.H. Mandjes⁵

¹ Alcatel Bell, Network Strategy Group
Francis Wellesplein1, B-2018 Antwerp, Belgium
{natalie.degrande|danny.de_vleeschauwer}@alcatel.be

² University Ghent, TELIN,
Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium

³ TNO Information and Communication Technology,
Brassersplein 2, Postbus 5050, 2600 GB Delft, the Netherlands
r.e.kooij@telecom.tno.nl

⁴ Delft University of Technology,
Mekelweg 4, 2628 CD Delft, the Netherlands

⁵ CWI,
P.O. Box 94079, 1090 GB Amsterdam, the Netherlands
michel.mandjes@cwil.nl

Abstract

In First Person Shooter (FPS) games the Round Trip Time (RTT), i.e., the sum of the network delay from client to server and the network delay from server to client, impacts the gamer's performance considerably. Game client software usually has a built-in process to measure this RTT (also referred to as Ping time), and as such gamers do not want to connect to servers with a large Ping time. This paper develops a methodology to evaluate the Ping time in a scenario where gamers access a common gaming server over an access network, consisting of a link per user that connects this user to a shared aggregation node that in turn is connected to the gaming server via a bottleneck link. First, a model for the traffic the users and the server generate, is proposed based on experimental results of previous papers. It turns out that the characteristics of the (downstream) traffic from server to clients differ substantially from the characteristics of the client-to-server (upstream) traffic. Then, two queuing models are developed (one for the upstream and one for the downstream direction) and combined such that a quantile of the RTT can be calculated given all traffic and network parameters (packet sizes, packet inter-arrival times, link rate, network load, ...). This methodology is subsequently used to assess the (quantile of the) RTT in a typical Digital Subscriber Line (DSL) access scenario. In particular, given the capacity dedicated to gaming traffic on the bottleneck link (between the aggregation node and gaming server), the number of gamers (or equivalently the gaming load the bottleneck link can support) is determined under the restriction that the quantile of the RTT should not exceed a predefined bound. It turns out that this tolerable load is surprisingly low in most circumstances. Finally, it is remarked that this conclusion depends to some extent on the details of the downstream traffic characteristics and that measurements reported in literature do not give conclusive evidence on the exact value of all parameters, such that, although the qualitative conclusions still remain valid, additional experiments could refine the detailed quantitative results.

1. Introduction

Since a few years the Internet Protocol (IP) is considered to be *the* enabling technology for multi-service networks. As a result more and more interactive services (such as telephony, videphony, and networked games) compete with the traditional elastic services (controlled by the Transport Control Protocol (TCP)).

To meet the delay (and packet loss) requirements of the interactive services, the traffic associated with these has to be (virtually) segregated from the elastic traffic. For that purpose the Internet Engineering Task Force (IETF) has defined two approaches to support Quality of Service (QoS) in IP networks: the Integrated Services model [5] (IntServ) and the Differentiated Services [4] (DiffServ) model. Both rely on some sort of prioritization of packets associated with interactive applications; these get a priority treatment via schedulers in the nodes of the network.

In traditional *First-In-First-Out* (FIFO) queues (per output interface) elastic traffic and interactive traffic cannot be segregated. As a result, under FIFO, the TCP-controlled sources may jeopardize the quality of service experienced by the users of the interactive services (gamers, for instance). On the other hand, one could have two queues (per output interface), served by a (non-pre-emptive) Head-of-Line (HoL) *priority* scheduler. In such a situation, however, the elastic traffic might suffer from starvation during busy periods of the high-priority queue. Therefore, an attractive alternative is a scheduler like *Weighted Fair Queuing* (WFQ) that assigns a minimum guaranteed link rate to each of the classes. In this way it provides the interactive service the capacity it needs without risking to starve the TCP-controlled traffic.

Under WFQ elastic traffic can only interfere with the interactive traffic by the fact that a data packet is in service at the instant a real-time packet becomes eligible. It can be justified to assume that this queuing delay due to the residual service of the data is statistically independent from the queuing delay due to competition in the queue dedicated to the interactive services [19]. Moreover, on links of moderate to high rate the delay associated with this kind of interference is negligible, while on links with a very small link rate (e.g., the DSL upstream link) it can be assumed that there is some form of pre-emption to interrupt a long data packet in service that would introduce too much delay on the interactive traffic. We conclude that in such a situation one can study the real-time queue in isolation.

The network provider will tune its schedulers such that the interactive traffic gets just the treatment that it needs to meet the delay (and packet loss) bound. So, although the link rates in the current and future IP-based networks are huge compared to the bit rate of one interactive flow, or even compared to the aggregate bit rate of all interactive flows, still queuing delay can occur, as the actual capacity provisioned for the interactive services will be just modestly higher than the (average) offered traffic.

In this paper we consider a system as described above, in which interactive traffic shares resources with elastic traffic. The study focuses on the situation in which the interactive traffic is generated by *gaming* users. More specifically, we assess the performance of client-server-based *First Person Shooters* (FPS) games. It is known that the quality the users experience is mainly determined by the delay. In a number of studies this influence was quantified in an objective manner, with many of these having focused on multiplayer games as representative applications [3, 10, 11, 12, 16, 17, 26, 27]. Ardent game players often cite network delay (referred to as ‘ping time’, or simply ‘ping’) as the main cause for degradation in their performance and/or scores. Most of so-called ‘hard-core’ gamers often simply choose not to connect to game servers that show a ping higher than, say, a few 100 ms [1, 2, 13, 14, 20]. This ping time is often seen as culprit because of built-in ping features in modern multiplayer games, with that ping value in itself possibly influencing the players’ performance.

Inspired by this ping time, the goal of this paper is to accurately estimate the packet round trip time (which we often abbreviate to RTT throughout this paper) in the setting described above. Here the round-trip-delay is defined as the time between the instant that a packet with information describing the action of the gamer in the virtual three-dimensional world departs from the client PC and the time instant the information with his motion vectors arrives at the client PC.

The delay incurred on an end-to-end path in the network can be decomposed into a deterministic and a stochastic (e.g. queuing) part. The deterministic delay is quite easy to determine, as it is just the sum of the serialization delay, propagation delay, and (server) processing delay. Queuing delay is more difficult to assess because it depends on the congestion state of the network. It is well known that deterministic upper bounds for queuing delay are easily obtained [7, 21, 22], but these worst-case upper bounds lead to unrealistically high values. Instead, this paper derives statistical ‘upper bounds’ (i.e., *quantiles*) for the queuing delay, leading to more realistic values [6, 9, 19].

Obviously, if gaming applications are to be offered on a commercial basis, then there is a strong need for reliable models that predict the delay (or, conversely, a need for a methodology to calculate the required capacity to be provisioned for the gaming service, given the delay bound). This paper answers this question, and can be used as a benchmark when deploying such interactive services on a large scale.

Finally, we spend a few words on the methodology and organization of this paper. A key ingredient in a performance study is the traffic model. Section 2 describes the traffic characteristics of a typical FPS gaming source, and relates these to the result of earlier measurement studies. Then this traffic model is used as the input of our queuing model. In Section 3 the queuing delay is assessed when several gaming sources are multiplexed; as argued above, such a queuing model gives a better impression of the experienced delay than deterministic worst-case bounds. The analysis relies on a decomposition of the upstream and downstream part. Section 4 uses the developed queuing model to assess the round-trip-delay typical for games played over a typical access network. Section 5 draws the conclusions.

2. Traffic Source Model

In this section, traffic characteristics of different popular on-line FPS games will be presented. We rely both on characteristics stemming from literature and on own measurements. Based on these characteristics the traffic source model to be used in this paper (most notably in the queuing model) will be further justified.

Since FPS games are highly interactive, they are the least tolerant type of online games with respect to (network) delay. As such, they are particularly appropriate in the scope of this paper, in which we have a focus on access network queuing delays of online games. In the context of this queuing analysis, packet sizes and packet arrival rates (and the distribution of the inter-arrival times) are the most relevant characteristics to consider.

Since most online gaming is real-time and requires low latency, most game communication uses small UDP packets sent almost periodically. Traffic flows are generated according to the following principle: a gaming server keeps track of the global gaming state. At timer-based intervals it sends a burst of back-to-back packets containing this gaming state-information to the clients. The server has the ability to set this update frequency (and as such the ‘inter-burst time’) as it wishes. In turn, the clients read and process these packets in order to update their current view on the screen. Next, the client’s commands are processed and an update packet, containing movement and state information of the player, is sent to the server. Also the clients send their packets at timer-based intervals, which may differ from client to client, depending on client hardware performance and settings. The server processes these packets to update the global gaming state.

In this section first an overview is given of traffic characteristics and source models for online games found in the literature. Next, traffic characteristics based on previous measurements are compared to the published ones. Finally, we summarize the traffic source model that is used further on in this paper.

