Lightweight QoS-Support for Networked Mobile Gaming

Marcel Busse
Praktische Informatik IV
University of Mannheim
L15, 16
D-68161 Mannheim
Germany
busse@informatik.uni-mannheim.de

Bernd Lamparter
NEC Europe Ltd.
Network Laboratories
Kurfürsten-Anlage 36
D-69115 Heidelberg
Germany
bernd.lamparter@ccrle.nec.de

Martin Mauve
Institut für Informatik
University of Düsseldorf
Universitätsstraße 1
D-40225 Düsseldorf
Germany
mauve@cs.uni-duesseldorf.de

Wolfgang Effelsberg
Praktische Informatik IV
University of Mannheim
L15, 16
D-68161 Mannheim
Germany
effelsberg@informatik.uni-mannheim.de

ABSTRACT

In this paper, we present an approach to provide Quality of Service (QoS) for networked mobile gaming. In order to examine the QoS requirements of mobile games, we ported a simple real-time game called GAV (GPL Arcade Volleyball) to a PDA and performed several traffic measurements over both GPRS and UMTS networks. We show that due to high end-to-end delay and delay jitter, real-time games are not supported by GPRS. While UMTS improves both delay and jitter, it still does not match the requirements of real-time games. The key reason for this problem is that overprovisioning, as it is used to allow real-time games in the Internet, is very expensive in mobile networks. At the same time, QoS classes for mobile networks are not tailored to real-time games. In order to reduce delay and jitter for this application class, while still accounting for the very bursty nature of real-time game flows, we propose to use a combination of statistical multiplexing and QoS guarantees. The general idea is to aggregate multiple game flows and perform reservation for that aggregate. As a theoretical background, we use a queuing system based model. Through simulation of a sample network with the traffic data generated by GAV, we validate our assumptions and demonstrate the performance and characteristics of our approach.

Keywords

Mobile Gaming, QoS-Support, Resource Reservation, Admission Control, Queuing Systems, Flow Aggregation

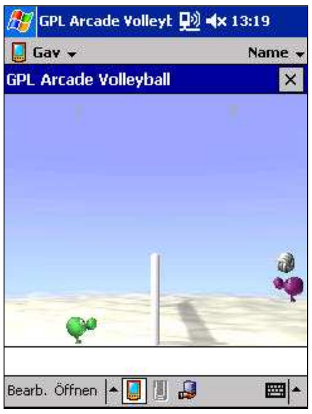
1. INTRODUCTION

With the introduction of 3G wireless systems like UMTS, many mobile real-time applications will become available. Besides audio and video applications, market researchers expect high revenues from online games played over wireless infrastructures. However, real-time applications require very low end-to-end delays, so that QoS aspects have to be taken into account. Considering the most popular networked real-time games in the (wired) Internet, three main game genres can be classified: First Person Shooter (FPS), Real Time Strategy (RTS), and Massive Multiplayer Online Role Playing Games (MMORPG). Games of the same genre require quite similar delay bounds, e.g. 150 ms for FPS games like *Halflife* [8] and 300 ms for RTS games such as *Age of Empire* [24].

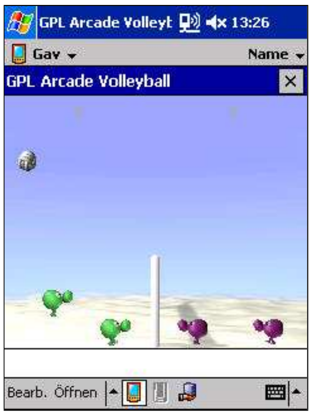
To understand the QoS requirements of mobile real-time games, we ported a very simple real-time multiplayer networked game called GPL Arcade Volleyball (GAV) [23] to a PDA. We then used this game to test the suitability of today's GPRS networks. As we expected, playing GAV was impossible due to high end-to-end delay and delay variations. Even though GPRS specifies some QoS aspects [1], network providers do not regularly implement them, resulting in poor performance for networked games. In a second set of experiments, we investigated playing GAV over an UMTS network which has just been launched in Germany. The results indicate a significant improvement over GPRS but also shows that even UMTS is not able to handle real-time games. While UMTS is currently planned to support two real-time QoS classes [2], none of them covers the QoS requirements and the very bursty nature of networked game traffic very well. Overprovisioning – in order to avoid having to deal with QoS aspects – on the other hand is quite expensive in mobile environments and currently not done on a sufficient scale to make networked games playable.

Thus, we used GAV to characterize this traffic type in more detail, performing measurements regarding bandwidth usage, server response times, and user behavior. Like other real-time games, GAV requires low bandwidth compared to audio and video applications while the burstiness of data traffic sent by clients is very high. As a maximum acceptable end-to-end delay we evaluated a value of between 100 ms and 200 ms. Meeting this delay bound in a mobile network without resorting to massive overprovisioning does require some form of resource reservation. However, because of the traffic burstiness, the low bandwidth usage, and the tight delay requirements resource reservation on a per session basis, as specified for UMTS for real-time traffic, would cause a significant overhead resulting in inefficient resource utilization and high reservation costs. We therefore propose to introduce a new QoS class for UMTS which is specifically tailored to the needs of mobile games.

For this purpose, we developed a model that aggregates multiple game data flows. Considering the QoS requirements of real-time games, our model is based on queuing systems and exploits statistical multiplexing gains. The actual reservation or priorization is performed for the aggregate of the game flows by traditional means such as Differentiated Services (DiffServ) [3, 18], Resource ReSerVation Protocol (RSVP) [17], or Multiprotocol Label Switching (MPLS) [19]. We validate the model by means of simulation on the basis of real game traffic produced by GAV.



