Modeling Ping Times in First Person Shooter Games

N. Degrande[1],
D. De Vleeschauwer[1,2]
[1]Alcatel Bell, Network Strategy Group
Copernicuslaan 50,
B-2018 Antwerpen, Belgium
[2]University Ghent, TELIN, SMACS
Sint_Pietersnieuwstraat 41,
B-9000 Gent, Belgium
{natalie.degrande|
danny.de_vleeschauwer}
@alcatel.be

R.E. Kooij[3,4]
[3]TNO Information and
Communication Technology
Brasserplein 2, P.O. Box 5050,
2600 GB Delft, the Netherlands
[4]Delft University of Technology
Dept. of Electrical Engineering,
Mathematics and Computer Science
Mekelweg 4,
2628 CD Delft, the Netherlands
r.e.kooij@telecom.tno.nl

M.R.H. Mandjes[5,6]
[5]CWI
P.O. Box 94079,
1090 GB Amsterdam, the Netherlands
[6]Korteweg-de Vries Institute
for Mathematics,
University of Amsterdam,
Plantage Muidergracht 24,
1018 TV Amsterdam, the Netherlands
michel.mandjes@cwi.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 therefore gamers do not want to connect to servers with a long 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 conclusion still remains valid, additional experiments could refine the detailed quantitative results.

CoNEXT'2006, December 4-7,2006, Lisbon, Portugal.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous

General Terms

Measurement, Performance, Theory

Keywords

On-line games, Round Trip Time, queuing model, analysis

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, videophony, 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 QoS 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 (WFQ) 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 packet 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 RTT 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 the motion vectors (resulting from that gamer's action) 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 (e.g., via a WFQ scheduler) 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 for 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 the decomposition of the upstream and downstream part. Section 4 uses the developed queuing model to assess the RTT typical for games played over a representative 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 discuss the characteristics that were derived in literature and confirm these by analyzing a set of new measurement data. 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. The nature of the observed traffic flows suggests that a networked game works according to the following principles. 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 new measurement data is compared to the published ones. Finally, we summarize the traffic source model that is used further on in this paper.

2.1 Related work

Some papers report 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 packets arrived in bursts with an inter-burst time that was slightly varying. In each burst, the server generates one packet for every active client. The modus of the 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 $f(x)$ of the extreme distribution, respectively the associated cumulative distribution $F(x)$, with parameters a and b, is given by [28]

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

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 later in this section; 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 byte and a CoV of 0.74. Again, he approximates this (using least square fitting of the probability density function with the histogram) 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 byte and a CoV of 0.12, is modeled by the extreme distribution Ext(80,5.7).

Note that Färber also mentions that shifted lognormal and shifted Weibull distributions lead to acceptable fits to the data as well.

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 (41 ms) 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 byte. Here normal and lognormal distributions lead to equally good fits. In general, the authors found that the traffic pattern was only affected by the map that was played (server-to-client packet sizes in particular) and the graphic rendering software of the client computer (in particular, inter-arrival times of the client-to-server traffic). Moreover, their model is roughly in line with the model proposed by Färber.

In another paper [17], Lang *et al*. present a model for the Xbox System Link game Halo. The server-to-client inter-burst time is modeled deterministically (40 ms), 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 (with a fixed size of 72 byte) are sent every 201 ms; 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 the traffic is strongly periodic, and that the traffic pattern is influenced by the number of players (specifically the server-to-client and client-to-server packet sizes) and the client Xbox hardware (particularly, the 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.

The main aim of [23] was to study the impact of delay and jitter on game play and subjective user experience. An experiment was set up in which 12 players combated against each other in the FPS game Unreal Tournament 2003. As a side result the traffic was logged during this LAN party. Part of this traffic will be used here to verify the traffic characteristics from literature that were summarized above. 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 between bursts.