2.1. Related work

There are already some papers on the characterization and/or modeling of traffic generated by different popular on-line FPS games. An early study is by Borella who presented traffic models for the FPS game ‘Quake’ [3]. A few years later, Färber did the same for the game ‘Counter Strike’ [11] which is based on the ‘Quake’ engine. He found that Borella’s game traffic model was in general still

valid. The traffic model for fast action multiplayer games Färber proposes consists of two sub-models: the server traffic model and the client traffic model.

- Regarding the *server-to-client* traffic, Färber found that the packet rate was slightly varying and that it was very bursty. In each burst, the server generates one packet for every active client. The peak inter-burst-time was found to be 55 ms, with mean 62 ms and a coefficient of variation (CoV) of 0.5. He approximates this inter-burst-time with an extreme distribution: he finds that the probability density function of the extreme distribution with $a = 55$ ms and $b = 6$ ms fits the experimental histogram best (using least square fitting). Recall that the density of the extreme distribution, respectively the associated cumulative distribution, with parameters a and b , is given by [28]:

$$f(x) = \frac{1}{b} \exp((a-x)/b) \exp(e^{-(a-x)/b}); \quad F(x) = \exp(e^{-(a-x)/b}). \quad (1)$$

We will denote this distribution by $Ext(a, b)$. Note that Färber considers inter-burst-times per client, and as such tacitly assumes that within each burst the order of the packets is the same. This is not necessarily true though – we come back to this issue in Section 2.3.2; as a consequence, the server traffic model presented by Färber might be misleading.

The server packet-size showed a higher variability. The characteristics he obtains are a mean size of 127 bytes and a CoV of 0.74. Again, he approximates (using least square fitting of the probability density function with the histogram) this with the extreme distribution $Ext(120, 36)$.

- Now focus on the *client-to-server* traffic. This can be characterized by an almost constant data and packet rate. He finds an inter-arrival time of 42 ms and a CoV of 0.24, which makes him propose a deterministic distribution, and he writes $Det(40)$. We think 40 ms is chosen, since this corresponds to a client that updates its local copy of the gaming world 25 times a second. Finally, the client packet-size distribution, characterized by a mean of 82 bytes and a CoV of 0.12, is accurately modeled by the extreme distribution $Ext(80, 5.7)$.

All these characteristics and approximations are summarized in Table 1. Note that Färber also mentions that shifted lognormal and shifted Weibull distributions lead to acceptable fits to the data as well.

	Server to client			Client to server		
	Mean	CoV	Approximation	Mean	CoV	Approximation
Packet size (in bytes)	127	0.74	$Ext(120, 36)$	82	0.12	$Ext(80, 5.7)$
Burst inter-arrival time	62	0.5	$Ext(55, 6)$	42	0.24	$Det(40)$

Table 1: Traffic characteristics for Counter Strike traffic (in terms of mean and CoV) and the suggested approximation, following Färber et al. Packet sizes are in bytes, burst inter-arrival times in ms.

Lang *et al.* developed a traffic model for the game Half-Life [16]. Their model states that the server-to-client inter-burst times are deterministic with values around 60 ms, while the server-to-client packet sizes can be modeled through lognormal distributions. It was found that the packet sizes depend on the map that is played. The client-to-server traffic on the other hand can be modeled using a deterministic function (41ms) for the inter-arrival times, and for the packet sizes it is seen that there is no dependency on whatever parameter and that the sizes range from 60 to 90 bytes. Here normal and lognormal distributions lead to equally good fits.

These findings are summarized in Table 2. In general, the authors found that the traffic pattern was only affected by the map that was played (packet sizes server to client) and the graphic rendering

software of the client computer (inter-arrival times client-to-server traffic). Moreover, their model is roughly in line with the model proposed by Färber.

	Server to client		Client to server	
	Mean	Approximation	Mean	Approximation
Packet size	Map-dependent	lognormal	No dependency	(log-)normal
Burst inter-arrival time	60	Det(60)	41	Det(41)

Table 2: Traffic characteristics for Half-Life traffic and the suggested approximation, following Lang et al. Packet sizes are in bytes, burst inter-arrival times in ms.

In another paper [17], Lang *et al.* present a model for the Xbox System Link Game Halo. The server-to-client inter-burst-time are modeled deterministically (40ms), and the same holds for the packet sizes (where the actual size depends on the number of players in the game). For the client-to-server traffic, they found that 33% of the packets (which have fixed size of 72 bytes) are sent every 201ms; the other 67% (of which the size depends on the number of players on the client Xbox) also have a constant inter-arrival time, but this inter-arrival time depends on the client Xbox hardware. From these observations the authors conclude that System Link traffic is strongly periodic, and that the traffic pattern is influenced by the number of players (server to client and client to server packet sizes) and the client Xbox hardware (client-to-server inter-arrival times).

In [18] Lang *et al.* present a traffic model for Quake3. They show that the packet lengths from server to client depend on the number of players participating in the game and to a lesser extent on the particular map. Packet lengths vary between about 50 byte and 400 byte. The server sends one update packet per client approximately every 50 ms. It is also shown in [18] that the packet length distribution for packets from client to server is independent of all observed parameters. The smaller packets are about 50 byte and the largest ones are around 70 byte. The packet transmission rate of a client is dependent on the map played and on the client’s graphic card. Typical inter-arrival times mentioned in [18] vary between 10 ms and 30 ms.

2.2. Analysis of more traces

In [23], the impact of delay and jitter on game play and subjective user experience was studied by analyzing the traffic that was logged during a LAN party in which 12 players combated against each other in the FPS game Unreal Tournament 2003. Part of this traffic will be used here to verify whether we can confirm the traffic characteristics that are summarized in Section 2.1. Because jitter was artificially introduced in this experiment we have to be careful in interpreting the inter-arrival time measurements. In particular we only used those experiments from [23] for which the introduced jitter was much lower than the mean inter-arrival time.

The traffic trace of six minutes consists of all traffic between the dedicated server and the twelve players in both directions. Like the authors in the previous paragraph, we found that the traffic from the server to the clients consists of traffic bursts, which arrive at regular intervals (about 47 ms) with a CoV of 0.07. Six of the bursts (which not even represents 0.1% of the total) however showed an inter-arrival time of almost 80 ms, whereas the following burst came with an inter-arrival time of about 15 ms. Their occurrence was not periodic and the reason for these ‘delayed’ bursts is unknown to us. With a few exceptions (about 0.5% of the bursts), all bursts contain 1 packet for each of the players. The reason why there was 1 (or in 1 case 2) packet(s) missing has not become clear; it might be due to packet loss in the network. Since we see no reason why the server would not send an update packet to each one of the clients, and the occurrence of this ‘missing packet’ is not high, we will not take this into account in the following.

Furthermore, we observed that the order of the packets within the burst was not the same for each burst. If this is only due to the jitter that was deliberately imposed during the measurements, or if this is already present at the moment the server sends the burst, is not clear. Therefore we cannot state that the order of the packets (at the moment the server sends the burst) is the same for each burst.

For the server packet size, we find a mean size of 154 bytes and a CoV of 0.28. Within a burst however, the packet size variation is less with a CoV varying from 0.05 to 0.11. Note that we also found, like the authors above, that the map that is played affects these characteristics (particularly the mean packet size). Concerning the burst sizes, we found a mean of 1852 bytes, with a CoV of 0.19. For the traffic from the clients to the server, we found a mean packet size of 73 bytes and a CoV of 0.06, the IAT was 30 ms with a CoV of 0.65. These findings are displayed in Table 3.

	Server to client		Client to server	
	Mean	CoV	Mean	CoV
Packet size	154	0.28	73	0.061
Burst inter-arrival time	47	0.07	30	0.65
Burst size	1852	0.19	-	-

Table 3: Overview of characteristics of traffic generated during a FPS (Unreal Tournament) gaming session on a LAN. Packet sizes are in bytes, burst inter-arrival times in ms, burst sizes in bytes.

2.3. Traffic Source Model

In this section, we will present the traffic source models that will be used in the remainder of this paper. Like the authors above, also here we will consider 2 traffic models: one for the server traffic and one for the client traffic.

2.3.1. Client traffic model

Based on the findings in Sections 2.1 and 2.2, the packet sizes and packet inter-arrival times of the client-to-server traffic can be approximated by deterministic distributions. Based on this, the client traffic model (representing the upstream traffic from the client’s point of view) that should be used corresponds to a periodic streams of packets (each stream corresponds to a user), but with random phasing between the streams. A large enough set of streams with periodic arrivals could be approximated by an aggregate stream with Poisson arrivals (see Section 3.1).

2.3.2. Server traffic model

Following Sections 2.1 and 2.2, the inter-arrival times of the server (burst) traffic, can be assumed to be deterministic. For the server (burst) traffic size, unlike previous authors (which concentrated on fitting the probability density function to the histogram or on the fitting the theoretical to the experimental central moments), we focus on fitting the tail of the distribution, since this dominates also the tail of the corresponding queue. We propose to model the server (burst) traffic size with an Erlang distribution; this is because this distribution fits the tail of the experimental results quite well (see Figure 1), and because of its analytical tractability.

