# Relevance-based Adaptive Event Communication for Mobile Environments with Variable QoS Capabilities

Stephen Workman, Gerard Parr, Philip Morrow, Darryl Charles

Faculty of Engineering, University of Ulster, Coleraine, BT52 1SA, Northern Ireland
{workman-s, gp.parr, pj.morrow, dk.charles}@ulster.ac.uk

**Abstract**. Recent trends in computing have been driving the demand for mobile multimedia applications, specifically distributed virtual environments (DVEs). These applications must deal with the variable resource availability of both connection and client device in order to achieve real-time event communication. Relevance-based event filtering is used to explore event stream adaptation in response to variable QoS. Results show that the performance gains from such adaptation are inconsistent due to the irregular nature of event communication. Increased reliability is proposed through dynamic consideration of the resource requirements of the various adapted event stream solutions.

## 1 Introduction

Recent trends in computing show increasing demand for mobile multimedia, specifically distributed virtual environments (DVEs) [1, 2]. Such multimedia applications take advantage of the mobility afforded by wireless networking and the pervasive nature of mobile devices. For example, in the Savannah Project [3] a DVE was used over mobile devices as part of an augmented reality game, to help educate children about lions and the savannah. For such virtual environments distributed over wireless networks, the tension between the real-time resource requirements and the inherent variable resource availability of both connection and client device must be managed by the server. Application layer adaptation of data has been suggested to overcome this resource variability with both general [4, 5] and event-based data communication [6]. This paper demonstrates how relevance-based event filtering as a method of data adaptation should take resource requirements of event type streams into consideration when adapting to the currently available device and connection resources.

## 2 Background and Existing Approaches

Real-time distributed applications in the wireless domain can be examined from two main viewpoints: communication and application [7]. The communication aspect deals with those issues pertaining to the transport of data across the network, while the application aspect deals with the encoding, decoding and use of data before and after transport.

The main resource requirement made of the communication aspect by DVEs is that data be transported in real-time. This requires a high quality of service (QoS) requested from the network (high data throughput, low latency etc.) and the client (processing power, storage etc.). The 3rd Generation Partnership Project (3GPP) provides traffic classes for resource reservation in the sub-IP layers, including the "conversational" class meant for real-time transport [8]. Resource reservation is also provided in the IP layer through IntServ and DiffServ – in [9] the authors suggest the use of aggregated flows as the best way to provide the necessary QoS for event communication.

There are three main issues with the use of such reservation schemes in the mobile domain, however. Firstly, due to host mobility, QoS cannot be guaranteed for the life of the session due to connection handovers [10, 11]. Although delays incurred by handover can be reduced, e.g. [10], longer term disruptions can be caused if the new base station does not have the same resources as the old one, e.g. in the case of vertical handoffs [10-12]. Secondly, the resources set aside by a base station do not guarantee that the mobile node will be able to use them, since the QoS actually achieved is variable [12-15]. Trade-offs at the lower layers, in response to changing interference and fading, can reduce link throughput, resulting in lower bandwidth available to the application. Thirdly, there is no guarantee that traffic which crosses autonomous systems will have the same resources set aside as those at the edge, i.e. in the case of internetworking. Dynamic resource management is suggested in the network layer as a response to this resource variability [12].

Feedback of all such resource variability must be provided to the application layer. In session description protocol over session initiation protocol (SDP/SIP) [16, 17] resources are negotiated between peers during session setup, and re-negotiated during session run-time should conditions change. This capability to renegotiate without restarting a session allows the application layer to adapt to the variable network conditions inherent to mobile and wireless networking. Nonetheless, under SDP all the resources of the session need to be renegotiated, and not just those which have changed. The authors of [18] highlight the long delay in session renegotiation following a vertical handover while using SDP/SIP, while in [19] the End-to-End Negotiation Protocol (E2ENP) is provided as an alternative to SDP, where only relevant resources are re-negotiated.