The traffic trace of 6 minutes consists of all traffic between the dedicated server and the 12 players in both directions. Similar to the papers mentioned above, we found that the traffic from the server to the clients consists of traffic bursts, which arrive with almost constant inter-arrival times (the average inter-burst time is 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 burst after that peculiar 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 byte 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 in the papers discussed 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 byte, with a CoV of 0.19. For the traffic from the clients to the server, we found (neglecting a few outliers) a mean packet size of 73 byte and a CoV of 0.06, the inter-arrival time was 28 ms with a CoV of 0.30. These findings are displayed in Table 1.

Table 1: Overview of characteristics of traffic generated during a FPS gaming session on a LAN. Packet sizes are in byte, inter-burst times in ms, burst sizes in byte.

	Server-to-client		Client-to-server	
	Mean	CoV	Mean	CoV
Packet size	154	0.28	73	0.06l
Inter-burst time	47	0.07	28	0.30
Burst size	1852	0.19	-	-

2.2 Traffic Source Models

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

2.2.1 Client traffic model

Based on the findings in Section 2.1, 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 periodic stream 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).

2.2.2 Server traffic model

Following Section 2.1, 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 fitting the theoretical to the experimental central moments), we prefer to fit the tail of the distribution, since this tail dominates also the tail of the corresponding queue occupation (and hence, the tail of the queuing delay). 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 1). In order to determine the order K of the Erlang distribution, we can focus either on fitting the CoV (which boils down to fitting the standard deviation), or on the tail of the distribution. Following the former approach, i.e., fitting the CoV and noticing from Table 1 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 the experiment as well as of the tail distribution functions of Erlang distributions for $K = 15$, 20 and 25, respectively.

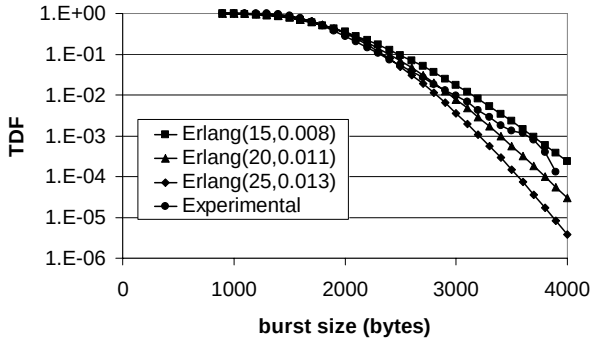

Figure 1: Tail distribution function of the measured burst sizes and Erlang tail distributions of different order for which the mean was already fit to the measured mean.

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 2.1), 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 and the analysis of the new measurement data, 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 RTT (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.

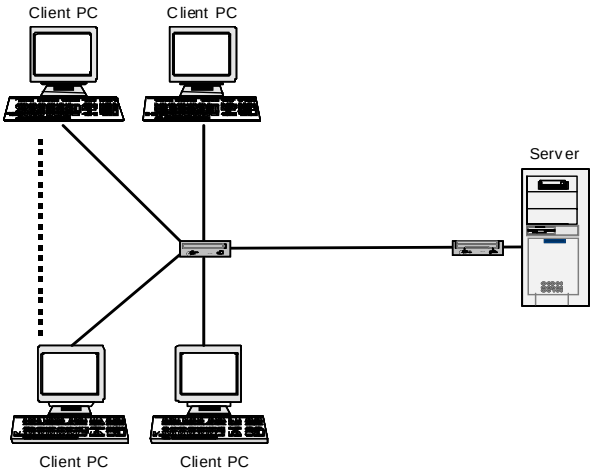

Figure 2: Client-server architecture for interactive gaming.

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 2.2.2) 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.

In this paper we merely state the probabilistic laws that are needed for the computation of the stochastic part of the RTT. The derivation of these laws is out of scope for this paper, but can be found in [8].

3.1 Upstream: M/D/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 it does not provide insight and is relatively computationally demanding. However, it can be argued that we can do with simpler formulas. In fact, the use of M/D/1 queues for modeling the upstream queuing delay is mathematically justified in [8]. Notice that, in [25], empirical observations indicate that the M/D/1 approximation works very well, particularly for relatively small load (while there are serious deviations for a load above, say, 90%). The regime with relatively low load is the regime we are interested in.