The Erlang(K, λ) distribution has two parameters: the shape parameter λ and the Erlang order K ; its mean is K/λ and its variance K/λ^2 . We determine the mean value by fitting it to the measured average burst size (1852 byte, as seen in Table 3). In order to determine the order K of the Erlang distribution, we can focus either on fitting the CoV (i.e., the standard deviation), or on the tail of the distribution. Following the former approach, i.e., fitting the CoV and noticing from Table 3 that it is 0.19, we derive that K is 28. Following the latter approach, i.e., (visually) fitting the tail, we see from Figure 1 that K is somewhere between 15 and 20. This figure presents the experimental tail distribution

function of the burst sizes as measured in our experiments as well as of the tail distribution functions of Erlang distributions for $K = 15, 20$ and 25 , respectively.

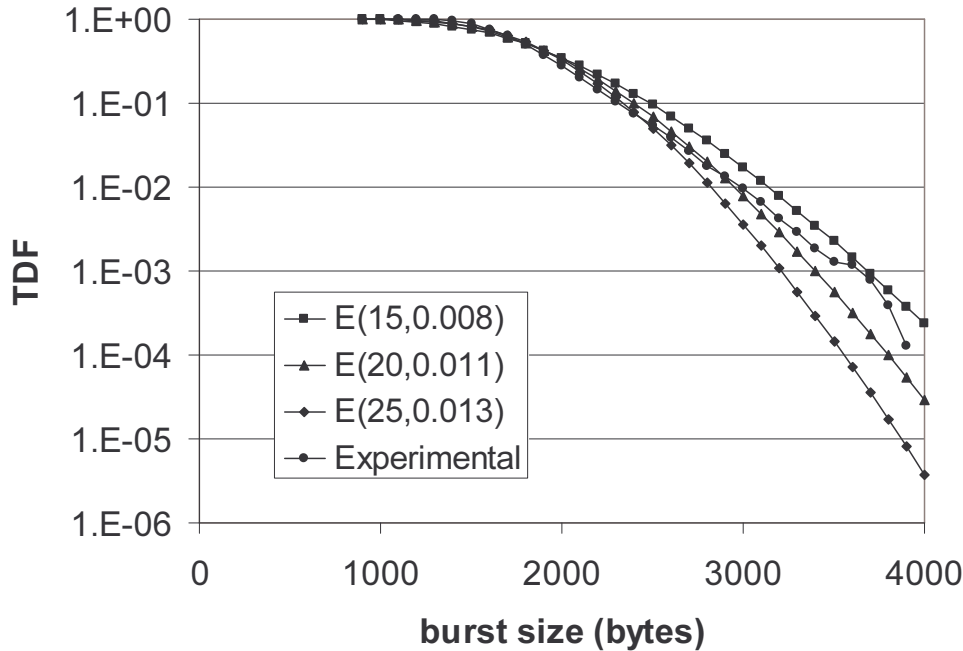


Figure 1: Tail distribution function of the measured burst sizes and Erlang tail distributions of different order for which the mean was already fit to the measured mean.

Following the above reasoning, the server traffic model could be approached by $D/E_K/1$, where K is of the order 20. As already mentioned (see Section 2.2), the CoV of packet sizes within a burst is a substantially smaller than the overall CoV of server packet sizes. Based on that fact, we expect that the impact of the number of gamers on the burst size distribution, and hence on K , is small. Nevertheless, the choice of K has a significant impact on the dimensioning; we return to this in Section 4.

Finally, taking into account the literature studies (Section 2.1) and our own measurements, it is clear that the traffic characteristics of online games differ considerably from game to game. For that reason, different choices for K will be considered in the remainder of the paper.

3. Queuing Model

In this section we assess the stochastic part of the round trip time (i.e., the ‘ping time’) a gamer experiences. As the deterministic part is straightforward to calculate (see Section 1), we do not consider it in this section, but we do take this deterministic part into account in Section 4. The stochastic part is the sum of two random delays, the upstream and downstream queuing delay, which we assume to be statistically independent.

The upstream queuing delay is due to the competition of packets stemming from different clients on the link to the server (see Figure 2). Indeed, it may happen that a packet from a certain client arrives on that aggregation point at the instant a packet from another client is being transported over the link towards the server, such that this packet will have to queue. In such a way a queue can temporary build up in that aggregation point. Because we assume that on the long run less packets are arriving at this point than can be served, this queue should regularly empty. Packets that arrive in an empty queue experience no queuing delay, while packet arriving in a full queue may experience substantial queuing delay. In Section 3.1 we determine the stochastic law that characterizes this queuing delay by arguing that the queuing behavior can be modeled by an $M/G/1$ queuing system.

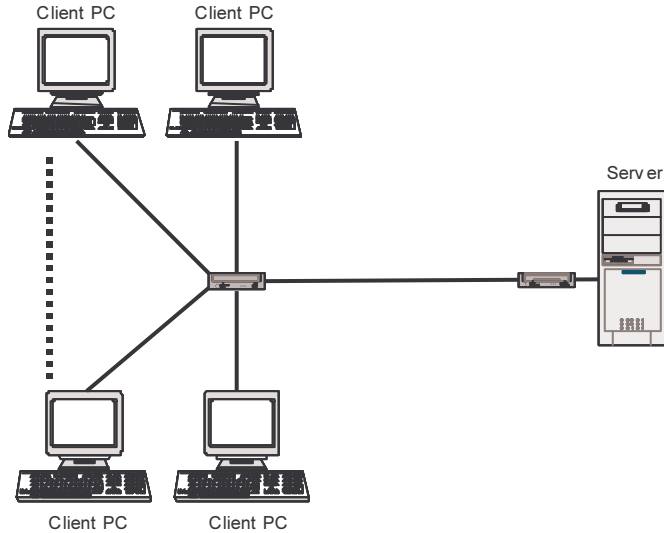


Figure 2: Client-server architecture for interactive gaming.

The downstream queuing delay is a consequence of the fact that an arriving burst on the link (from the server towards the fan-out point, see Figure 2) to the clients may see a residual part of the previous burst. Indeed, the burst size is highly variable (which we model by an Erlang distribution, see Section 2.3.2) and it is not guaranteed that the (e.g., 40 ms) inter-arrival times will always be long enough to transport all packets within the burst over that link. In that way a queue may build up. Since we assume that also in the downstream direction the queuing system is stable, this queue has to regularly empty as well. Bursts that see an empty queue experience no queuing delay, while other bursts may experience a considerable queuing delay. We determine the stochastic law that characterizes the queuing delay of the bursts in Section 3.2.1. On top of that delay, a packet in a particular burst sees an additional delay due to all packets within that burst that are in front of it. Packets in the beginning of the burst hardly see any queuing delay of this type, while the packets at the end of the burst will have to wait a significantly longer time. In Section 3.2.2 we determine the probabilistic law that governs this delay.

We determine all probabilistic laws by deriving the associated probability generating functions. In Section 3.3 we determine the probability generating function of the total queuing delay and explicitly invert this probability generating function to obtain the tail distribution function of the total queuing delay. From this tail distribution function we determine the quantile of the queuing delay.

3.1. Upstream: M/G/1

Relying on the findings of Section 2, we can assume that in the upstream direction equally sized packets are transmitted with virtually identical inter-arrival times, say D ; let N be the number of users. The resulting queuing model would be $N*D/D/1$, see for instance Roberts *et al.* [25]. An explicit formula for the queuing delay in the $N*D/D/1$ queue is known, but quite intransparent, and relatively computationally demanding. We will argue here that we can do with simpler formulas.