The application layer must respond to variable network resource availability by changing the resource requirements of the pending traffic (i.e. events). All types of DVE contain client and server components [6], where the role of the server(s) is to maintain a consistent state of the virtual environment across all participating clients. In the mobile environment this role expands to take resource variability and connection heterogeneity into consideration. In [20] the authors suggest that the server deal with varying latencies through buffering real-time updates. This, however, doesn't deal with other variations in resources e.g. bandwidth. Event filtering methods can be used to cut down on the amount of data transmitted to certain clients in distributed virtual environments. Criteria for what is dropped can be based on spatial location in the gaming world, i.e. only transmitting data to players whose immediate environment is affected [21]. Interest in events can also be specified by clients, allowing them to choose event types which are relevant to them [21-23]. Another method suggests dropping obsolete events [6] i.e. events are dropped because their

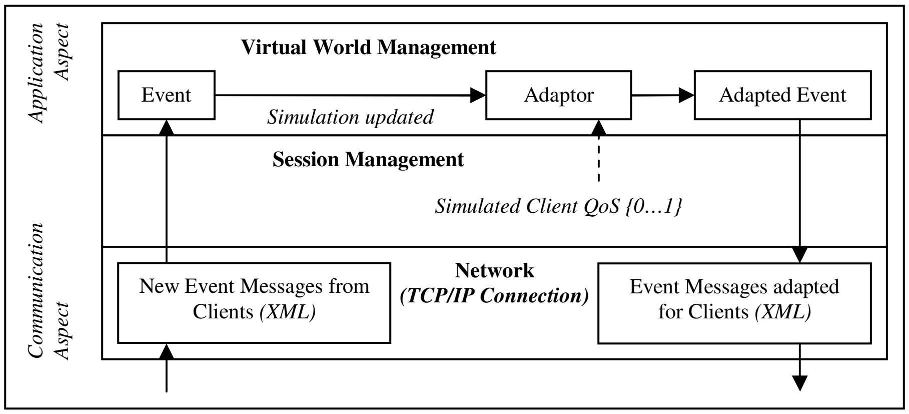

**Figure 1 Event flow between arrival on and departure from the server**

effect on the environment is contained in the effects of subsequent events. Methods can also be combined: the use of a two-tiered server network involving a separation of relevance-filtering and bandwidth adaptation is suggested in [24]. In general, these filtering techniques try to reduce the resource requirements (specifically bandwidth) of the data that is to be sent across the network to the connected nodes, while maintaining a state consistency which is adequate for user interaction.

This paper demonstrates through simulation that event filtering based purely on location relevance does not take into consideration the effect that changing levels of activity in the virtual world can have on resource requirements. Adaptation which considers the resource requirements of the potential adapted event streams is proposed; the resources of the event stream are conditioned to match those resources which are available on the network and client.

## 3 Simulation Architecture

The aim of these simulations was to explore location-relevance filtering as a way to adapt event communication based on variable QoS. A two-player version of the classic arcade game Bomberman was chosen as an initial study of adaptive multipoint communication, as multiplayer games are typical of expected real-time mobile multimedia applications [1, 2]. Scalability testing and related issues are to be explored in future work. The rest of this section describes the current simulation test-bed, with particular emphasis given to the event communication model and the adaptation algorithm, in the context of the client/server architecture used.

### 3.1 Test-bed Architecture

To understand the adaptation, it is important to understand the context of the game architecture. Virtual environments consist of a collection of virtual objects, as discussed in depth in [7]; Bomberman's virtual environment, or game world, is a 2D maze which is filled with piles of dirt and rocks. The aim of the game is to manoeuvre one's avatar around the dirt and rocks, laying bombs to clear paths through the dirt, with the ultimate goal of blowing up the other player's avatar and being the last alive. These objects are based on *SharedObjects*, from [7], since replicas of each exist at participating nodes. To maintain consistency between replicas, attribute changes are distributed between peers. This event processing is described in more detail later.

A client/server architecture was chosen for this implementation, as it the basis for all DVEs [6]. The internal components of this architecture, based on the OSI 7 layer model, are shown in Figure 1. The game world is located on the application level, and identical copies are created on both the server and the clients when a session begins, i.e. during session setup the server establishes the initial state of the game world, and communicates this to the clients via an XML-based protocol. This part of the communication does not have real-time constraints. For the purposes of these simulations, adaptation is only performed on event messages leaving the server.