(a) GAV with two players



(b) GAV with four players

Figure 1: GAV screenshots

The remainder of this paper is structured as follows: In the next section we briefly outline related work. We then describe the functionality and architecture of GAV in Section 3. Section 4 presents delay measurements over GPRS and UMTS. Section 5 contains the results from detailed measurements of GAV data traffic and user behavior. The model for aggregation and reservation is presented in Section 6. Also, this section contains the simulation of our approach in a virtual network and demonstrates its performance and characteristics on the basis of real data. We conclude this paper in Section 7 and give an outlook to future work.

2. RELATED WORK

Mobile Gaming is a quite new field of research, as games with real-time demand are currently not well supported by network providers. One of the key reasons is the absence of suitable QoS in the underlying network in combination with a very high cost for overprovisioning. Although the current specification of UMTS [2] proposes QoS classes and support for real-time applications it still remains to be seen when UMTS will be capable to meet the very tight delay requirements of real-time games.

Fitzek et al. therefore propose an architecture consisting of a combination of UTMS and Wireless LAN [6]. The architecture provides multiplayer real-time games among multiple wireless users by using Wireless LAN for the actual game traffic. UMTS is only used for authentication, billing, and game score distribution.

Another architecture for mobile games is presented in [20]. The authors employ both Bluetooth and 433 MHz RF modules to support ad-hoc multiplayer games. Because of the short transmission range of Bluetooth (only 10 meters for Class 3 devices), the 433 MHz RF modules are used to enable ad-hoc gaming over longer distances. However, there is no explicit resource reservation and due to large handover times, this architecture is not suited for real-time games.

Deploying mobile games enables a wide area of novel game concepts. For example, Cheok et al. present with *Human Pacman* a mobile game as a combination of real and virtual world [5]. Pacmen and Ghosts are human players, which can move in a wide outdoor physical area. Virtual cookies and Bluetooth embedded physical tangible objects offer a novel game experience. Game information is distributed by a central server to all humans over Wireless LAN. The movements and the actual position of a human, which is either a Pacman or a Ghost, are obtained via GPS (Global Positioning System) and sent to the server. Each player uses a head mounted display to see the physical real world combined with virtual objects (e.g., cookies) and game information. A quite similar approach is used in [14]. Mitchell et al. developed a mobile game called *Real Tournament*, which is (like *Human Pacman*) an augmented reality multiplayer real-time game.

Numerous approaches exist that target at providing network QoS support and resource reservation. Typically QoS uses admission control to admit or to reject a new traffic flow to meet the QoS requirements of all active connections. Knightly and Shroff give a basic overview of admission control in [11], and evaluate different algorithms. In a first step, admission control algorithms can be divided in deterministic and statistical services. In contrast to deterministic services that often consider the worst case, statistical services attempt to achieve statistical gains resulting in more efficient resource utilization. Therefore, many admission control algorithm exploit the statistical multiplexing effects (economies of scale).

In a second step, admission control algorithms differ in whether they use measurement-based or predefined parameters. A measurement-based algorithm is presented in [15], which exploits statistical multiplexing gains and uses “maximal rate envelops” of aggregate traffic flows to estimate future traffic of applications to support required QoS demands. An example for a predefined algorithm is given in [16]. It is based on leaky buckets, smoothers at the networked edges, and bufferless statistical multiplexing within the network. Reisslein et al. have shown that their algorithm can support significantly more connections than deterministic services. However, a drawback of that algorithm is the insufficient consideration of arrival processes, since only leaky bucket parameters specifying peak rate, average rate, and burst length are used. Especially for highly bursty traffic statistical information about arrival distributions should be taken into account to improve the exploitation of statistical gains. Furthermore, assuming bufferless data transfer through the network results in worst case behavior and wastes resource utilization potential.

Admission control algorithms that handle more detailed information are quite complex, resulting in high computational complexity. The aim of our work is to find an algorithm well suited to the class of mobile real-time games without resorting to an approach that is computationally complex.

3. GPL ARCADE VOLLEYBALL

As an example of mobile games, we use GPL (General Public License) Arcade Volleyball (GAV), a multiplayer real-time volley-

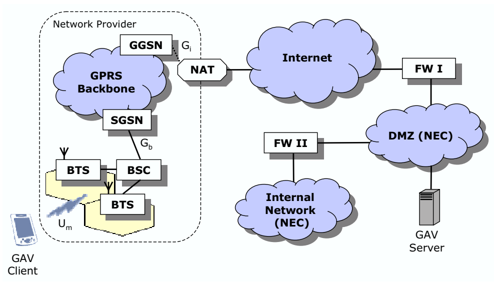


Figure 2: GPRS measurement setup

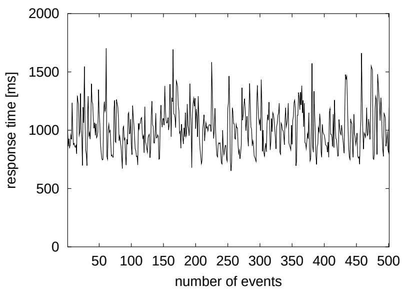


Figure 3: Server response time (RTT) over GPRS

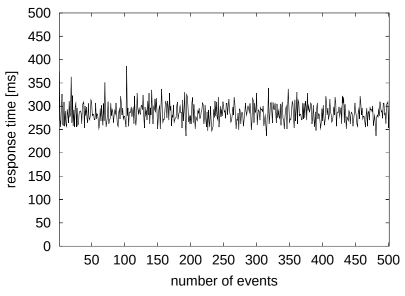


Figure 4: Server response time (RTT) over UMTS

ball game, which is available as open-source on Sourceforge [23]. To use GAV as a mobile game and perform several traffic measurements, we ported it to a PDA, which reflects a more realistic environment for mobile games than a notebook or a PC.