Let Q be the steady-state buffer content in an $N*D/D/1$ queue with link rate C (where it is noted that a buffer content can be translated into delay immediately, due to the constant service speed). Define $A(t)$ as the amount of traffic arriving in an arbitrary window of length t . It is a known fact that the steady-state buffer content Q has the same distribution as the maximum of the so-called free process $A(t)-Ct$ (where the process is called ‘free’ because $A(t)-Ct$ can attain any value in $(-\infty, \infty)$; unlike the queuing process that is reflected at 0, and hence lives on $[0, \infty)$). We thus have

$$P(Q > B) = P(\sup_{t \geq 0} A(t) - Ct > B) = P(\exists t \geq 0 : A(t) - Ct > B). \quad (2)$$

An often very accurate approximation is to replace the union of events $\{\exists t \geq 0 : A(t) - Ct > B\}$ by its dominant term (i.e., the term with the largest contribution). This leads to

$$P(Q > B) = P(\exists t \geq 0 : A(t) - Ct > B) \approx \sup_{t \geq 0} P(A(t) - Ct > B) . \quad (3)$$

Now it is noted that $A(t)$ has a binomial distribution with parameters N and t/D . In self-evident notation, we arrive at the approximation

$$P(Q > B) \approx \sup_{t \geq 0} P(A(t) - Ct > B) = \sup_{t \geq 0} P(\text{Bin}(N, t/D) > B + Ct) . \quad (4)$$

This expression still has the drawback that a tail probability of the binomial distribution needs to be calculated. Therefore we do additional simplifications. For any random variable X with finite moment generating function in a neighborhood of 0, one can apply the celebrated Chernoff bound:

$$P(X > x) \leq \inf_{s \geq 0} \left(e^{-sx} \cdot Ee^{sX} \right) ; \quad (5)$$

where the right-hand side is known to be not only an upper bound but also a rather accurate estimate. Our random variable $A(t)$ satisfies the requirements of the Chernoff bound, and

$$Ee^{sA(t)} = \left(1 - \frac{t}{D} + \frac{t}{D} e^{ps} \right)^N , \quad (6)$$

where p is the packet size. This all leads to the ‘large-deviations estimate’

$$P(Q > B) \approx \sup_{t \geq 0} \inf_{s \geq 0} e^{-s(B+Ct)} \left(1 - \frac{t}{D} + \frac{t}{D} e^{ps} \right)^N \quad (7)$$

or

$$\log P(Q > B) \approx - \inf_{t \geq 0} \sup_{s \geq 0} \left(s(B+Ct) - N \log \left(1 - \frac{t}{D} + \frac{t}{D} e^{ps} \right) \right) . \quad (8)$$

As the optimizing s reads

$$s^* = \frac{1}{p} \log \left(\frac{D-t}{t} \cdot \frac{B+Ct}{Np - B - Ct} \right) \quad (9)$$

it further simplifies to

$$\log P(Q > B) \approx - \inf_{t \geq 0} \left(\frac{B+Ct}{p} \log \left(\frac{D-t}{t} \cdot \frac{B+Ct}{Np - B - Ct} \right) - N \log \left(\frac{D-t}{D} \cdot \frac{Np}{Np - B - Ct} \right) \right) . \quad (10)$$

Now we have an accurate expression for buffer content distribution in the $N^*D/D/1$ queue, we now study what happens when the number of users grows large. To keep the load constant, we let the inter-arrival times grow at the same pace. In other words: we replace N by nN , and D by nD . Interestingly,

$$Ee^{sA(t)} = \left(1 - \frac{t}{nD} + \frac{t}{nD} e^{ps} \right)^{nN} \xrightarrow{n \rightarrow \infty} \exp \left(\frac{tN}{D} (e^{ps} - 1) \right) , \quad (11)$$

which coincides with the moment generating function of a Poisson stream (of packets of size p), with arrival rate N/D ! In other words: the input stream converges to Poisson when (for given load) the number of periodic streams increases, and at the same time the inter-arrival times increase. This result is reminiscent of those for renewal streams by Cox [8] and Cao and Ramanan [24]. In other words, when the number of gamers grows large, we are to analyze the $M/G/1$ queue. With the same techniques as applied above, we obtain

$$\log P(Q > B) \approx - \inf_{t \geq 0} \left(\frac{B+Ct}{p} \log \left(\frac{D}{t} \cdot \frac{B+Ct}{Np} \right) - N \left(\frac{B+Ct}{Np} - \frac{t}{D} \right) \right) . \quad (12)$$

This formula also appear when we replace N by nN , and D by nD in the above formula for $P(Q > B)$ for the $N^*D/D/1$ queue, and we let n grow large.

Now that we have further mathematically justified the use of M/D/1 queues when the number of sources grows large, we notice that empirical observations indicate [25] that the M/D/1-approximation works very well, particularly for relatively small load (there are serious deviations for load above, say, 90%). The regime with relatively low load is the regime we are interested in.

Let us now investigate what happens when there are two classes of gamers (consisting of N_1 and N_2 users, respectively), each with their own packet size and packet inter-arrival times. The above reasoning still applies if the number of users in each of the classes grows large, and we find the approximate moment generating function

$$Ee^{sA(t)} = \exp\left(\frac{tN_1}{D_1}(e^{p_1s} - 1)\right) \exp\left(\frac{tN_2}{D_2}(e^{p_2s} - 1)\right) . \quad (13)$$

Based on this moment generating function we can again do the above steps (i.e., dominant term and Chernoff bound) to find an explicit expression. An alternative is to use standard machinery for M/G/1 (in which class this model falls; in fact at any arrival one could flip a coin to decide from which class the arrival is). For M/G/1 powerful tools are available; in particular, the transform of the buffer content is explicitly known in terms of the transform of the service requirements.

The conclusions from this section are twofold: (1) when the number of users grows, one can safely assume Poisson arrivals (for which claim additional algebraic support was provided), (2) when there are multiple classes of users, one may use results for the M/G/1 queue (either by applying the exact results known for this queue, or by resorting to fairly explicit ‘inf sup’-type of approximations).

The moment generating function of the delay in an M/G/1 system is well known [15]. Later in this paper, we approximate this moment generating function of the upstream queuing delay as

$$D_u(s) \approx 1 - \rho_u + \rho_u \frac{\gamma}{\gamma - s} , \quad (14)$$

where γ is the dominant pole of the exact moment generating function (and ρ_u is the upstream load).

3.2. Downstream D/E_K/1

As explained in Section 2, in the downstream direction the gaming server sends the packets in bursts, which (in all, except a few, cases) contain a packet for the each gamer. The burst sizes are modeled as i.i.d. random variables with an Erlang distribution of order K as marginal distribution. Bursts are sent at constant inter-departure time T (presumably the refresh interval with which the server updates its virtual 3D world).

If the traffic generated by 1 server is sent over a reserved bit pipe, the queuing in the downstream direction is modeled with the D/E_K/1 queuing model. If traffic stemming from more servers is transported over a reserved bit pipe, the N*D/G/1 queuing model applies where $G = \sum E_K$ (i.e., a weighted mix of Erlang distributions), which, based on an argument similar to the one developed in the previous paragraph, is very well approximated by M/G/1, if the number of servers is high enough.

In the next subsection we first determine the tail distribution of the delay seen by the bursts in the D/E_K/1 queuing model, then we determine the tail distribution function of a (tagged) packet within a burst.

3.2.1. Delay distribution of bursts in the D/E_K/1 queuing model

Let w_n and v_n be the remaining work (expressed in [s]) in the system just before and just after the n -th arrival instant respectively. Notice that the variable w_n is also the delay seen by the n -th burst. The system evolution is then described as

$$\begin{aligned} v_n &= w_n + b_n \\ w_{n+1} &= (v_n - T)^+ \end{aligned} , \quad (15)$$

with b_n the amount of work arriving (expressed in [s], i.e., the burst size divided by the reserved rate) at the n -th arrival instant, which are i.i.d. with an Erlang distribution of order K as marginal distribution, and T (expressed in [s]) the amount of work that can be performed between two arrival instants.

In steady state (which we assume to exist) the pdfs of random variables tend to a limit. We define the moment generating functions for the random variables in steady state as, e.g.,

$$V(s) = E[e^{sv_n}] \quad , \quad (16)$$

and similarly for other random variables. Since the burst size follows a Erlang- K distribution we have

$$B(s) = \left(\frac{\beta}{\beta - s} \right)^K \quad , \quad (17)$$

with $\beta = K/\bar{b} > 0$ and \bar{b} is the average service time for one burst.

We assume that $W(s)$ is of the form

$$W(s) = \left(1 - \sum_{j=1}^J a_j \right) + \sum_{j=1}^J a_j \frac{\alpha_j}{\alpha_j - s} \quad . \quad (18)$$

We expect a constant term (because we expect the system to be empty from time to time) and we assume that all singularities are poles (corresponding to exponential tails), with each pole α_j of multiplicity 1 (a fact that we will verify later). A necessary condition for expression (18) to be an moment generating function of a random variable is: $\text{Re}[\alpha_j] > 0$ for all j (i.e., all poles should lie in the left half of the complex plane).

Since w_n and b_n are statistically independent, the first of the evolution eqs. (15) directly yields a relation between moment generating functions:

$$V(s) = W(s)B(s) \quad . \quad (19)$$

In Appendix A it is proven that this can be rewritten as

$$V(s) = \sum_{k=0}^{K-1} \frac{(-\beta)^k W^{(k)}(\beta)}{k!} \left(\frac{\beta}{\beta - s} \right)^{K-k} + \sum_{j=1}^J a_j B(\alpha_j) \frac{\alpha_j}{\alpha_j - s} \quad . \quad (20)$$

Notice that we (tacitly) assumed that the poles α_j are different from the pole β (a fact that we will verify later).