The state of the virtual world is maintained consistent across participating nodes, via event communication. When a client-side user interacts with the game world, the event is sent to the session layer where it is converted into XML and passed to the network layer for transmission to the server. On the server, the XML is parsed in the session layer, and passed up to the application layer, where it is assimilated into the server-side copy of the virtual world. After the simulation is updated, the event is passed to the adaptor (discussed in Section 3.2). Based on the client's simulated QoS obtained from the session layer, it is updated with events as is deemed necessary by the adaptor. Note that in the standard version of the game there is no adaptor.

The software was created using a combination of "off-the-shelf" and new components. Both the server and client were written in Java (J2SE and J2ME, respectively) using the NetBeans IDE due to Java's widespread use among mobile devices [25]. XML was used as the basis for the event communication protocol because of its suitability for use with an object-oriented model. The protocol also included basic session setup capabilities because of the lack of readily available support for SDP/SIP in J2ME. Existing Java TCP socket classes were used because of TCP's standard use in industry and as a basis for future studies involving variants which are more suitable to wireless communications.

### 3.2 Adaptation

The adaptation used changes how a client's copy of the game state is synchronised. This is based on the relevance of the events to be sent to the client, and the current QoS available to it, as in the algorithm outlined in figure 3. QoS availability was simulated and followed a sawtooth pattern, continuously cycling between poor and good service, in order to see how the adaptation functioned in response to varying QoS. QoS values ranged from 0 (i.e. minimum resources needed for gameplay) to 1 (i.e. optimal resources requiring no adaptation), with increments of 0.1 between.

This adaptation was based on the notion that events closer to an avatar are more relevant and should be given more priority in distribution (Figure 2) [21]. "Player movement" events were deemed to have absolute relevance, regardless of location, since knowledge of other players' locations is needed in order to play the game properly; thus these are added to the clients' queues without being adapted. Similarly,

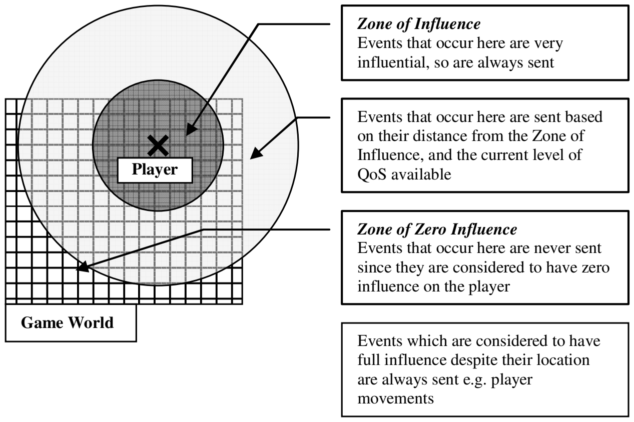
*Zone of Influence*
Events that occur here are very influential, so are always sent
Events that occur here are sent based on their distance from the Zone of Influence, and the current level of QoS available
*Zone of Zero Influence*
Events that occur here are never sent since they are considered to have zero influence on the player
Events which are considered to have full influence despite their location are always sent e.g. player movements
Player
Game World

**Figure 2 Location based relevance filtering**

if a client has QoS of 1 all events are sent as it has enough resources to process this data. If the client's simulated QoS is between 1 and 0, however, the events are checked for relevance, based on their location relative to the client's avatar (figure 2 explains the notion of relevance based on location). The zone of influence varies in size with the client's simulated QoS; in other words, with more resources available, the zone is bigger and more events are sent; whereas with fewer resources, the zone is reduced in size and more events are adapted. The basic size was varied in the simulations, as is explained further in the results section. It is important to note that since these are foundational simulations, the client event queues work purely on a first-come-first-served basis, and are not priority based. Further work is also to be conducted into the upper and lower thresholds of available QoS.

An adapted event is one whose message is incorporated into a simulation wide update message. Rather than sending multiple smaller event messages, those deemed less relevant by the algorithm discussed previously, are dropped. The server then sends an update message, i.e. the current game state as viewed by the server, after a certain time interval, which varies with the client's QoS. Thus, during a period of low QoS, instead of receiving many smaller event messages, the client receives one or a few larger update messages.