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

$$D_u(s) \approx 1 - \rho_u + \rho_u \frac{\gamma}{\gamma - s}, \qquad (2)$$

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 each gamer. The burst sizes are modeled as independent, identically distributed (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 (joint) reserved bit pipe, the N*D/G/1 queuing model applies where $G=\Sigma E_K$ (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 this paper we only consider the case of 1 gaming server.

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 be the remaining work (expressed in [s]) in the system just before the n-th arrival instant. Notice that the variable w_n is also the delay seen by the n-th burst. In addition, b_n denotes the amount of work arriving (expressed in [s], i.e., the burst size divided by the reserved gaming rate) at the n-th arrival instant, which are i.i.d. with an Erlang distribution of order K as marginal

distribution. Finally, T (expressed in [s]) is the amount of work that can be performed between two arrival instants.

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

$$W(s) = E[e^{sw_n}] \quad , \tag{3}$$

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

$$B(s) = \left(\frac{\beta}{\beta - s}\right)^K \quad , \tag{4}$$

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

In [8] an explicit expression is derived for $W(s)$:

$$W(s) = \left(1 - \sum_{k=1}^{K} a_k\right) + \sum_{k=1}^{K} a_k \frac{\alpha_k}{\alpha_k - s} \quad , \tag{5}$$

where each pole is given by

$$\alpha_k = -\frac{K(\zeta_k - 1)}{\bar{b}} \ , \ \forall k \in \{1,...,K\} \quad , \tag{6}$$

where ζ_k is the unique solution of

$$z = \exp\left(\frac{z-1}{\rho_d} + j\frac{2\pi(k-1)}{K}\right), \quad \forall k \in \{1,...,K\}, \tag{7}$$

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

Finally, the weights a_j can be shown to satisfy

$$a_j = (\zeta_j)^K \prod_{\substack{k=1 \\ k \neq j}}^{K} \frac{\zeta_k - 1}{\zeta_k - \zeta_j}, \quad j = 1,...,K \quad . \tag{8}$$

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

$$d_{n,t} = w_n + p_{n,t} \quad . \tag{9}$$

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

$$D_d(s) = W(s)P(s) \quad . \tag{10}$$

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 probability density function $p_u(u_{n,t})$ 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

$$P(s) = \int_0^1 d\tau \left(\frac{\beta}{\beta - s\tau}\right)^K p_u(\tau) \quad . \tag{11}$$

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

We assume that from burst to burst the packet can reside anywhere in the burst, i.e., that $p_u(u_{n,t})$ is the uniform distribution. In this case, the integral defining $P(s)$ can be calculated (by identifying a primitive function of the integrand) and satisfies, for K>1:

$$P(s) = \frac{\beta}{(K-1)s}\left[\left(\frac{\beta}{\beta - s}\right)^{K-1} - 1\right] \quad . \tag{12}$$

Notice that this can be interpreted as the residual lifetime of an Erlang variable of order (K-1) [15].

It can be shown using Horner's rule that $P(s)$ can be written as a weighted sum of Erlang terms:

$$P(s) = \frac{1}{(K-1)} \sum_{k=0}^{K-2} \left(\frac{\beta}{\beta - s}\right)^{K-1-k} \quad . \tag{13}$$

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 function of the total delay is a product $D_u(s)W(s)P(s)$. As can be seen from eqs. (2), (5) and (13) respectively, each of these factors can be (approximately) written as a sum of Erlang terms, such that, with the technique explained in [8], we can write this product as a sum of Erlang terms:

$$\begin{aligned} &(W(\gamma)P(\gamma)\rho_u)\left(\frac{\gamma}{\gamma - s}\right) + \sum_{j=1}^{K} \left(D_u(\alpha_j)P(\alpha_j)a_j\right)\left(\frac{\alpha_j}{\alpha_j - s}\right) \\ &+ \frac{1}{(K-1)} \sum_{k=0}^{K-2} \left(\sum_{l=0}^{k} (-\beta)^l \sum_{m=0}^{l} \frac{D_u^{(m)}(\beta)W^{(l-m)}(\beta)}{m!(l-m)!}\right)\left(\frac{\beta}{\beta - s}\right)^{K-1-k} \end{aligned} \tag{14}$$

which is straightforward to invert.

