**A simple bound on the game server computing power**

*By: Lam Ngok*

**Objective**
In this preliminary study, we employed some simple results from the queueing theory literature to derive a bound on the computing power of an online game server. The objective of this study is to conduct a simple study of a real online system using results from the queueing literature. The purpose of this study is to get myself familiar with the tools from the queueing theory in attacking real world networking problem.

**Introduction**
The number of online game players had almost doubled from 1999 to 2003 (grew from 38 million to 68 million). Research companies like the Forrester Research had predicted that 24% of the revenues of the gaming industry will be from online games by the end of this decade. Right now, about 40% of all the game players play online games, they spend an average of 7 hours per week on online games in 2004[1]. It is expected that by 2008, online gaming traffics will contribute to 33% of the backbone traffic[2]. The gaming traffic is a relatively new topic in the research community, only limited number of literature is available.

In the literature, online games are classified as computer or console games that being played over the networks. Two or more players are often involved in playing the games. There are three categories of online games, which are:

1. First Person Shooter (FPS) games,
2. Massive Multiplayer Online Role Playing Games (MMORPG),
3. Real Time Strategy (RTS) games.

First Person Shooter games are fast-paced first person action games, typically there consists of tens of users connected simultaneously, examples of this type of games are Quake and Doom. Massive Multiplayer Online Role Playing Games are games that involve the control or development of an in-game character, generally there are thousands of users connected at the same time to play the game. Examples of this kind of games are Everquest, and Ultima online. Real Time strategy games usually
concerns with the building, and the upgrading of virtual cities, players are in
cmand of the development of these virtual cities. This type of games typically
involves tens of players connected simultaneously to the network, examples are
Warcraft, and Age Of Empires.

The majority of online games utilize personal computers or game boxes (like
PS2, XBOX) as platforms, these platforms usually connect to the network through the
Ethernets. Though games offered through wireless connections are also available, but
this is still an immature market with low penetration.[3]

A larger number of current literature on online games concern with the software
designing aspects of the online games. Major issues are state consistence, scalability,
cheating prevention, price charging architectures, etc [5-12]. But there are also a
number of papers that deal with the modeling of the gaming traffics, and the QoS
requirements of the online games[13-21].

Four major QoS parameters are identified in the literature; they are the throughputs,
the jitters, the losses and the delays. Most of today's online games require only slow
connections to fulfill the data throughput requirement, for example, RTS games like
WarCraft III only needs ~8kbps of throughput on an average, which can be satisfied
even by slow dialup connections. The impact of jitters is not clear to the research
community now, very few researches have been done on this parameter, apparently
only some subjective quality assessment studies are available. The impact of packet
losses on the performance is not as large as imagined, and they do not seem to affect
the online players too much, because many games are designed to deal with losses.
Most of the studies suggested that delay is the most important factor that affects the
online gaming experience, and it is also one of the focuses of current gaming research.
Different types of games require different delay requirements. For first person shooter
games, delays less than 50ms will give excellent responses to the players, and delays
in the range of 50-100ms will still provide good responses, players are likely to notice
the delays when the they fall into the range 100ms-150ms, and the users will be
significantly affected when the delays are in the range of 150ms-200ms, anything
larger than 200ms will be intolerable to the players. For real time strategy games, the
delay requirement is looser and usually it is acceptable to the users for even 500ms of
delays. For MMORPG games, the maximum average delays is around 100ms-150ms.

Different gaming architectures will give different delay characteristics. There are
three types software architectures for the online games[11]. The first type of
architecture is the peer-to-peer architecture as shown in the following figure:

In this kind of structure, the players are connected in a pure peer-to-peer manner.

The second type of architecture is the client/server type as shown above. In this type of architecture, players are connected through a centralized server.

The third type of software architecture is the network of servers architecture. In this design every client is attached to a single server and the servers are connected in a peer-to-peer manner. This architecture is usually good for supporting larger number of players, as it is easier to scale up with the number of players.