## 4 Results and Interpretation

Both the minimum size for the zone of influence, and the update interval were varied in the simulations to test the adaptation in a two-player environment. Baseline results were obtained from a standard version of the game, for comparison with the adaptive versions. Results were also verified using similar methods to those described in [26, 27].

```
Event received from Client

Server-side simulation state updated

IF Event has preset relevance {
    Event posted on all Client queues
}
ELSE {
    FOR EACH Client {
        IF Client has QoS of 1 {
            Event posted on this Client's queue
        }
        ELSE IF Client has QoS between 0 and 1{
            Calculate size of zone of influence
            IF Event occurs within zone of influence {
                Event posted on this Client's queue
            }
            ELSE IF Event occurs in the zone of zero influence {
                Event not sent
                Client put on timer for state update
            }
        }
        ELSE IF Client has QoS of 0 {
            Event not sent
            Client put on timer for state update
        }
    }
```

**Figure 3 Algorithm used to adapt events**

## 4.1 Methodology

The simulations were carried out on a host platform; the J2ME Wireless Toolkit v2.2, using MIDP v.2.0 [25], emulated the mobile device environment for the client software on two Dell Optiplex GX270 (Pentium 4 CPU running at 3GHz, with 1GB of RAM) desktop computers connected via a 100 MBps switched Ethernet LAN. A third such machine ran the server software (using J2SE 1.4.2_04 [25]).

Events were logged as and when they arrived and departed on all three nodes, along with their size, which client they pertained to, the time of arrival or departure, and the simulated QoS of the client at that time (NB: QoS values were recorded server-side only, since this is where their simulation occurred). The results used for analysis were based on the events sent and received from the perspective of the server, since this is where the adaptation took place. These results were verified by comparison of the recorded and captured logs, as well as comparison of logs from each node.

The two variables under scrutiny in these experiments were the basic size of the zone of influence, and the time interval between sending game state updates. Event streams, which are dependent on user activity, were controlled by being recorded and re-used under test conditions seen in table 1, including a standard game with no adaptation. These values were chosen to allow exploration of basic trends for future follow-up. Event logging, which was incorporated in the software, was validated by capturing the network traffic using Ethereal [28].

**Table 1.** Values used in the experiments

| *Test* | *Update Timer (ms; 0…System Max.)* | *Zone of Influence (basic size and max. extension) (movement units; 0…√200(max. distance))* |
|---|---|---|
| 1 | n/a (Standard version of game) | |
| 2 | 1000 | |
| 3 | 2000 | |
| 4 | 3000 | 4 |
| 5 | 4000 | |
| 6 | 5000 | |
| 7 | | 2 |
| 8 | | 3 |
| *4 (not repeated)* | 3000 | *4* |
| 9 | | 5 |
| 10 | | 6 |

## 4.2 Results Analysis

Performance was measured based on the comparison of results from the adaptive version of the game with those from the standard version and with the simulated client-connection QoS values. Following this, logs of events sent by clients and received by the server were used to compare the performance of the adaptation using identical data.

Results for the standard version of the game ("Test 1" in figure 4) show almost identical traffic patterns for the two clients; there is no differentiation between clients. This is as expected, since without adaptation, all events are processed by the server and sent on to all clients, without distinction. The only significant difference between the two is seen at session start, which can be attributed to the simulation setup phase. This phase is not synchronised by event, so a difference is not unexpected, and does not affect the rest of the stream, which the server distributes on a per event basis, producing the similar streams as in the charts.

Results from the adaptive versions of the game (Figure 5) show that the event streams sent to each client are distinct. Over time, the amount of data that is sent to each client changes, as is seen in the charts with the variations in gradient. Thus, differentiation between clients is possible with this technique. Comparison with the simulated QoS, however, shows no correlation. Re-using the event streams sent from the client to the server shows how the different test conditions affect the adapted