In the next section we will use eq. (14) to determine quantile values for the RTT.

4. NUMERICAL RESULTS

In this section we assess the impact of several parameters on the RTT performance for FPS 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 gaming load* (defined as the average amount of gaming traffic divided by the capacity reserved for the gaming traffic 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 CoV of the size of the burst of packets from the server to the clients*; for given load the lower the CoV, the lower the queuing delay; note that in our model the burst sizes are assumed to follow an Erlang distribution of order K therefore CoV = $1/\sqrt{K}$.

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 kb/s, the downlink access bandwidth is fixed at R_{down} = 1024 kb/s. In addition we suppose that the inter-arrival time T of bursts of packets from the server equals the inter-arrival time D of packets from the clients (per client). We will fix T at the values 40 ms and 60 ms. This choice is motivated by the findings of Section 2.

With C we denote the capacity reserved on the aggregation link (i.e., the link between fan-out point and the gaming server) for gaming traffic. As explained in Section 1, this capacity C is that part of the aggregation link that the WFQ scheduler reserves for gaming traffic. We also assume that the other traffic classes always sends traffic such that the WFQ scheduler cannot borrow capacity from other classes. We choose C = 5000 kb/s.

For the size of packets sent from the server, denoted by P_S, we take the values 125 byte, 100 byte and 75 byte. In order to assess the impact of the CoV of the burst sizes we vary the order K of the Erlang distribution (representing the variation of the size in the bursts of packets from the server); the following values are considered: K = 2, 9 or 20.

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

$$\rho_d = \frac{8NP_s}{TC} \quad , \qquad (15)$$

where N denotes the number of active gamers and P_S given in byte, T in ms and C in kb/s. A similar formula applies for the uplink load. In the remainder of this section we vary the downlink load by varying the number of active gamers N.

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 as a function of the load ρ_d for the case P_S = 125 byte, T = 60 ms, for several values of the Erlang order K. For this set of parameter 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. As explained above this packet position delay has the same distribution as the residual lifetime of an Erlang distribution of order (K-1) and as such 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 ρ_d = 1.

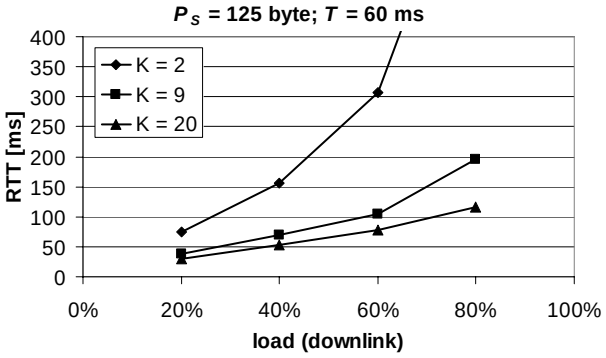

Figure 3: Impact of Erlang order *K* on the RTT.

It is obvious from Figure 3 that the order K of the Erlang distribution, and hence the CoV of the burst sizes, 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 values, 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.

The fact that for low loads the RTT (virtually equal to the (dominant) packet position delay in these cases) is still large is tightly connected to our assumption that the scheduler follows the WFQ discipline. Indeed, after a certain packet in the burst is processed, the WFQ scheduler visits queues of other traffic classes (which we assumed to nearly always contain packets) before it returns relatively fast (a time of the order of T/N) to the gaming queue, such that overall the gaming queue gets drained by a rate C (which is a small fraction of the link capacity). Consider, however, another (burst-non-preemptive) scheduler: it processes a

complete burst of packets before visiting other queues, spends a relatively long time (just shorter than T) on queues of other traffic classes, all this in such a way that in the long run the gaming queue is drained by a rate C as well. Such a scheduler would seriously reduce the packet position delay component of the RTT and would result in much lower quantile values for low loads. This scheduler is a topic for future research.

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.