The Peer-to-Peer software architecture is the simplest approach, but it does not work well for large networks, as the number of message exchanges for a conversation will be in the order of \( n^2 \), where \( n \) is the number of hosts. There are also the issues of state consistence and cheating prevention for the Peer-to-Peer architecture. Centralized servers and network of servers can scale up better, the number of required messages for a conversation will be in the order of \( n \). They are also the better solutions for cheating prevention and state consistence issues. Most large scale online games use the client/server or the network of servers approach. We can easily see that the
computing powers of the server machines will be one of the major factors that affects the quality of services for this two types of software structures.

The study

Surprisingly, there does not seem to have any queueing analysis on the internet games. Therefore we performed an interesting queueing analysis based on the client/server architecture.

For the client/server architecture, the delay experienced by a user consists of two major parts, the first part is the network delay part due to queueing time and transmitting time at the intermediate routers. The second part is the server delays due to the processing time of central server.

This is depicted in the picture above, suppose a host located at the left hand side of the figure (not shown) wishes to communicate with a host at the right (not shown). The total delay experienced by this host will be the sum $T + D_{d1} + D_{d2} + D_{u1} + D_{u2}$, where $T$ is the processing time at the server and $D_{d1}$, $D_{d2}$, $D_{u1}$, $D_{u2}$ are the downstream and upstream delays of the network. We wish to derive the minimum computing power for the server so as to guarantee QoS when we are given the network conditions (i.e. network delays).

Assume the arrival process is Poisson, also assume the service time for the arrivals has a general distribution, then we can use the $M/G/1$ model to study this system. Suppose we need the maximum average delay to be less than $D$ for the game to be playable. Then we will have the following inequality:

$$E[T] + D_{d1} + D_{d2} + D_{u1} + D_{u2} < D$$

$$\Rightarrow E[T] < D'$$

where $D' = D - D_{d1} - D_{d2} - D_{u1} - D_{u2}$. Since this is a $M/G/1$ system, we can rewrite (1) as:
\[ \frac{\lambda}{\mu} E[X^2]/2(1 - \lambda/\mu) + 1/\mu < D' \]  

----- (2)

where \( \lambda \) is the aggregated outgoing data rate for the player, \( \mu \) is the processing rate for the server, \( X \) is the processing time and \( E[X]=1/\mu \). For the player to receive satisfactory gaming performance, inequality (2) must be satisfied. As a result we can solve the required server computing power (\( \mu \)) by rewriting (2) as follows:

\[
( \lambda \ E[X^2] - 2 \ D' ) \ \mu^2 + (2 \ D' \ \lambda + 2) \ \mu - 2 \lambda < 0
\]

----- (3)

We can solve (3) in two ways (will show both of them). The first method involves rewriting (3). We know \( E[X^2] \) can be expressed as \( \text{Var}(x) + E^2[x] \) and we also know \( \mu=1/E[x] \), therefore we can write:

\[
\mu^2 \ E[x^2] = (\text{Var}(x) + E^2[x])/E^2[x]
\]

\[= Cb^2 + 1 \]  

----- (4)

where \( Cb \) is the “coefficient of variation for service time”.

Plug (4) into (3) and rewriting we have:

\[ 2D' \ \mu^2 - (2D' \ \lambda + 2) \ \mu - \lambda (Cb^2 - 1) > 0 \]

----- (5)

solving (5) we have:

\[ \mu > \lambda + 1/D' + F/D' \]  

or \[ \mu < \lambda + 1/D' - F/D' \]  

----- (6)

where \( F=((D' \ \lambda)^2 + 2D' \ \lambda \ Cb^2 + 1)^{1/2} \), but in order to guarantee stability, we need \( \mu > \lambda \). However, if \( \mu < \lambda + 1/D' - F/D' \), then we have:

\[
\mu < \lambda + 1/D' - F/D'
\]

\[ = \lambda + (1 - ((D' \ \lambda)^2 + 2D' \ \lambda \ Cb^2 + 1)^{1/2})/D'
\]

\[ = \lambda + \text{negative number}
\]

\[ < \lambda \]