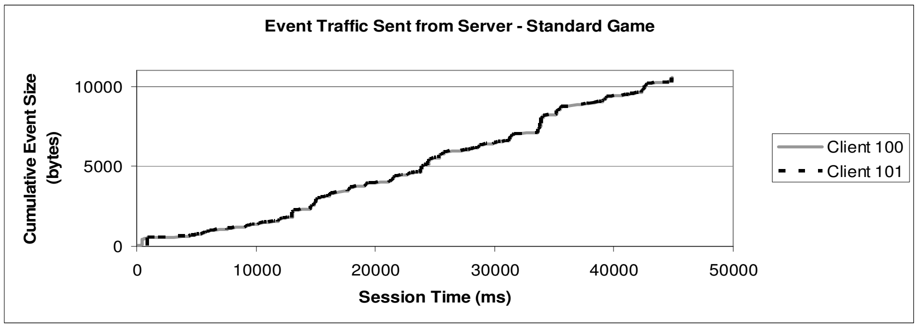

**Figure 4 Event traffic without adaptation**

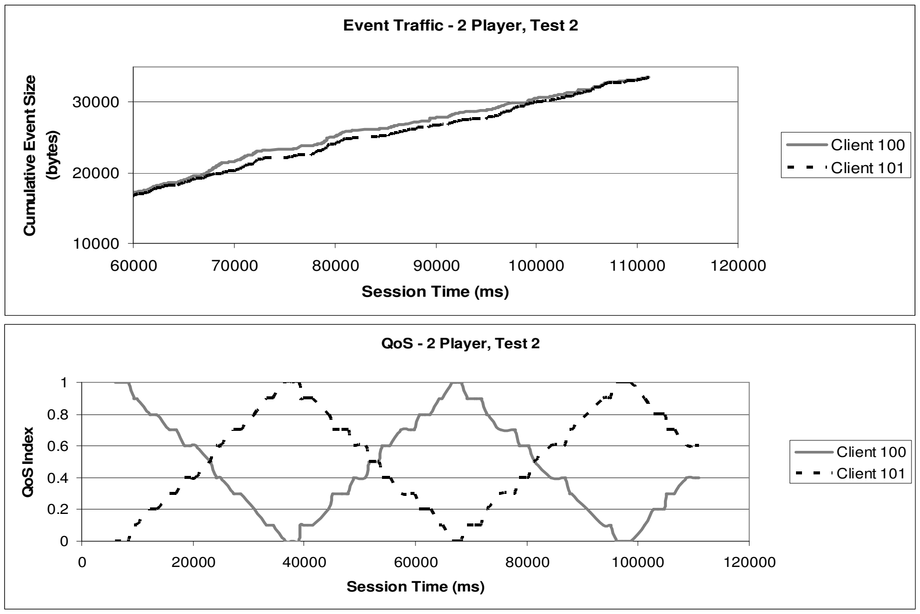

**Figure 5 Sample event traffic for adaptive game with simulated QoS**

event streams sent from the server. Figure 6 shows a sample chart of the event streams produced using the same data set, under the four test conditions representing the extremes of the two variables – Tests 2 and 6 for update interval, and Tests 7 and 10 for zone of influence. Typical of the results, this chart shows that the zone of influence has a greater effect on the event streams than the update interval. Figure 7 shows a summary of the results, based on the overall data rate. Again, these charts suggest that the zone of influence has a greater affect on the data rates than the update interval does. Successful adaptation, however, would show a correlation between low simulated QoS and low data rates, but this is not the case.

Because of the nature of the gameplay, the user follows the centre of activity around the game world. All events are initiated, directly or indirectly, by the users'