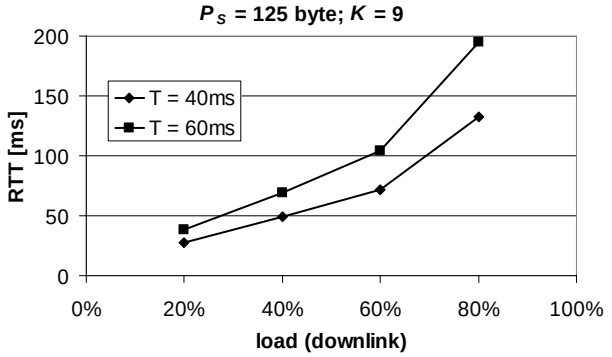

Figure 4: Impact of the inter-arrival time on the RTT.

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 (of which C is just a fraction).

The above results can be used for dimensioning purposes. Consider for instance the case $P_S = 125$ byte, $K = 9$, $T = 40$ ms. Then in order to realize a maximum RTT of 50 ms (corresponding with excellent game play, according to [11]) the gaming bit pipe allows a load (in the downlink direction) of at most 40%, see Figure 4. It is obvious that the maximum allowable load $\rho_{\max}$ is very sensitive with respect to the parameter K and hence with respect to the CoV of the burst sizes. In fact, for $P_S = 125$ byte, $T = 40$ ms the maximum allowable load is about 20% for $K = 2$, while for $K = 20$ the threshold $\rho_{\max}$ is around 60%. This can be deduced from Figure 3 (and the fact that the RTT is proportional to the inter-arrival time T).

Using eq. (15) the dimensioning rule leads to a maximum number of on-line gamers $N_{\max}$ of

$$N_{\max} = \frac{TC\rho_{\max}}{8P_s} \quad . \tag{16}$$

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 (FPS) 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 RTT 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 gaming 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 CoV of burst sizes of packets sent from server to clients; we observed a remarkably strong impact of this CoV on the RTT quantiles. In view of this effect, we expect that it would pay off to more accurately determine the tail behavior of burst sizes by tracing packets in real-life FPS games on a considerably larger scale.

6. ACKNOWLEDGMENTS

This work was partly carried out within the framework of the project CHAMP sponsored by the Flemish Institute for the promotion of Scientific and Technological Research in the Industry (IWT).

7. REFERENCES

[1] Armitage, G., “Sensitivity of quake3 players to network latency”. In ACM SIGCOMM Internet Measurement Workshop 2001, Berkeley (CA), Nov. 2001.

[2] Armitage, G ., “An experimental estimation of latency sensitivity in multiplayer quake 3”. In Proceedings of the 11th IEEE Int. Conf. on Networks (ICON 2003), Sydney (Australia), 2003.

[3] Borella, M., “Source models of network game traffic”. In Computer Communications, vol. 23, no. 4, pp. 403–410, 2000.

[4] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and Weiss, W., “An Architecture for Differentiated Service”, IETF RFC2475, 1998.

[5] Braden, R., Clark, D. and Shenker, S., “Integrated Services in the Internet Architecture: and Overview”, IETF RFC1633, 1994.

[6] Brichet, F., Massoulié, L. and Roberts, J.W., “Stochastic Ordering and the Notion of Negligible CDV”. In Proceedings of ITC 15, pp. 1433-1444, Washington (DC), 1997.

[7] Charny, A. and Le Boudec, J.-Y., “Delay Bounds in a Network with Aggregate Scheduling”. In Proceedings of the First COST 263 International Workshop, pp. 1-13, QofIS 2000, Berlin (Germany), 2000.

[8] Degrande, N., De Vleeschauwer, D., Kooij, R.E. and Mandjes, M.R.H, “Modeling Ping times in First Person Shooter Games”, CWI rapport PNA-R0608, ISSN 1386-3711, June 2006.

[9] De Vleeschauwer, D., Petit, G.H., Wittevrongel, S., Steyaert, B. and Bruneel, H., “An Accurate Closed-Form Formula to Calculate the Dejittering Delay in Packetised Voice Transport”. In Proceedings of the IFIP-TC6 / European Commission International Conference NETWORKING2000, pp. 374-385, Paris (France), 2000.