In general, the second of the evolution eqs. (15) is hard to translate in an explicit relation between moment generating functions, but for a weighted sum of Erlang terms, as eq. (20), an explicit relation can be constructed. Using the result of Appendix B we have that

$$W(s) = \sum_{k=0}^{K-1} \frac{(-\beta)^k W^{(k)}(\beta)}{k!} \left[1 + \sum_{i=1}^{K-k} \left[\left(\frac{\beta}{\beta - s} \right)^i - 1 \right] \frac{(\beta T)^{K-k-i}}{(K-k-i)!} \exp(-\beta T) \right] \\ + \sum_{j=1}^J a_j B(\alpha_j) \left[1 + \left[\left(\frac{\alpha_j}{\alpha_j - s} \right) - 1 \right] \exp(-\alpha_j T) \right] \quad . \quad (21)$$

For the right hand side of this equation to be equal to the original $W(s)$ of eq. (18)

1. all terms in the first sum in the right hand side of eq. (21) need to be 0 (for all s), and
2. the coefficients of the terms in the second sum of the right hand side of eq. (21) need to be equal to the coefficients of the corresponding terms in the original $W(s)$ of eq. (18).

This leads to a set of $(K+L+1)$ equations:

$$W^{(k)}(\beta) = 0 \quad \forall k \in \{0, \dots, K-1\} \quad , \quad (22)$$

$$\sum_{j=1}^J a_j [1 - \exp(-\alpha_j T)] B(\alpha_j) = 1 - \sum_{l=1}^L a_l \quad , \quad (23)$$

$$\exp(-\alpha_j T) B(\alpha_j) = 1 \quad \forall j \in \{1, \dots, J\} \quad . \quad (24)$$

The set of equations (24) determines the poles. In Appendix C we prove that there are K poles, and that each pole is given by

$$\alpha_k = -\frac{K(\zeta_k - 1)}{\bar{b}} \quad \forall k \in \{1, \dots, K\} \quad , \quad (25)$$

where ζ_k is the unique solution of

$$z = \exp\left(\frac{z-1}{\rho_d} + j \frac{2\pi(k-1)}{K}\right) \quad \forall k \in \{1, \dots, K\} \quad , \quad (26)$$

in $\text{Re}[z] < 1$, where ρ_d is the load defined as \bar{b}/T (which is a value between 0 and 1 for a stable system).

Once the poles α_j are determined the set of equations (22) together with equation (23) determine the weights a_j . In Appendix D we prove that set (22) together with equation (23) (comprising $K+1$ equations) contains one redundant equation and has as solution

$$a_j = \left(\zeta_j\right)^K \prod_{\substack{k=1 \\ k \neq j}}^K \frac{\zeta_k - 1}{\zeta_k - \zeta_j} \quad j = 1, \dots, K \quad . \quad (27)$$

Notice that for the special case D/M/1 ($K=1$) exactly the same solution as in [15] is obtained.

3.2.2. Delay distribution of packets

A tagged packet arriving in a certain burst experiences a delay $d_{n,t}$ composed of two contributions

$$d_{n,t} = w_n + p_{n,t} \quad . \quad (28)$$

First, the tagged packet has to wait for all the remaining work in the system the instant it arrived, which is equal to the delay w_n the burst sees. Second, it has to wait a time $p_{n,t}$ due the packets that jointly arrived in the same burst and that are in front of the tagged packet.

Similarly as above we define the moment generating functions $D_d(s)$ and $P(s)$ for the steady state random variables $d_{n,t}$ and $p_{n,t}$ respectively. As the delay components w_n and $p_{n,t}$ are statistically independent, we have that

$$D_d(s) = W(s)P(s) \quad . \quad (29)$$

To assess $P(s)$ we need to know where in the arriving burst the tagged packet resides. This is described by a random variable $u_{n,t}$ with pdf $p_u(u_{n,t})$ with support $[0,1]$. For instance, $u_{n,t}=0$ means that that the tagged packet is the first of the n -th burst. We assume that that there are so many packets in a burst, such that the size of a packet is negligible with respect to the size of the burst, and hence, $u_{n,t}$ can take any value in $[0,1]$. For instance, $u_{n,t}=1$ means that the packet is the last one of the burst and sees (practically) the whole burst before it. By first conditioning on the burst size, then averaging over all possible burst sizes and a change of variables, it follows that

$$P(s) = \int_0^1 d\tau \left(\frac{\beta}{\beta - s\tau} \right)^K p_u(\tau) \quad . \quad (30)$$

For general distributions $p_u(u_{n,t})$, no conclusions can be drawn with respect to the poles of $P(s)$, but we consider two special cases.

The first case is the one where the tagged packet always resides on the same spot θ (a number in $[0,1]$) in the burst, i.e.,

$$p_u(\tau) = \delta(\tau - \theta) \quad . \quad (31)$$

In this case

$$P(s) = \left(\frac{\beta}{\beta - s\theta} \right)^K \quad . \quad (32)$$

This moment generating function $P(s)$ has a pole β/θ with multiplicity K , which is most dominant if $\theta=1$. Even in this worst case, the dominant pole of $W(s)$ dominates this pole.

The second case is the one where from burst to burst the packet can reside anywhere in the burst, i.e., where $p_u(u_{n,t})$ is the uniform distribution. In this case, the integral defining $P(s)$ can be calculated (by identifying a primitive function of the integrand)

$$P(s) = \begin{cases} \frac{\beta}{(K-1)s} \left[\left(\frac{\beta}{\beta-s} \right)^{K-1} - 1 \right] & K > 1 \\ \frac{\beta}{s} \ln \left(\frac{\beta}{\beta-s} \right) & K = 1 \end{cases} \quad . \quad (33)$$

Note that in both cases $s=0$ is no singularity. For $K=1$ the singularity is a branching point $s=\beta$, while for all other values of K the singularity is a pole $s=\beta$ of multiplicity $K-1$. In fact in the latter case it can be written as a weighted sum of Erlang terms (using Horner's rule)

$$P(s) = \frac{1}{(K-1)} \sum_{k=0}^{K-2} \left(\frac{\beta}{\beta-s} \right)^{K-1-k} \quad K > 1 \quad . \quad (34)$$

Again the dominant pole of $W(s)$ dominates this pole.

Further, we only consider this case where the packet can be anywhere in the burst and $K > 1$.

3.3. Combining uplink and downlink queuing delay

There are three contributions to the total queuing delay: the upstream queuing delay, the downstream delay of the burst the packet is part of and the delay due to the fact that the packet can be in a random place in the burst.

We assume that the upstream and downstream queuing delay are statistically independent, so that the moment generating functions of the total delay is a product $D_u(s)W(s)P(s)$. As can be seen from eqs. (14), (18) and (34) respectively, each of these terms can be (approximately) written as a sum of Erlang terms, such that, with the technique in Appendix A, we can write this product as a sum of Erlang terms

$$\begin{aligned} & (W(\gamma)P(\gamma)\rho_u) \left(\frac{\gamma}{\gamma-s} \right) + \sum_{j=1}^K (D_u(\alpha_j)P(\alpha_j)\alpha_j) \left(\frac{\alpha_j}{\alpha_j-s} \right) \\ & + \frac{1}{(K-1)} \sum_{k=0}^{K-2} \left(\sum_{l=0}^k (-\beta)^l \sum_{m=0}^l \frac{D_u^{(m)}(\beta)W^{(l-m)}(\beta)}{m!(l-m)!} \right) \left(\frac{\beta}{\beta-s} \right)^{K-1-k} \quad , \end{aligned} \quad (35)$$

which is trivial to invert.

A further approximation is to neglect all terms but the dominant pole in eq. (35). This is a good approximation as long as the residue associated with the dominant pole is not too small.

The method of the dominant pole is almost equivalent to using the Chernoff bound:

$$P(D > d) = P(D_{up} + D_{down} > d) \approx \inf_{s \geq 0} \left(e^{-sd} D_u(s)W(s)P(s) \right) \quad (36)$$

as long as $P(D>d)$ is small enough.

Finally, the quantile of a sum of (statistically independent) delay contributions can be approximated by the sum of the quantiles of the individual delay terms.

In this paper we use the first method to determine the quantile.

4. Numerical Results

In this section we assess the impact of several parameters on the RTT performance for First Person Shooter games by considering a number of representative scenarios. Our starting point is the architecture depicted in Figure 2 in Section 3.

It has been shown in the previous sections that several variables have impact on the RTT performance. In this section we wish to quantify this impact. The variables we have identified are:

- *the load on the aggregation link*; evidently, a higher load induces a higher delay;
- *the mean size of packets from the server to the clients*; for a fixed load larger packet sizes lead to both higher queuing and serialization delays;
- *the inter-arrival time of bursts of packets from the server to the clients*; longer inter-arrival times lead to possible longer waiting times;
- *the order K of the Erlang distribution of the size of the burst of packets from the server to the clients*; the higher the K for given load the lower the CoV, and hence the lower the queuing delay.