Therefore we need to reject \( \mu < \lambda + 1/D' - F/D' \), our solution now becomes:

\[ \mu > \lambda + 1/D' + F/D' \]
Denote by C the computing power of the server, assume the data sent by the client has an average size of E[s] bits, and the processing time is linear with respect to the data size, then we have:

\[
1/ \mu = E[s]/C
\]

\[
\text{-----(8)}
\]

plug (8) into (7) we have:

\[
E[s]/C < 1/(\lambda + 1/D'+F/D')
\]

\[
=> C > E[s](\lambda + 1/D'+F/D')
\]

Therefore the minimum computing power needed for the server is E[s](\lambda + 1/D'+F/D'), where \(F = ((D' \lambda)^2 + 2D' \lambda Cb^2 + 1)^{1/2}\).

This solution involves the coefficient of variation for service time Cb. We know that computers process data in a deterministic way, the service time is stochastic just because the data flows from the client is stochastic. Therefore we can consider Cb as a term that involves the first and second moments of the client data flow. In order to gain more insights we also solve inequality (3) alternatively as follows:

First let's assume \((\lambda E[X^2] - 2D')\) is positive, then we have:

\[
( -(2D' \lambda +2) +H )/2( \lambda E[X^2] -2D' ) > \mu > ( -(2D' \lambda +2) -H )/2( \lambda E[X^2] -2D' )
\]

where \(H = ((2D' \lambda +2)^2 +8 \lambda ( \lambda E[X^2] -2D' ))^{1/2}\)

but detailed calculations show that \(( -(2D' \lambda +2) +H )/2( \lambda E[X^2] -2D' ) < \lambda\), therefore \(\mu\) is also smaller than \(\lambda\), and the stability condition can not be satisfied. As a result there is not solution if \((\lambda E[X^2] - 2D')\) is positive! (i.e. no solution if \(\lambda E[X^2]/2 > D'\))

Now let's assume \((\lambda E[X^2] - 2D')\) is negative, then we have:

\[
( -(2D' \lambda +2) +H )/2( \lambda E[X^2] -2D' ) > \mu
\]

or

\[
( -(2D' \lambda +2) -H )/2( \lambda E[X^2] -2D' ) < \mu
\]
simple (but somewhat tedious) calculations show that \(( -(2D' \lambda + 2) + H )/2( \lambda E[X^2] - 2D')\) is smaller than \(\lambda\), but \(( -(2D' \lambda + 2) - H )/2( \lambda E[X^2] - 2D')\) is larger than \(\lambda\). Therefore we now have the inequality:

\[
( -(2D' \lambda + 2) - H )/2( \lambda E[X^2] - 2D') < \mu
\]

Using (9) we can find that the minimum computing power as:

\[
E[s] ( -(2D' \lambda + 2) - H )/2( \lambda E[X^2] - 2D').
\]

After some simplifications, this result is effectively the same as the one we found previously.

**Conclusion**

In this study, we derived the server computing power as an expression of the network delays, incoming data rates, incoming packet sizes, first and second moments of the incoming traffic. We used two different approaches in obtaining the final solution and they are the same for both approaches, but in the second approach we gained further insight by realizing that solution does not exist if \(\lambda E[X^2]/2 > D'\). This inequality can be used to guide the design of client software so that the product \(\lambda E[X^2]\) will not be too large.

Using the results, we can calculate the exact computing power required to guarantee QoS. More precise result may be obtained by using the G/G/1 model. With that, it should be useful for gaming company to decide the hardware requirements for the servers and guide them in the development of their gaming networks. Moreover by combining the equation with the utility function of the players, we may derive a pricing model to charge different delays at different prices.
References
[17] The effects of latency on online Madden NFL football, James Nichols et al.
[18] An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3, GrenVille J.A.
[22] internet-based games, R. Michael Young.