GAV is implemented in C++ and based on the Simple DirectMedia Layer (SDL) [22], which provides among other things multimedia and network support. Thus, porting GAV on a PDA with PocketPC 2002/2003 could be done with Microsoft eMbedded Visual C++. All game information is managed by a server and distributed to clients in states periodically, depending on a frames per second (FPS) rate. A game state contains the positions of all clients, the position of the ball, and the game score. Each client is associated with a single player and can generate three types of events identifying the direction and movement of the player: left, right, and jump. However, the events are not displayed until the client receives a new game state from the server. If a player moves by pressing the appropriate key, the client will create an event and send it to the server. The server receives the event and computes the next game state with a local model. Finally, the new state is sent back to all clients. The packet sizes of states and events are constant and equal to 59 resp. 44 bytes, including the UDP header.

GAV supports up to four players in two teams, this includes both human and artificial computer players. In case of computer players, the control of these players is performed by the GAV server. Figure 1 gives an impression of the game showing two screenshots with two and four players of the ported PDA version.

4. GPRS AND UMTS MEASUREMENTS

In a first step, we use GAV to investigate the suitability of the General Packet Radio Service (GPRS) for mobile real-time games. Our main aim is to determine the game experience of a single GPRS user and the extent to which GPRS currently supports real-time data.

The GPRS measurements are based on a two player GAV game session and the network setup visualized in Figure 2. We use a HP iPAQ h5555 with a GPRS wireless jacket, running Pocket PC 2002 and the GAV client. The second client is located on the GAV server and is controlled by an artificial computer player.

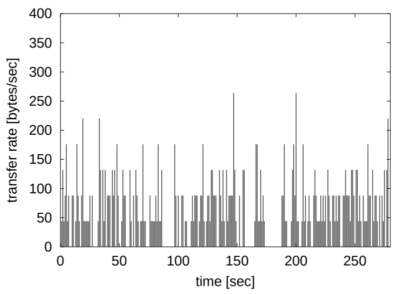
Since our network provider uses Network Address Translation (NAT), we had to use TCP instead of UDP. This changed the data traffic of the original UDP-based GAV significantly. Of 1175 packets sent by a client, only 207 packets corresponds to game events, which is an usage of around 17.6%. The rest is needed for acknowledgements, whereas on the downlink most of the client packets could be acknowledged by piggy-backing, so that just 1.7% explicit acknowledgements were required. This behavior is caused by the sporadic sending of events by clients and the periodic sending of game states by the server. Using TCP also increases the delay due to the packet processing in particular within the iPAQ. Measurements in a controlled WLAN environment have shown that the TCP overhead caused an additional delay of about 15 ms, including processing both by the sender and the receiver of a TCP segment.

The server response time over GPRS measured as round-trip time (RTT) between one client and server is depicted in Figure 3. In both GPRS and UMTS measurements, the server sends on average 20 game states per second. The mean response time is about 1 second, with a range of variation between around 650 and 1700 ms. As can be seen, not only the response time, but also the jitter as the difference between two successive response times is very high. Further investigations have shown that most of the delay is caused by the GPRS access network. On average, the GPRS access network and backbone introduce a delay of about 950 ms, i.e., on the path between client and GGSN.

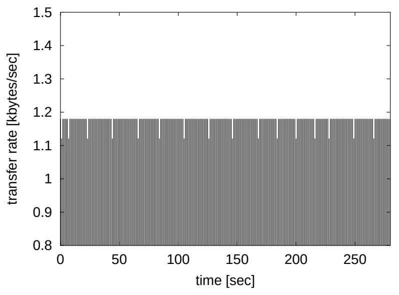
Similar results are presented in [4], where the majority of measurements were performed over Vodafone UK. The mean RTT was in many cases above 1 sec, with different behavior for up- and downlink. The uplink delay varied from 200 to 900 ms and the downlink delay from 200 to 1500 ms. The 95-percentile delay on the uplink was around 1 second and around 2.5 second on the downlink, respectively.

GPRS measurements from Finland led to a mean RTT of about 700 ms varying from 500 to 1100 ms [7]. Compared to previous measurements [12] this was an improvement of about 200 ms. During a stationary connection the RTT was quite stable. However, if an user changed his cell, the RTT varied heavily, and sometimes the connection was cancelled.

Because of these high delays, playing GAV over GPRS is not possible. Since measurements in other countries are quite similar to our results, this is not restricted to our local network provider. Although the GPRS specification contains 4 delay classes [1], QoS is currently not provided by network providers, so that all data is sent with best-effort. Even if GPRS QoS were implemented, the delay classes specified for GPRS support only non real-time appli-

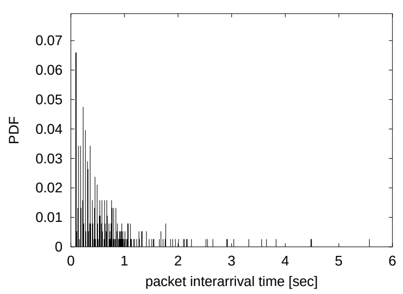


(a) Client transfer rate

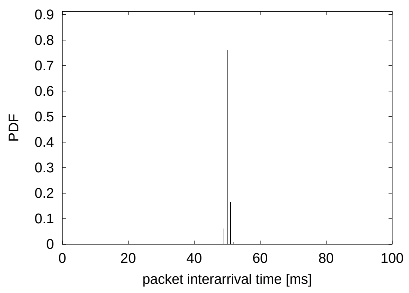


(b) Server transfer rate

Figure 5: Transfer rate of client and server

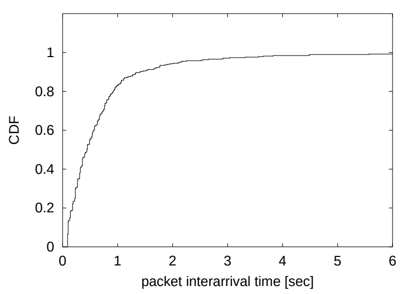


(a) PDF of client packet interarrival time

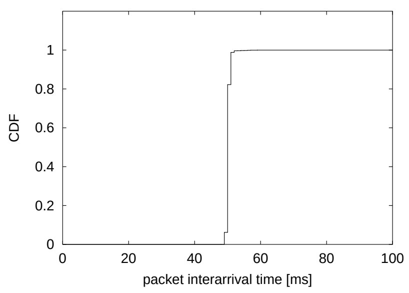


(b) PDF of server packet interarrival time

Figure 6: Probability distribution function (PDF) of client and server interarrival time



(a) CDF of client packet interarrival time



(b) CDF of server packet interarrival time

Figure 7: Cumulative distribution function (CDF) of client and server interarrival time

cations and even the best class would not be sufficient to meet our requirements.