In order to keep the number of scenarios limited we make the following assumptions: the size of packets from client is fixed at $P_C = 80$ byte, the uplink access bandwidth is fixed at $R_{up} = 128$ kbps, the downlink access bandwidth is fixed at $R_{down} = 1024$ kbps. In addition we suppose that the inter-arrival time T of bursts of packets from the server equals the inter-arrival time of packets from the clients (per client). We will fix T at either 40 ms or 60 ms. This choice is motivated by the findings of Section 2.

For the aggregation link rate C we initially consider the value $C = 5000$ kbps. For the size of packets sent from the server, denoted by P_S , we take the values 125 byte, 100 byte and 75 byte. The order K of the Erlang distribution (representing the size of the bursts of packets from the server) is assumed to be $K = 2, 9$ or 20 .

For each scenario we have computed the RTT for various values of ρ , the load on the aggregation link in the downlink direction. One can verify that this load is given by

$$\rho_d = \frac{8NP_s}{TC}, \quad (37)$$

where N denotes the number of active gamers and P_S given in bytes, T in ms and C in kbps. A similar formula applies for the uplink load.

In line with [6, 9, 19] we choose to compute 99,999% quantiles of the RTT. Obviously the RTT also includes serialization delay on the access link and the aggregation link, both in uplink and downlink direction.

For all scenarios considered we have used the methodology developed in Section 3 to assess the RTT quantiles. Figure 3 shows the RTT quantiles for the case $P_S = 125$ byte, IAT = 60 ms, for several values of the Erlang order K . For this parameter set the uplink load ρ_u is always much smaller than the downlink load ρ_d . Hence, the upstream delay component D_{up} is always negligible with respect to the downstream delay component D_{down} . As mentioned before the latter has two components: the delay of a burst and the delay due to the position of a packet within a burst. For low loads it is easy to see that the burst delay is small (because the probability that a burst is still being processed when the next burst arrives is very small), and hence, that the packet position delay dominates. This packet position delay

(and its quantile) is proportional to the number of packets within a burst (i.e., the number N of gamers) and hence, is proportional to the load ρ_d . This linear behavior can be observed in Figure 3 for low load values. As the load ρ_d increases, the burst delay starts to dominate and the curves tend to the asymptote $\rho_d = 1$.

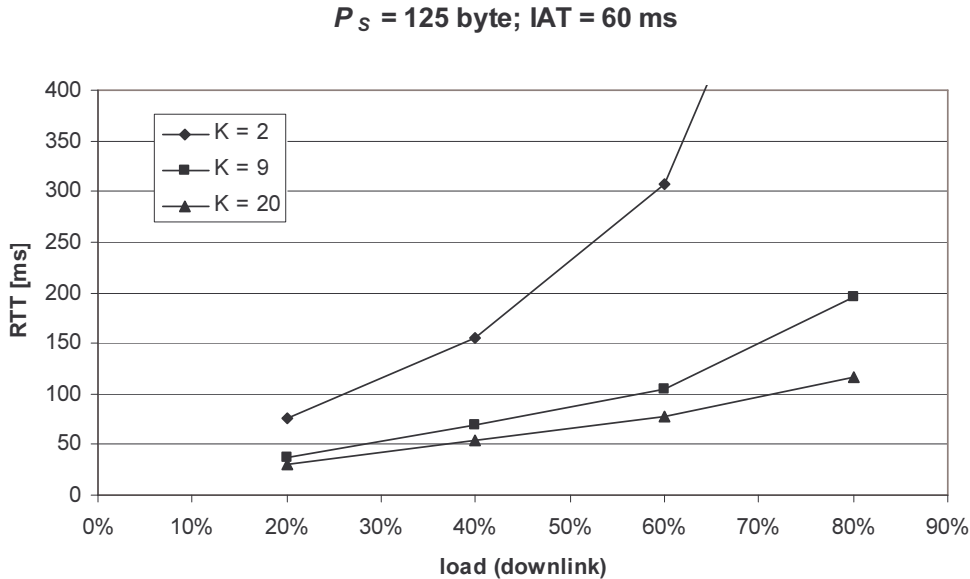


Figure 3: Impact of Erlang order K on the RTT.

It is obvious from Figure 3 that the order K of the Erlang distribution has a strong impact on the RTT quantiles. We observe that indeed RTT quantiles are highly sensitive to the Erlang order; even at moderate load, low values of K lead to unacceptable RTT performance.

This observation is robust with respect to the choice of P_S . We have done the same experiment for $P_S = 100$ byte and $P_S = 75$ byte and obtained nearly the same behavior as depicted in Figure 3. Note that if $P_S > P_C$ (which is the case for $P_S = 125$ byte and $P_S = 100$ byte) the downlink load is higher than the uplink load. Hence, we expect the contribution of the downlink queuing model ($D/E_K/1$) to dominate the contribution of the uplink model. On the contrary, if $P_S < P_C$, then for a sufficiently high downlink load the uplink will be dominant. In fact, one can verify that in our case, for $P_S = 75$ byte, a downlink load of 75/80 corresponds to an uplink load of 1.

Figure 4 shows the RTT quantiles for the scenario $P_S = 125$ byte, $K = 9$, for the cases $T = 40$ ms and $T = 60$ ms.

As expected, Figure 4 exemplifies that higher inter-arrival times lead to higher RTT quantiles. In fact, it can be shown that if the downlink contribution is dominant (which is the case if the downlink load is higher than the uplink load) then the RTT is virtually proportional to the inter-arrival time T . To be more precise, in Figure 4 the RTT for $T = 60$ ms is about 3/2 times as high as the RTT for $T = 40$ ms.

It turns out that if we change the parameters R_{up} , R_{down} and C , then the results hardly change. This is due to two facts. First of all, the structure of our downlink queuing model is such that it is invariant with respect to the capacity C : only the load determines the quantile value. So the only effect of changing R_{up} , R_{down} and C is a different value for the serialization delay. However, this is a minor effect (in the order of 1 or 2 ms) because of the small packet sizes in combination with relatively high access and aggregation link rates.

Note that if the aggregation capacity is realized through WFQ, then the actual capacity may be higher, particularly if the other WFQ classes do not use their share of the capacity. This will result in a lower load, and hence smaller RTTs.

$P_S = 125$ byte; $K = 9$

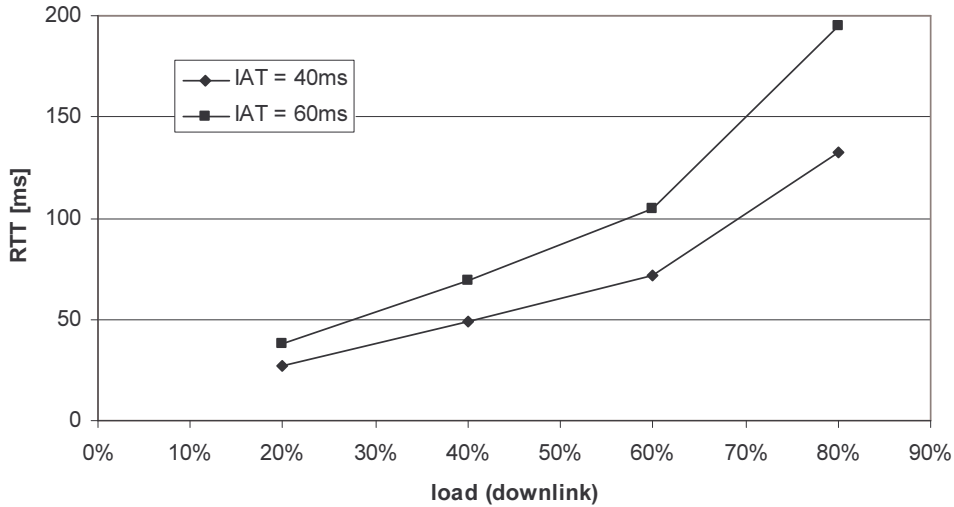


Figure 4: Impact of the inter-arrival time (IAT) on the RTT

The above results can be used for dimensioning purposes. For instance, assume we consider the case $P_S = 125$ byte, $K = 9$, $IAT = 40$ ms. Then in order to realize a maximum RTT of 50 ms (corresponding with excellent game play, according to [11]) the aggregation link allows a load (in the downlink direction) of at most 40%, see Figure 4. It is obvious that the maximum allowable load ρ_{max} is very sensitive with respect to the parameter K . Based on Figure 3 and using the fact that the RTT is proportional to T (see Figure 4), it can be concluded that for $P_S = 125$ byte, $T = 40$ ms the maximum allowable load is about 20% for $K = 2$, while for $K = 20$ the threshold ρ_{max} is around 60%.

Using equation (37) this dimensioning rule leads to a maximum number of on-line gamers N_{max} of

$$N_{max} = \frac{TC\rho_{max}}{8P_s}.$$

So in our example, where $P_S = 125$ byte, $T = 40$ ms and $C = 5000$ kbps, the maximum number of on-line gamers is 40, 80 and 120 for $K = 2, 9$ and 20 , respectively.