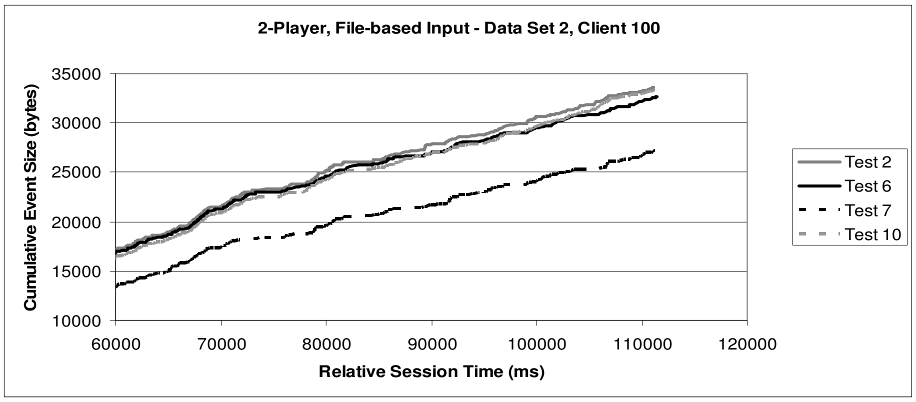

**Figure 6 Data set 2 under different test conditions**

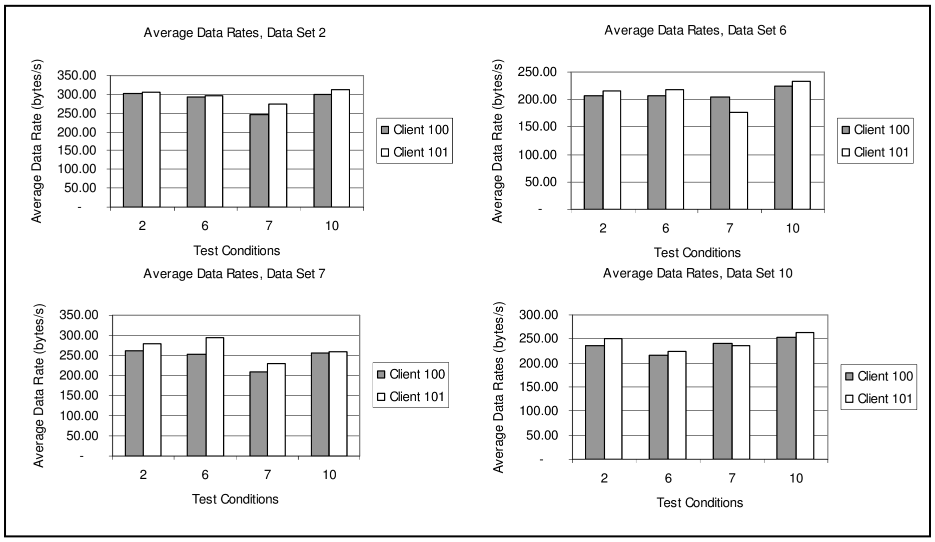

**Figure 7 Summary results for adaptive game**

intervention in the environment; furthermore, the users' avatars are likely to locate near each other, as the goal is to blow up the other's avatar. As such, the amount of activity outside the zone of influence is somewhat limited. This explains why larger zones of influence do not produce average data rates which are much different to those of a standard game, as the vast majority of events fall inside the zone and are thus sent to the client. An eleventh test condition was added to the simulations whereby the minimum size of the zone of influence and its maximum extension are no longer the same. In this test the variation in the size of the zone of influence is more closely tied to the QoS; where maximum QoS is available, the zone of influence covers the entire game world; when QoS is at a minimum, the zone of influence is reduced to the immediate vicinity of the player's avatar.

Overall, these results show better correlation with the simulated QoS than the previous test cases, but imperfections still remain. Figure 8 shows a sample chart of these results, showing a short session created by data set 6. This chart shows very similar traffic for the two clients for the first 13 seconds even though their QoS values are at opposite ranges of the scale, following the same pattern as seen in figure 5.