In a second set of experiments, we investigated the use of UMTS for networked mobile games. The UTMS service used for our measurements was just recently launched in Germany and is currently available for data communication only. It is a pure best-effort service and does not provide any QoS guarantees. The setup of the experiments was the same as for GPRS. The results of the delay measurements over UMTS are depicted in Figure 4. The mean RTT is around 285 ms and varies between 236 and 386. As 90-percentile delay we get a value of around 230 ms. Although the delay over UMTS is much better than the one over GPRS, the tight delay requirements of all real-time games cannot yet be met even by UMTS.

Using an artificial delay, we evaluated the maximum acceptable delay for GAV, which was between 100 ms and 200 ms. This is similar to Schaefer et al. [21], who specify a maximum delay of 139 ms for mobile real-time games. Neither GPRS nor UMTS are currently able to meet these delay bounds.

5. TRAFFIC MEASUREMENTS OF GAV

To further analyze the traffic characteristics of GAV in more detail, we performed measurements between client and server in an IEEE 802.11b Wireless LAN in ad-hoc mode with 11 Mbps. We used two PDAs and a 650 MHz Sony Vaio notebook, running Microsoft Windows XP. Both PDAs were Toshiba Pocket PCs e740 with a 400 MHz XScale processor and PocketPC 2002 operating system. The GAV client was started on each PDA and the GAV server on the notebook. All data packets were captured with Ethereal on the network interface of the notebook. During a Tournament, we created trace files for a total of 25 game sessions consisting of two clients playing against each other. As a result, we got 50 data flows for each up- and downlink, which were further analyzed.

5.1 Measurements of a Single Game Session

The measurements in this section are an example of one game session but represent typical characteristics of client and server traffic. As one can see in Figure 5, the transfer rate of a client is very bursty while it is relatively smooth for the server. The mean transfer rate is around 60 bytes/sec (client) and around 1.17 kbytes/sec (server) with a variation coefficient of about 0.95 resp. 0.01. Compared to other real-time applications like video, these rates are very low. However, the transfer rate is extremely bursty for the client: it varies from 0 to 264 bytes/sec. Note that the packet sizes of events and states are constant, so the transfer rate measured as packets/sec would have the same shape as depicted in Figure 5, with a mean of around 1.4 packets/sec resp. 20 packets/sec.

Figure 6 shows the probability distribution function and Figure 7 the cumulative distribution function of the packet interarrival times (distance between two packets in time) of client and server with a mean of around 0.7 sec (client) and 50 ms (server). The variation coefficient is about 1.7 resp. 0.04. The interarrival times of the client vary from 90 ms to 15 sec. This is caused by active and inactive phases of clients: events were mostly sent in active phases when the ball is on the player's side. Otherwise, the client is inactive and is waiting for the other player to return the ball. Regarding Figure 5(a), the ratio between active and inactive phases is around 1:2. The interarrival times of the server are almost constant and vary from only 48 ms to at most 60 ms but are very often close to the mean. Reasons might be the timer management and in some cases the server load.

The server response time (RTT) is depicted in Figure 8. The mean response time is around 74 ms and varies from 40 ms to 103 ms because of periodical sending of states by the server. However, the network latency was under 10 ms, so the most part of the delay is caused by the server itself.

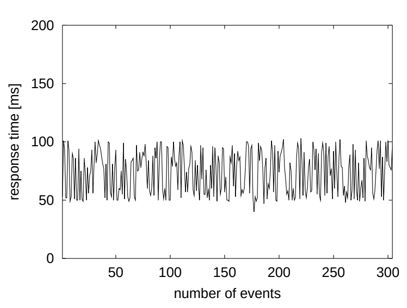
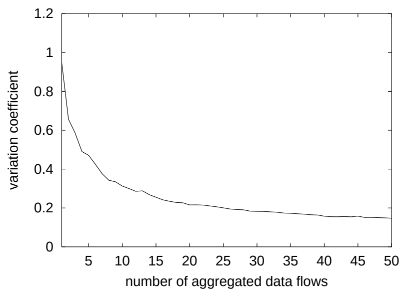
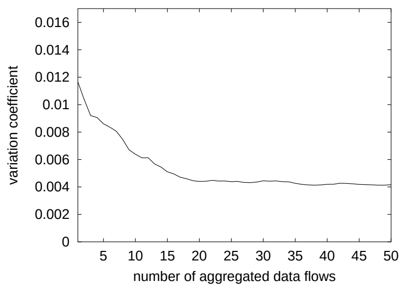


Figure 8: Server response time (RTT) over Wireless LAN



(a) Variation coefficient of aggregated client flows



(b) Variation coefficient of aggregated server flows

Figure 9: Variation coefficient of aggregated client and server data flows

5.2 Aggregating Data Flows

As we have seen, GAV requires only a low bandwidth, but client data traffic is very bursty, so that constant-bit-rate resource reservation for a single game session might be inefficient and quite expensive. Reducing reservation costs and smoothing single bursts could be possible, if we consider data traffic of more than one game session together and aggregate multiple data flows of different players. The smoother the aggregated data flow is, the better will be the resource usage. To measure how smooth a data flow is, we use the coefficient of variation, which is the ratio between standard deviation and the average value.

Figure 9 shows the variation coefficient with respect to the number of aggregated data flows. The variation coefficient decreases with 10 aggregated flows by around 67% for the client and by around 45% for the server data flows. If all 50 flows are aggregated the decrease is around 85% resp. 65%. Although each server data flow is already very smooth, the improvement is still considerable. Regarding the resource reservation, this shows that the aggregation of only a few data flows would already lead to a much more efficient bandwidth usage. This is important, since for a mobile network most likely only a few game sessions can be aggregated per cell.