5. Concluding remarks

In this paper we have proposed a methodology for predicting Ping Times in First Person Shooter games. Our study consisted of three parts: (1) reflections on the traffic model to be chosen; (2) a queuing-theoretic methodology to determine the quantiles of the Round Trip Time (RTT); (3) assessment of the round trip time for realistic scenarios.

We numerically evaluated the RTT quantiles for an architecture where traffic of several gamers is aggregated on a bottleneck link. Interestingly, our main observation is that our model indicates that for realizing excellent game play, the allowable load on this aggregation link is relatively low. Furthermore, it is shown that the RTT performance is strongly determined by the burst inter-arrival times (of packets from the server to the clients). Less straightforward is the dependence on the order of the Erlang distribution (which is used to model the burst size); we observed a remarkably strong impact of this Erlang order on the RTT quantiles. In view of this effect, we expect that it would pay off to more accurately determine the Erlang order by tracing packets in real-life FPS games on a considerably larger scale.

6. Appendices

6.1. Appendix A

In this appendix we consider the product of (rational) moment generating functions where each individual factor in the product can be written as a sum of Erlang terms and a constant term and where the factors do not have poles in common.

We first remark that if all factors in the product have a constant term, then the product has a constant term as well. This constant term corresponds to a Dirac-pulse.

The inverse of a (rational) moment generating function (with the degree of the numerator not higher than the degree of the denominator) can be calculated by calculating the residue in each pole. Since (by assumption) the factors in the product do not have any poles in common, the poles of the product are the poles of each factor in the product and the multiplicity of each pole in the product remains the same as the multiplicity of the poles of the factors. Consider one of these poles, λ with multiplicity M . Isolating the terms corresponding to this pole, the product can be written as

$$R(s) \left(T(s) + \sum_{m=0}^{M-1} A_m \left(\frac{\lambda}{\lambda - s} \right)^{M-m} \right), \quad (38)$$

where $T(s)$ and $R(s)$ are a rational functions that are analytical in the neighborhood of $s=\lambda$. We calculate the contribution of this pole λ to the inverse of the moment generating function. By definition, this contribution is given by

$$\lim_{s \rightarrow \lambda} \left\{ \frac{(-1)}{(M-1)!} \frac{d^{M-1}}{ds^{M-1}} \left[R(s) \sum_{m=0}^{M-1} (-\lambda)^{M-m} A_m (s-\lambda)^m \exp(-st) \right] \right\} \quad (39)$$

which, using the rule for the $(M-1)$ -th derivative of a product, can be modified to

$$\sum_{k=1}^{M-1} \lim_{s \rightarrow \lambda} \left\{ \frac{1}{k!} \frac{d^k}{ds^k} \left[R(s) \sum_{m=0}^{M-1} (-\lambda)^{M-m} A_m (s-\lambda)^m \right] \right\} \frac{1}{(-\lambda)^{M-k}} \lambda \frac{(\lambda T)^{M-k-1}}{(M-k-1)!} \exp(-\lambda t) . \quad (40)$$

Using the same rule again for the product between the square brackets and using the fact that

$$\lim_{s \rightarrow \lambda} \left\{ \frac{1}{l!} \frac{d^l}{ds^l} \left[\sum_{m=0}^{M-1} (-\lambda)^{M-m} A_m (s-\lambda)^m \right] \right\} = (-\lambda)^{M-l} A_l \quad (41)$$

we have that

$$\sum_{k=0}^{M-1} B_k \lambda \frac{(\lambda T)^{M-k-1}}{(M-k-1)!} \exp(-\lambda t) \quad (42)$$

with

$$B_k = \sum_{l=0}^k \frac{(-\lambda)^{k-l} R^{(k-l)}(\lambda)}{(k-l)!} A_l , \quad (43)$$

where $R^{(k-l)}(\lambda)$ is the $(k-l)$ -th derivative of $R(s)$ evaluated in λ . Remark that the above equation expresses B_k as a discrete convolution of the sequence A_l with some other sequence depending on the derivatives of $R(s)$ evaluated in λ .

Taking the Laplace transform of this again, we have that

$$\sum_{k=0}^{M-1} B_k \left(\frac{\lambda}{\lambda - s} \right)^{M-k} . \quad (44)$$

If this procedure is repeated for all poles of each term in the product, this proves that the product of rational moment generating functions can be written as a sum of Erlang terms. The coefficients in this new sum can be found by convolving the coefficients in the original terms with an appropriate series that depends on derivatives of the factors in the original product.

Now let us consider the special case of Section 3.2.1

$$V(s) = W(s)B(s) \quad . \quad (45)$$

We assume that the pole β is different from all poles α_j so that we have to apply the above procedure $L+1$ times: once for each pole α_j , each of multiplicity 1, and once for pole β of multiplicity K .

The contribution associated with each pole α_j is

$$a_j B(\alpha_j) \frac{\alpha_j}{s - \alpha_j} \quad , \quad (46)$$

while the one associated with pole β is

$$\sum_{k=0}^{K-1} \frac{(-\beta)^k W^{(k)}(\beta)}{k!} \left(\frac{\beta}{\beta - s} \right)^{K-k} \quad . \quad (47)$$

This gives the desired results of eq. (20).

Finally, let us rewrite the moment generating function of the total delay, i.e., the product $D_u(s)W(s)P(s)$ of Section 3.3 as a sum of Erlang terms. As can be seen from Eqs. (14), (18) and (34) respectively each factor is a weighted sum of Erlang terms. The only approximation needed is the one shown on Eq. (14). Applying the above procedure to the poles γ and α_j is trivial, while for the pole β we make use of

$$(D_u(s)W(s))_{s=\beta}^{(l)} = l! \sum_{m=0}^l \frac{D_u^{(m)}(\beta)W^{(l-m)}(\beta)}{m!(l-m)!} \quad . \quad (48)$$

Taking all these contributions together this leads to the result of Eq. (35).

6.2. Appendix B

In this appendix we translate the second of the evolution Eqs. (15) in a relation between mgfs, provided $V(s)$ is a sum of Erlang terms.

Using the definition of v_n we have that

$$\begin{aligned} W(s) &= E\left[\exp(-s(v_n - T)^+)\right] \\ &= \Pr[v_n \leq T]E\left[\exp(-s(v_n - T)^+) \mid v_n \leq T\right] + \Pr[v_n > T]E\left[\exp(-s(v_n - T)^+) \mid v_n > T\right] \\ &= 1 - \Pr[v_n > T] + \Pr[v_n > T]E\left[\exp(-s(v_n - T)) \mid v_n > T\right] \\ &= 1 - f(0, T) + f(s, T) \end{aligned} \quad (49)$$

with

$$f(s, T) = \int_T^{+\infty} d\tau \exp(-s(\tau - T))p_{v_n}(\tau) = \int_0^{+\infty} d\tau \exp(-s\tau)p_{v_n}(\tau + T) \quad , \quad (50)$$

which is the Laplace transform (not a mgf!) of the left-shifted version of the pdf of v_n . If $V(s)$ is a sum of Erlang terms, $f(s, T)$ can be explicitly calculated.

First, we consider one Erlang term

$$\left(\frac{\gamma}{\gamma - s} \right)^l \quad . \quad (51)$$

For such a form the auxiliary function is given by

$$f(s, T) = \sum_{i=1}^I \left(\frac{\gamma}{\gamma - s} \right)^i \frac{(-\gamma T)^{I-i}}{(I-i)!} \exp(\gamma T) \quad , \quad (52)$$

and hence, the Erlang term has a contribution

$$1 + \sum_{i=1}^I \left[\left(\frac{\gamma}{\gamma - s} \right)^i - 1 \right] \frac{(-\gamma T)^{I-i}}{(I-i)!} \exp(\gamma T) \quad (53)$$

to $W(s)$.

This result can be extended directly to a weighted sum of Erlang terms, due to the linearity of the integration operator.

6.3. Appendix C

The set of equations (24) can be interpreted as follows. All poles of $W(s)$ are necessarily solutions of the equation

$$\left(1 - \frac{s}{\beta} \right)^K = \exp(-sT) \quad . \quad (54)$$

Remember that a root of this equation only qualifies as pole if $\text{Re}[s] < 0$.

We perform the substitution

$$z = 1 - \frac{s}{\beta} \quad , \quad (55)$$

with inverse

$$s = -\beta(z - 1) \quad . \quad (56)$$

Notice that the inequality $\text{Re}[s] < 0$, translates in $\text{Re}[z] < 1$ (since $\beta < 1$).

Using the fact that, if $a^K = b^K$, then $a = b \exp(j2\pi(k-1)/K)$ with $k=1, 2, \dots, K$, Eq. (54) leads to the K equations in z given in Eqs. (26). It can be proven that each of these Eqs. (26) has exactly 1 root ζ_k with $\text{Re}[z] < 1$, so that there are K different poles.