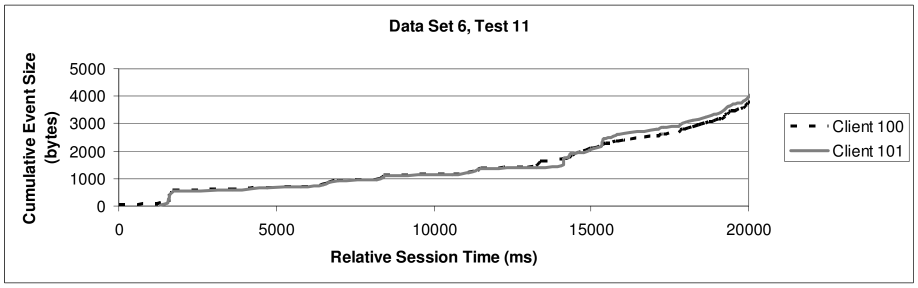

**Figure 8 Event streams for data set 6, test 11**

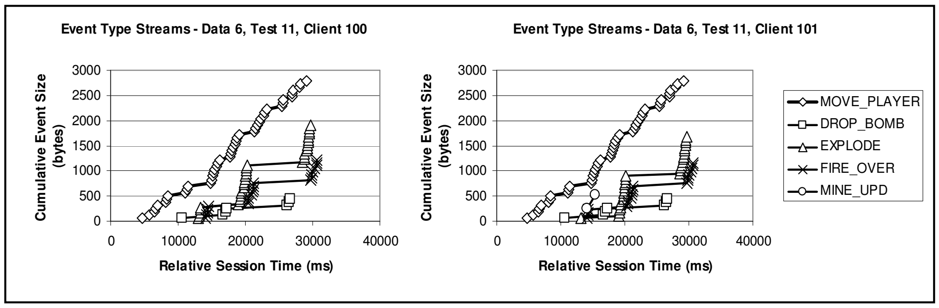

**Figure 9 Event type streams for data set 6, test 11**

Differences are clear between 13 and 16 seconds, but the event streams seem to return to similarity following this.

Examining the data streams for the adapted games shows the amount of activity within the zone of influence. Since events are sent directly if they fall within the zone, a large number of standard events would indicate a lot of activity within the zone, especially if the QoS were low. Figure 9 shows the individual traffic streams created by each event type on each client. These individual event type streams go together to form the overall event streams as seen in Figure 8, determining the shape of these overall streams. Analyzing the period between 13 and 16 seconds which was highlighted in the discussion on figure 8, shows that client 100, which has almost full QoS (see figure 6), receives no update ("MINE_UPD") events; this is as expected, since the size of its zone of influence covers most of the game world. Client 101, however, receives two update events and standard events in this period, indicating activity both within and without its zone of influence; client 101's zone of influence is quite small as its QoS is also quite small. In this situation, client 101 would have benefited from a quick update sent immediately, and not put on a timer, which would have demanded less bandwidth, rather than the series of smaller "EXPLODE" events and update events which demanded quite a lot. If this had been the case, client 101's overall event stream would have had a smoother gradient at this time, rather than the larger jump. Figure 10 shows how this would look.

This example suggests a lack of consideration for the QoS requirements of the various streams. In the present algorithm, the zone of influence shrinks and grows depending on the QoS available; the user receives high detail over time nearby in the virtual world, and varying accuracy in the middle and long distances. Rather than reducing the number of events sent to the client to lower the overall QoS requirement,

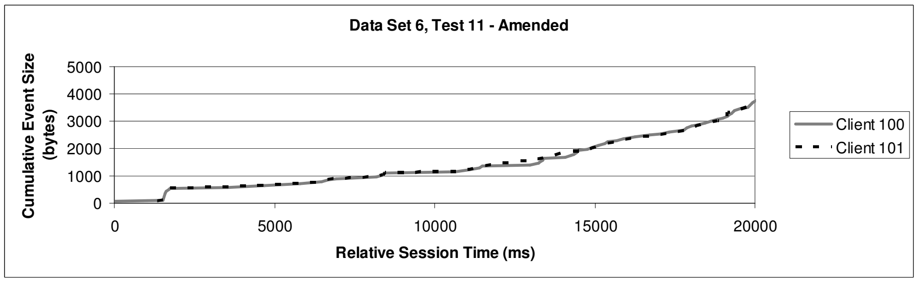

**Figure 10 Amended chart of data set 6**

it may be possible to use a different event message which has smaller QoS requirements, as just discussed. Potential results could mean lower QoS requirements for the overall stream, while maintaining a higher number of events. To achieve this demands knowledge of the QoS requirements of the different event types; this will require further experimentation in the future.

## 5 Conclusion and Future Work

This paper has presented an exploratory study of the use of data adaptation for the variable resource environment of wireless networking. Simulation suggests that in order for relevance-based adaptation to work optimally it must take the QoS requirements of the different event type streams into consideration. For DVEs, this means that there must be a method of describing events and event streams in terms of the resources they require, in order to provide a set of event stream solutions which have a range of QoS requirements. The decision of how best to adapt the basic event stream can then be based on such a set of solutions and the resources currently available. Issues involved in this continued work will examine how best to characterize QoS requirements using different metrics, as well as simulating the dynamic resource profile of wireless networking. Scalability and cost/efficiency analyses of the adaptation using different distribution architectures are also essential.