6. RESOURCE RESERVATION

Since the communication required for networked games such as GAV is very bursty, we propose to make reservations for aggregates of multiple game sessions. Given such an aggregate we need to know how much bandwidth should be reserved, taking multiplexing gains into account. We assume that the actual allocation of bandwidth for the aggregate is then performed by existing means such as RSVP [17].

In order to determine the amount of bandwidth required for an aggregate, we derive an algorithm for admission control from a simple queuing model. Taking into account the multiplexing gains, this algorithm decides by how much an additional flow would increase the bandwidth required for the aggregate and whether it may be accepted or not. We show that the model is appropriate for traffic from networked games by means of simulation with real game data produced by GAV.

6.1 Queuing Model

In order to calculate the required bandwidth, we employ M/G/1 queuing systems [9, 10], which assume a Poisson distributed arrival process, general distributed service times, one service unit, an infinite waiting room, and FIFO as the queuing discipline.

Considering a specific router i, we use for each link j a queuing system. The appropriate notation is then as follows:

QS_{ij}	-	Queuing system of router i and link j
Ω_{ij}	-	Set of all data flows k arriving at QS_{ij}
λ_k	-	Mean arrival rate of data flow k
λ_{ij}	-	Mean arrival rate at QS_{ij}
μ_{ij}	-	Mean service rate of QS_{ij}
$\tilde{\mu}_{ij}$	-	Mean bandwidth on link j of router i
ρ_{ij}	-	Mean system utilization of QS_{ij} ($\rho_{ij} = \lambda_{ij}/\mu_{ij}$).
N_{ij}	-	Mean number of packets in QS_{ij}
W_{ij}	-	Mean waiting time of QS_{ij}
N_{ij}^{max}	-	Maximum mean buffer size of QS_{ij}
W_{ij}^{max}	-	Maximum waiting time of QS_{ij}
B_{ij}^{max}	-	Maximum mean bandwidth on link j of router i
τ_i	-	Mean processing time of router i
$Size$	-	Mean packet size

Our aim is to compute the optimal mean bandwidth $\tilde{\mu}_{ij}$ of the queuing system QS_{ij} under following constraints:

$$\begin{array}{lll} \text{(i)} & 0 < \lambda_{ij} < \mu_{ij} & \forall i,j \geq 0, \\ \text{(ii)} & E(W_{ij}) \leq W_{ij}^{max} & \forall i,j \geq 0, \\ \text{(iii)} & E(N_{ij}) \leq N_{ij}^{max} & \forall i,j \geq 0, \\ \text{(iv)} & 1/\mu_{ij} > \tau_i & \forall i,j \geq 0, \\ \text{(v)} & \tilde{\mu}_{ij} \leq B_{ij}^{max} & \forall i,j \geq 0. \end{array}$$

The first constraint is also called Condition for Stability and must be met to make all formulas valid. The second and third constraint bound the mean waiting time (delay) and the mean number of packets in the system. Although we assume an infinite waiting room, we would like to control the mean buffer size of the queuing system, even if dropping of packets is not modelled. The fourth constraint makes sure that the mean service time, consisting of routing and transfer time, is at least as large as the mean router processing time (e.g., time for header inspection). Finally, the last constraint enforces a maximum on the mean bandwidth.

Let B_{ij} be the service time of QS_{ij}, then $\mu_{ij} = 1/E(B_{ij})$. According to the Pollaczek-Khintchine formula [9], the mean number of packets $E(N_{ij})$ in a M/G/1 queuing system can be computed for $\rho_{ij} < 1$ as

$$E(N_{ij}) = \frac{\rho_{ij}^2}{2(1-\rho_{ij})}(VC^2(B_{ij})+1) + \rho_{ij},$$

with the variation coefficient $VC(B_{ij})$ of the service time B_{ij}.

According to Little's Law [13], constraint (ii) is equivalent to $E(W_{ij}) \leq N_{ij}^{max}/\lambda_{ij}$. Hence, we combine constraint (ii) and (iii) to $E(W_{ij}) \leq \min\{W_{ij}^{max},\ N_{ij}^{max}/\lambda_{ij}\}$.

```
*Admission Control Algorithm* (data flow $\hat{k}$):

(1) Compute path $\omega_{\hat{k}}$ of flow $\hat{k}$ as a set of tuples $(i,j)$,
    with router $i$ and link $j$.

(2) Compute optimal transfer rate for all involved links:
        $b \leftarrow true$
        For each tuple $(i,j) \in \omega_{\hat{k}}$ do
            Compute aggregated arrival rate $\lambda_{ij}$:
                $\lambda_{ij} \leftarrow \sum_{k \in \Omega_{ij}} \lambda_k + \lambda_{\hat{k}}$
            Compute optimal service rate $\mu^*_{ij}$:
                $\xi \leftarrow \min\{W^{max}_{ij},\ N^{max}_{ij}/\lambda_{ij}\}$
                $\mu^*_{ij} \leftarrow \frac{1+\xi\lambda_{ij}+\sqrt{1+(\xi\lambda_{ij})^2+2\xi\lambda_{ij}VC^2(B_{ij})}}{2\xi}$
            Compute optimal bandwidth $\tilde{\mu}^*_{ij}$, if $1/\mu^*_{ij} > \tau_i$:
                $\tilde{\mu}^*_{ij} \leftarrow \left(\frac{1}{\mu^*_{ij}} - \tau_i\right)^{-1} \cdot Size$
            If $\tilde{\mu}^*_{ij} > B^{max}_{ij}$ or $1/\mu^*_{ij} \leq \tau_i$
                then $b \leftarrow false$

(3) If the available bandwidth is sufficient, accept the
    new connection:
        If $b = true$
            then accept data flow $\hat{k}$:
                For each tuple $(i,j) \in \omega_{\hat{k}}$ do
                    $\Omega_{ij} \leftarrow \Omega_{ij} \cup \{\hat{k}\}$
            else reject flow $\hat{k}$ .
```