Additionally, it is easy to prove, by taking the modulus of both sides of Eqs. (26), that if $\text{Re}[z] < 1$, then $|z| < 1$, and that $|\zeta_1|$ is larger than any other $|\zeta_k|$. Moreover, each root can be found iteratively by using Eqs. (26) as an iterative scheme, starting from $z = 0$. For the equation with $k = 1$, this is obvious from a plot of the left and right hand side of eq. (26), but this can be extended to any value of k .

6.4. Appendix D

In this appendix we prove that set (22) together with equation (23) (comprising $K+1$ equations) contains one redundant equation and rewrite the set in the form (27).

Using Eqs. (24), Eq. (23) can be rewritten as

$$\sum_{j=1}^J a_j B(\alpha_j) = 1 \quad , \quad (57)$$

or

$$\sum_{j=1}^J a_j \left(\frac{\beta}{\beta - \alpha_j} \right)^K = 1 \quad . \quad (58)$$

The first equation of set (22) of equations can be rewritten as

$$1 - \sum_{j=1}^J a_j + \sum_{j=1}^J a_j \frac{\alpha_j}{\alpha_j - \beta} = 0 \quad (59)$$

while the others of this set can be rewritten as (where we multiplied the k -th equation with $(-\beta)^k$)

$$\sum_{j=1}^J a_j \frac{\alpha_j (-\beta)^k}{(\alpha_j - \beta)^{k+1}} = 0 \quad \forall k \in \{1, \dots, K-1\} \quad (60)$$

Summing the k first equations (i.e., Eq. (59) and the $k-1$ first of Eq. (60)) yields (using Horner's rule)

$$1 - \sum_{j=1}^J a_j + \sum_{j=1}^J a_j \frac{\alpha_j}{\alpha_j - \beta} \left[\frac{1 - \left(\frac{\beta}{\beta - \alpha_j} \right)^k}{1 - \left(\frac{\beta}{\beta - \alpha_j} \right)} \right] = 0 \quad , \quad (61)$$

which after a few trivial operations leads to

$$\sum_{j=1}^J a_j \left(\frac{\beta}{\beta - \alpha_j} \right)^k = 1 \quad k = 1, \dots, K \quad . \quad (62)$$

If $k=K$ this is exactly Eq. (58), showing that Eq. (23) is redundant.

Using the ζ_k defined in previous appendix and the fact that there are K poles, yields the set

$$\sum_{j=1}^K a_j \left(\frac{1}{\zeta_j} \right)^k = 1 \quad k = 1, \dots, K \quad . \quad (63)$$

This set of equations can be solved explicitly by noticing that the matrix associated with this set of equations (in the unknowns a_j/ζ_j) is a Vandermonde matrix. By using Cramer's rule and the explicit formula for Vandermonde determinants, the solution (27) yields.

Acronyms

CoV	Coefficient of Variation
DSL	Digital Subscriber Line
FIFO	First-in-First-out
FPS	First Person Shooter
HoL	Head-of-Line
IAT	Inter-Arrival Time
IETF	Internet Engineering Task Force
IP	Internet Protocol
i.i.d.	independent identically distributed
QoS	Quality of Service
RTT	Round Trip Time
TCP	Transport Control Protocol
TDF	Tail Distribution Function

UDP User Datagram Protocol

WFQ Weighted Fair Queuing

References

- [1] G. Armitage. “Sensitivity of quake3 players to network latency”. In ACM SIGCOMM Internet Measurement Workshop 2001, Berkeley, CA, USA, Nov. 2001.
- [2] G. Armitage. An experimental estimation of latency sensitivity in multiplayer quake 3. In 11th IEEE Int. Conf. on Networks (ICON 2003), Sydney, Australia, 2003.
- [3] M. Borella. Source models of network game traffic. In *Computer Communications*, vol. 23, no. 4, pages 403–410, 2000.
- [4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An Architecture for Differentiated Service”, RFC 2475, 1998.
- [5] R. Braden, D. Clark, S. Shenker, “Integrated Services in the Internet Architecture: and Overview”, RFC 1633, 1994.
- [6] F. Brichet, L. Massoulié, J.W. Roberts, “Stochastic Ordering and the Notion of Negligible CDV”, *Proceedings of ITC 15*, pp. 1433-1444, Washington (USA), 1997.
- [7] A. Charny, J.Y. Le Boudec, “Delay Bounds in a Network with Aggregate Scheduling”, *Proceedings of First COST 263 International Workshop*, pp. 1-13, QofIS2000, Berlin, Germany, 2000.
- [8] D.R. Cox. *Renewal Theory*. Methuen, London, 1962. (Chapter 6).
- [9] De Vleeschauwer, D., Petit, G.H., Wittevrongel, S., Steyaert, B. and Bruneel, H., 2000, “An Accurate Closed-Form Formula to Calculate the Dejittering Delay in Packetised Voice Transport”, *Proceedings of the IFIP-TC6 / European Commission International Conference NETWORKING 2000*, pp. 374-385, Paris.
- [10] M. Dick, O. Wellnitz, L. Wolf, “Analysis of Factors Affecting Players’ Performance and Perception in Multiplayer Games”, *Netgames ’05*, Hawthorne, New York, U.S.A., October 10-11, 2005.
- [11] J. Färber, *Network Game Traffic Modelling*, NetGames 2002, Braunschweig, Germany, April 16-17, 2002.
- [12] W. Feng, F. Chang, W. Feng, J. Walpole, “Provisioning On-Line Games: A Traffic Analysis of a Busy Counter-Strike Server”, *SIGCOMM Internet Measurement Workshop*, November 2002.
- [13] T. Henderson. Latency and user behaviour on a multiplayer game server. In *Proc. of the Third Int. COST264 Workshop on Networked Group Communication*, pages 1–13. Springer-Verlag, 2001.
- [14] T. Henderson and S. Bhatti. Networked games: a QOS-sensitive application for qos-insensitive users? In *Proc. of the ACM SIGCOMM workshop on Revisiting IP QoS*, pages 141–147. ACM Press, 2003.
- [15] L. Kleinrock, “*Queuing Systems Volume 1: Theory*”, John Wiley & Sons, New York, 1975.
- [16] T. Lang, G. Armitage, P. Branch, H.-Y. Choo, “A Synthetic Traffic Model for Half Life“, *Proceedings of the Australian Telecommunications Networks & Applications Conference 2003 (ATNAC 2003)*, Melbourne, Australia, December 2003
- [17] T. Lang, G. Armitage, “A Ns2 Model for the System Link Game Halo”, *Proceedings of the Australian Telecommunications Networks & Applications Conference 2003 (ATNAC 2003)*, Melbourne, Australia, December 2003
- [18] T. Lang, P. Branch, G. Armitage, “A Synthetic Traffic Model for Quake3”, *Proceedings of ACM SIGCHI ACE 2004*, Singapore, June 3-5, 2004.
- [19] M.R.H. Mandjes, J.C. Van Der Wal, R.E. Kooij, H.J.M. Bastiaansen, “End-to-end Delay models for Interactive Services on a Large-Scale IP Network”, *Proceedings of the 7th workshop on performance modelling and evaluation of ATM & IP networks (IFIP99)*, 1999.

- [20] L. Pantel, L. C. Wolf. On the impact of delay on real-time multiplayer games. In Proc. of the 12th int. workshop on network and operating systems support for digital audio and video, pages 23–29. ACM Press, 2002.
- [21] A. Parekh, R. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: The single node case”, IEEE/ACM Transactions on Networking, Vol. 1, pp. 344-357, 1993.
- [22] A. Parekh, R. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: The multiple node case”, IEEE/ACM Transactions on Networking, Vol. 2, pp. 137-150, 1994.
- [23] P. Quax, P. Monsieurs, W. Lamotte, D. De Vleeschauwer, N. Degrande, “Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a First Person Shooter Game”, Proceedings of the Third Workshop on Network and System Support for Games (NetGames04), pp. 152-156, Portland (OR), August 30 – September 3, 2004.
- [24] K. Ramanan, J. Cao: A Poisson Limit for Buffer Overflow Probabilities. Proc. INFOCOM 2002, pp. 497–505, 2002.
- [25] J. Roberts, U. Mocci, J. Virtamo, Broadband Network Teletraffic - Performance Evaluation and Design of Broadband Multiservice Networks: Final Report of Action COST 242. Springer, Berlin, 1996.
- [26] N. Sheldon, E. Girard, S. Borg, M. Claypool, E. Agu, “The effect of latency on user performance in warcraft 3”, Proc. of the 2nd workshop on Network and system support for games, pages 3–14. ACM Press, 2003.
- [27] Ubiocom Inc., “OPScore, or Online Playability Score: A Metric for Playability of Online Games with Network Impairments”, <http://www.ubicom.com/pdfs/whitepapers/IP3K-DWP-OPSCORE-10.pdf>, March 2005.
- [28] <http://mathworld.wolfram.com/ExtremeValueDistribution.html>