[10] Dick, M., Wellnitz, O. and Wolf, L., “Analysis of Factors Affecting Players’ Performance and Perception in Multiplayer Games”. In Proceedings of Netgames ’05, Hawthorne, New York, U.S.A., October 10-11, 2005.

[11] Färber, J., “Network Game Traffic Modelling”. In Proceedings of NetGames 2002, Braunschweig, Germany, April 16-17, 2002.

[12] Feng, W., Chang, F., Feng, W. and Walpole, J., “Provisioning On-Line Games: A Traffic Analysis of a Busy Counter-Strike Server”. In Proceedings of SIGCOMM Internet Measurement Workshop, November 2002.

[13] Henderson, T., “Latency and user behaviour on a multiplayer game server”. In Proceedings of the Third Int. COST264 Workshop on Networked Group Communication, pages 1–13. Springer-Verlag, 2001.

[14] Henderson, T. and Bhatti, S., “Networked games: a QoS-sensitive application for QoS-insensitive users?” In Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS, pages 141–147. ACM Press, 2003.

[15] Kleinrock, L., 1975, “Queuing Systems Volume 1: Theory”, John Wiley & Sons, New York.

[16] Lang, T., Armitage, G., Branch, P. and Choo, H.-Y., “A Synthetic Traffic Model for Half Life“. In Proceedings of the Australian Telecommunications Networks & Applications Conference 2003 (ATNAC 2003), Melbourne, Australia, December 2003.

[17] Lang, T. and Armitage, G., “A Ns2 Model for the System Link Game Halo”, In Proceedings of the Australian Telecommunications Networks & Applications Conference 2003 (ATNAC 2003), Melbourne (Australia), December 2003.

[18] Lang, T., Branch, P. and Armitage, G., “A Synthetic Traffic Model for Quake3”, In Proceedings of ACM SIGCHI ACE 2004, Singapore, June 3-5, 2004.

[19] Mandjes, M.R.H., Van Der Wal, J.C., Kooij, R.E. and Bastiaansen, H.J.M., 1999, “End-to-end Delay models for Interactive Services on a Large-Scale IP Network”, In Proceedings of the 7th workshop on performance modelling and evaluation of ATM & IP networks (IFIP99), 1999.

[20] Pantel, L. and Wolf, L. C., “On the impact of delay on real-time multiplayer games”. In Proceedings of the 12th Int. workshop on network and operating systems support for digital audio and video, pp. 23–29, ACM Press, 2002.

[21] Parekh, A. and Gallager, R., “A generalized processor sharing approach to flow control in integrated services networks: The single node case”. In IEEE/ACM Transactions on Networking, vol. 1, pp. 344-357, 1993.

[22] Parekh, A. and Gallager, R., “A generalized processor sharing approach to flow control in integrated services networks: The multiple node case”. In IEEE/ACM Transactions on Networking, vol. 2, pp. 137-150, 1994.

[23] Quax, P., Monsieurs, P., Lamotte, W., De Vleeschauwer, D. and Degrande, N., “Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a First Person Shooter Game”. In Proceedings of the Third Workshop on Network and System Support for Games (NetGames04), pp. 152-156, Portland (OR), August 30 – September 3, 2004.

[24] Ramanan, K. and Cao, J., “A Poisson Limit for Buffer Overflow Probabilities”. In Proceedings INFOCOM 2002, pp. 497–505, 2002.

[25] Roberts, J., Mocci, U. and Virtamo, J., “Broadband Network Teletraffic - Performance Evaluation and Design of Broadband Multiservice Networks: Final Report of Action COST 242”. Springer, Berlin, 1996.

[26] Sheldon N., Girard, E., Borg, S., Claypool, M. and Agu, E., “The effect of latency on user performance in warcraft 3”. In Proceedings of the 2nd workshop on Network and system support for games, pages 3–14. ACM Press, 2003.

[27] Ubicom 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