Figure 10: Admission control algorithm

For $\xi = \min\{W^{max}_{ij},\ N^{max}_{ij}/\lambda_{ij}\}$ we get then

$$E(W_{ij}) = \frac{E(N_{ij})}{\lambda_{ij}} = \frac{1}{\lambda_{ij}}\left[\frac{\rho^2_{ij}}{2(1-\rho_{ij})}(VC^2(B_{ij})+1)+\rho_{ij}\right]$$

$$= \frac{\lambda_{ij}/\mu_{ij}(VC^2(B_{ij})+1)+2(1-\lambda_{ij}/\mu_{ij})}{2(\mu_{ij}-\lambda_{ij})} \leq \xi$$

$$\Leftrightarrow \quad 0 \leq 2\xi\mu^2_{ij} - (2\xi\lambda_{ij}+2)\mu_{ij} - \lambda_{ij}(VC^2(B_{ij})-1).$$

Because of constraint (i), only one solution is valid. So this leads to

$$\mu_{ij} \geq \frac{1+\xi\lambda_{ij}+\sqrt{1+(\xi\lambda_{ij})^2+2\xi\lambda_{ij}VC^2(B_{ij})}}{2\xi}.$$

Since $E(W_{ij})$ is concave for $\rho_{ij} < 1$, we get under optimization of the system utilization: $\mu^*_{ij} = \min\{\mu_{ij}\}$. In addition, due to $\mu^*_{ij} > (1+\xi\lambda_{ij}+\sqrt{(\xi\lambda_{ij})^2})/(2\xi) > (2\xi\lambda_{ij})/(2\xi) = \lambda_{ij}$ constraint (i) is fulfilled.

The optimal bandwidth is then

$$\tilde{\mu}^*_{ij} = \left(\frac{1}{\mu^*_{ij}} - \tau_i\right)^{-1} \cdot Size.$$

The physical transport time on a link is obtained by the service time minus the routing time. The reciprocal of this value is then the physical transport time measured in packets. Multiplied with the mean packet size this leads to the optimal bandwidth, which is required to meet constraints (i) to (iv) under the given assumptions.

6.2 Admission Control Algorithm

Based on this result we can now give the algorithm for the admission control shown in Figure 10. Consider a single data flow $\hat{k}$ and a network of routers and links. Each tuple (i,j) represents the queuing system of router i and link j. First, we compute the path $\omega_{\hat{k}}$ of data flow $\hat{k}$ through the network. In a second step, we consider each involved queuing system on the path and compute the required bandwidth. Since we assume Poisson distributed arrivals, the aggregation of existing data flows with the new flow $\hat{k}$ is also a Poisson process. Therefore, the mean arrival rate is $\lambda_{ij} = \sum_{k \in \Omega_{ij}} \lambda_k + \lambda_{\hat{k}}$. Only if the computed bandwidth for each link on the path is smaller than or equal to the maximum available bandwidth and is also greater than zero, the new game session is accepted. In this case, data flow $\hat{k}$ is added to the set of already accepted flows Ω_{ij}.

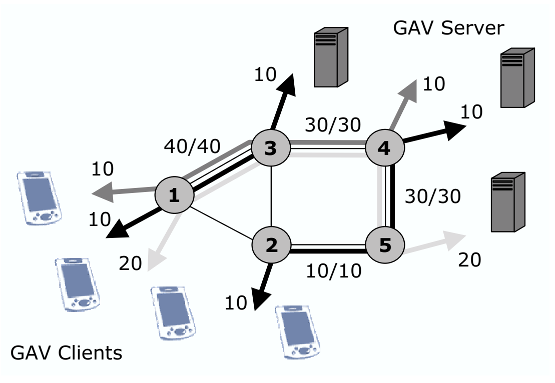


Figure 11: Example simulation network

6.3 Different Service Times

Up to this point, we did not take different packet sizes into account. Therefore, we compute the service time depending on the mean packet size and its coefficient of variation.

Let $E(B^r)$ be the mean router processing time and $E(B^t)$ the mean transfer time. With $\mu = 1/E(B)$, this leads to $E(B) = E(B^r) + E(B^t)$. For a constant transfer rate $\tilde{\mu}$ and with $B^t = Size/\tilde{\mu}$, we get $E(B^t) = E(Size)/\tilde{\mu}$ and $V(B^t) = V(Size)/\tilde{\mu}^2$.

Let B^r and B^t be stochastically independent, then we can compute the variance of the service time as follows:

$$V(B) = V(B^r) + V(B^t) = V(B^r) + \frac{V(Size)}{\tilde{\mu}^2}$$

$$= V(B^r) + \frac{V(Size)}{E^2(Size)}(E(B)-E(B^r))^2$$

$$= V(B^r) + VC^2(Size)(E(B)-E(B^r))^2.$$

This leads directly to the coefficient of variation:

$$VC^2(B) = \frac{V(B)}{E^2(B)}$$

$$= V(B^r)\mu^2 + VC^2(Size)(1-2E(B^r)\mu+E^2(B^r)\mu^2)$$

$$= (V(B^r)+VC^2(Size)E^2(B^r))\mu^2 - 2VC^2(Size)E(B^r)\mu + VC^2(Size).$$

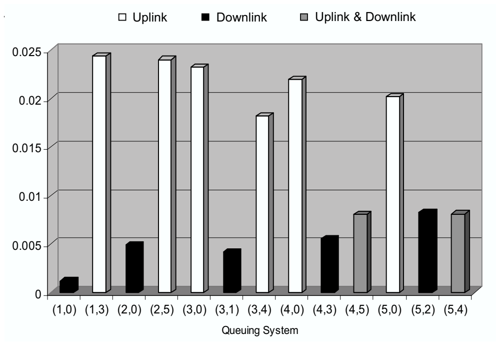
With

$$a = 2\xi - \lambda_{ij}(V(B^r)+VC^2(Size)E^2(B^r)),$$

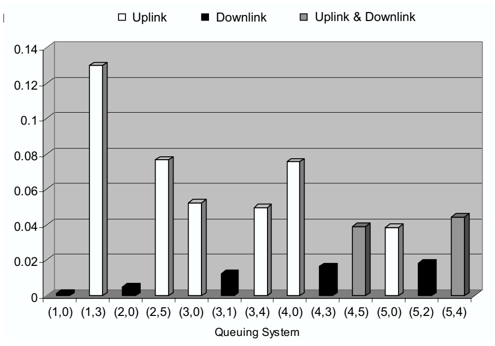
$$b = \lambda_{ij}(2VC^2(Size)E(B^r)-2\xi)-2, \text{ and}$$

$$c = \lambda_{ij}(1+VC^2(Size)).$$

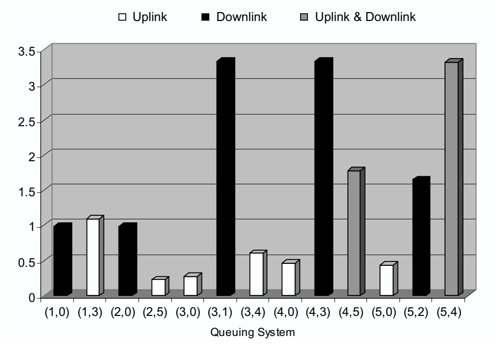
the optimal service rate is then $\mu^*_{ij} = \frac{-b+\sqrt{b^2-4ac}}{2a}$.



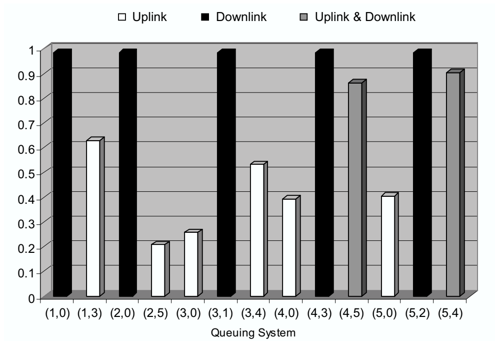
(a) Mean delay $E(W)$



(b) Maximum delay $Max(W)$



(c) Mean number of packets $E(N)$



(d) Mean utilization ρ

Figure 12: Simulation results

6.4 Simulation

In this section, we investigate the performance and characteristics of the presented admission control algorithm on the basis of real GAV data flows by means of simulation. We use the simplified, virtual network shown in Figure 11 with five nodes and four aggregated data flows, which we have captured as described in Section 5. The number of flows which are aggregated is depicted on each arrowhead referring to the appropriate direction. For the number of flows on each link we use the notation $< uplink >$ / $< downlink >$.

Each node represents a router i with $(1 \leq i \leq 5)$. Each tuple (i, j) with $(1 \leq i, j \leq 5)$ then identifies a queuing system in router i and link to router j. For all data flows which leave the virtual network we use the queuing systems $(i, 0)$. The optimal bandwidth on each link shall be computed according to the admission control algorithm shown in Figure 10. For the model the following assumptions were made:

1. Arrivals on uplinks are Poisson distributed, and arrivals on downlink are deterministic.
2. Packets are routed according FIFO. The router processing time is neglected.
3. The available bandwidth and buffer size are infinite.
4. The mean delay shall be at most 25 ms for each link.

The mean packet rate of a client over all game sessions was around 1.1 packets/sec and around 20 packets/sec for a server. So we get for n data flows $\lambda_{up} = 1.1n$ and $\lambda_{down} = 20n$.

Because of deterministic arrivals of server packets and deterministic service times, the downlink bandwidth would actually be $\mu^*_{down} = \lambda_{down}$. Due to smoothing small variations on the downlink, we consider the corresponding coefficient of variation and use $\mu^*_{down} = 1.01\lambda_{down}$ instead.

We consider up- and downlink separately because of different arrival processes. Hence, the optimal bandwidth on each link is then $\mu^* = \mu^*_{up} + \mu^*_{down}$, with μ^*_{up} computed as described in the previous section.

The results of the simulation are depicted in Figure 12. Figure 12(a) shows the mean delay on each link, which is always smaller than our delay bound of 25 ms. However, the first time a data flow enters the network the mean delay is close to 25 ms. After that, the delay decreases on each link along the routing path. This is caused by the used queuing system type because each system assumes Poisson distributed arrivals. Due to deterministic service times, the appropriate departure process is in general not Poisson distributed and a little smoother than before.

Figure 12(b) shows the maximum delay and Figure 12(c) the mean number of packets in the system. In spite of a mean delay of at most 25 ms, even a maximum delay above 120 ms is possible (see queuing system $(1, 3)$). Finally, Figure 12(d) depicts the mean system utilization. The downlink shows a constant utilization of 99% due to $\mu^*_{down} = 1.01\lambda_{down}$. The uplink system utilization is much smaller, but this is necessary in order to meet the maximum mean delay.

The low mean delay shows that the assumption about Poisson distributed arrivals on uplink fits well. However, to meet the maximum mean delay, the serving rates have to be sometimes much bigger than the arriving rate. The ratio between both values is expressed in the system utilization. The more flows are aggregated and the higher the maximum mean delay can be, the better is the system utilization.

The effect of statistical multiplexing compared to a constant-bit-rate reservation is as follows: For example, we consider queuing system $(1, 3)$ and assume a 95-percentile constant-bit-rate reservation for each single game session, which is around 8 packets/sec averaged over all captured sessions. So we get for all sessions a constant-bit-rate reservation of 320 packets/sec. However, the optimal bandwidth computed for queuing system $(1, 3)$ is only around 73 packets/sec, which is a bandwidth decrease of around 77%.

7. CONCLUSIONS AND OUTLOOK

In this paper, we have presented a lightweight QoS approach to support networked mobile gaming. Based on a simple real-time multiplayer game called GAV we analyzed the network traffic of wireless action games in detail. As we have shown, the required bandwidth is low, compared to other real-time applications (e.g., audio and video). The client traffic is very bursty and the server traffic quite smooth. This is caused by sporadic events sent by clients opposed to a periodic transmission of the game states by the server.

In a first measurement over GPRS, we encountered a high mean RTT of around 1 sec, which varied heavily and made playing virtually impossible. Measurements over UMTS had shown a delay of around 285 ms, which is still problematic for most real-time games. Since those applications need a very small end-to-end delay, QoS support in the network is essential as long as a substantial overprovisioning cannot be provided. However, because of the burstiness of data traffic and the resulting costs, reserving resources, e.g., for the maximum transfer rate, is not advisable. Therefore, aggregating multiple data flows to smooth bursts is reasonable and achieves statistical multiplexing gains. As we have seen, the coefficient of variation could be decreased by about 85% on the uplink and 65% on the downlink, if 50 data flows were aggregated. A common reservation for aggregated flows is therefore very reasonable.

Based on a M/G/1 queuing system, we developed an algorithm to compute the optimal bandwidth, which is required to meet a specific delay bound. Through simulation, we demonstrated the performance and the characteristics of the algorithm based on real data in a simplified, virtual network.

In future work, we plan to extend the admission control algorithm to account for games with distinct traffic patterns and allow for per-flow priorities. Furthermore, we would like to provide an efficient way to perform incremental modifications to the reserved rate for the aggregate.

8. REFERENCES

[1] 3GPP TS 22.060. General Packet Radio Service (GPRS), Service description, Stage 1.

[2] 3GPP TS 23.107. Quality of Service (QoS) concept and architecture.

[3] M. Brunner, A. Gonzalez, and P. Martinez. From dynamic IP transport service ordering to DiffServ network configuration. *Computer Networks*, 43:25–41, 2003.

[4] R. Chakravorty and I. Pratt. Performance Issues with General Packet Radio Service, under submission to Journal of Communication and Networks (JCN), 2002.

[5] A. D. Cheok, S. W. Fong, K. H. Goh, X. Yang, W. Liu, and F. Farzbiz. Human Pacman: A Sensing-based Mobile Entertainment System with Ubiquitous Computing and Tangible Interaction. In *Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames)*, pages 98–109, Redwood City, CA, USA, May 2003.

[6] F. Fitzek, G. Schulte, and M. Reisslein. System Architecture for Billing of Multi-Player Games in a Wireless Environment using GSM/UMTS and WLAN Services. In *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames)*, pages 58–64, Braunschweig, Germany, Apr. 2002.

[7] A. Gurtov, M. Passoja, O. Aalto, and M. Raitola. Multi-Layer Protocol Tracing in a GPRS Network. In *Proceedings of the IEEE Vehicular Technology Conference (Fall VTC 2002)*, Vancouver, Canada, Sep. 2002.

[8] T. Henderson. Latency and user behaviour on a multiplayer games server. In *Proceedings of the 3rd International Workshop on Networked Group Communication (NGC)*, pages 1–13, London, UK, Nov. 2001.

[9] L. Kleinrock. *Queuing Systems*, Volume I: *Theory*. J. Wiley & Sons, New York, 1995.

[10] L. Kleinrock. *Queuing Systems*, Volume II: *Computer Applications*. J. Wiley & Sons, New York, 1995.

[11] E. Knightly and N. Shroff. Admission Control for Statistical QoS: Theory and Practice. *IEEE Network*, 13(2):20–29, 1999.

[12] J. Korhonen, O. Aalto, A. Gurtov, and H. Laamanen. Measured Performance of GSM HSCSD and GPRS. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 2001.

[13] J. D. C. Little. *A Proof of the Queuing Formula $L = \lambda W$.* Operation Research, 9:383–387, 1961.

[14] K. Mitchell, D. McCaffery, G. Metaxas, J. Finney, S. Schmid, and A. Scott. Six in the City: Introducing Real Tournament – A Mobile IPv6 Based Context-Aware Multiplayer Game. In *Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames)*, pages 83–92, Redwood City, CA, USA, May 2003.

[15] J. Qiu and E. W. Knightly. Measurement-based admission control with aggregate traffic envelopes. *IEEE/ACM Transactions on Networking*, 9(2):199–210, 2001.

[16] M. Reisslein, K. W. Ross, and S. Rajagopal. Guaranteeing Statistical QoS to Regulated Traffic: The Single Node Case. In *INFOCOM (3)*, pages 1061–1072, 1999.

[17] RFC 2205. Resource ReSerVation Protocol (RSVP).

[18] RFC 2475. An Architecture for Differentiated Services.

[19] RFC 3031. Multiprotocol Label Switching Architecture.

[20] H. Ritter, T. Voigt, M. Tian, and J. Schiller. Experiences Using a Dual Wireless Technology Infrastructure to Support Ad-hoc Multiplayer Games. In *Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames)*, pages 93–97, Redwood City, CA, USA, May 2003.

[21] C. Schaefer, T. Enderes, H. Ritter, and M. Zitterbart. Subjective Quality Assessment for Multiplayer Real-Time Games. In *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames)*, pages 74–78, Braunschweig, Germany, Apr. 2002.

[22] Simple DirectMedia Layer. SDL-Library. http://www.libsdl.org/.

[23] Sourceforge. GPL Arcade Volleyball (GAV). http://sourceforge.net/projects/gav/.

[24] M. Terrano and P. Bettner. 1500 archers on a 28.8: Network programming in Age of Empires and beyond. In *Proceedings of the 15th Games Developers Conference*, San Jose, CA, USA, Mar. 2001.