## References

1. Ward, Mark. “Mobile games poised for take-off.” BBC News. News, 2 May 2005. Available at http://news.bbc.co.uk/1/hi/technology/4498433.stm (17 May 2005).
2. Holden, Windsor. Juniper Research. White Paper, February 2005. Mobile Fun & Games - Second Edition
3. Facer, K, R Joiner, D Stanton, J Reid, R Hull, and D Kirk. “Savannah: mobile gaming and learning?” Journal of Computer Assisted Learning 20.6 (December 2004): 399-409.
4. Workman, SJH, G Parr, P Morrow, and D Charles. “Enabling Adaptive Multipoint Multimedia over Wireless IP Networks.” Proceedings from PGNET 2004 (28-29 June 2004), 266-271.
5. Yan, Bo, and Kam W. Ng. “A Survey on the Techniques for the Transport of MPEG-4 over Wireless Networks.” IEEE Transactions on Consumer Electronics 48.4 (November 2002): 863-873.
6. Ferretti, Stefano, and Marco Roccetti. “A novel obsolescence-based approach to event delivery synchronisation in multiplayer games.” International Journal of Intelligent Games and Simulation 3.1 (March/April 2004): 7-19.
7. Matijasevic, Maja, Denis Gracanin, Kimon P. Valavanis, and Ignac Lovrek. “A Framework for Multiuser Distributed Virtual Environments.” IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics 32.4 (August 2002): 416-429.
8. 3rd Generation Partnership Project (3GPP). Technical Specification Group Services and System Aspects. Technical Specification, March 2004. Quality of Service (QoS) concept and architecture (Release 6)
9. Busse, Marcel, Bernd Lamparter, Martin Mauve, and Wolfgang Effelsberg. “Lightweight QoS-Support for Networked Mobile Gaming.” SIGCOMM’04 Workshops (30 August 2004).

10. Lo, Shou-Chih, Guanling Lee, Wen-Tsuen Chen, and Jen-Chi Liu. “Architecture for Mobility and QoS Support in All-IP Wireless Networks.” 30 January 2003. IEEE Journal on Selected Areas in Communications 22.4 (May 2004): 691-705.
11. Moon, Bongko, and A. Hamid Aghvami. “Quality-of-Service Mechanisms in All-IP Wireless Access Networks.” 1 June 2003. IEEE Journal on Selected Areas in Communications 22.5 (June 2004): 873-888.
12. Mirhakkak, Mohammad, Nancy Schult, and Duncan Thomson. “Dynamic Bandwidth Management and Adaptive Applications for a Variable Bandwidth Wireless Environment.” December 2000. IEEE Journal on Selected areas in Communications 19.10 (October 2001): 1984-1997.
13. Mukhtar, Rami G., Stephen V. Hanly, and Lachlan L.H. Andrew. “Efficient Internet Traffic Delivery over Wireless Networks.” IEEE Communications Magazine (December 2003), 46-53.
14. Fu, Zhengua, Xiaoquo Meng, and Songwu Lu. “A Transport Protocol for Supporting Multimedia Streaming in Mobile Ad Hoc Networks.” 1 October 2002. IEEE Journal on Selected Areas in Communications 21.10 (December 2003): 1615-1626.
15. Akyildiz, Ian, Yucel Altunbasak, Faramarz Fekri, and Raghupathy Sivakumar. “AdaptNet: An Adaptive Protocol Suite for the Next-Generation Wireless Internet.” IEEE Communications Magazine, Volume: 42, Issue: 3, March 2004, pp.128-136.
16. Handley, M., V. Jacobson, and C. Perkins. “SDP: Session Description Protocol.” Internet Engineering Task Force. Stds. org, 27 October 2003. Available at http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdp-new-15.txt (9 February 2004).
17. Liscano, Ramiro, Allan Jost, Anand Dersingh, and Hao Hu. “Session-base Service Discovery in Peer-to-Peer Communications.” Proceedings from CCECE 2004-CCGEI 2004, Niagara Falls (May 2004).
18. Pangalos, Paul A., Konstantinos Boukis, Louise Burness, Alan Brookland, Caroline Beauchamps, and AH Aghvami. “End-to-End SIP Based Real Time Application Adaptation During Unplanned Vertical Handovers.” GLOBECOM '01. IEEE 6 (2529 November 2001): 3488-3493.
19. Guenkova-Luy, Teodora, Andreas J. Kassler, and Davide Mandato. “End-to-End Quality of Service Coordination for Mobile Multimedia Applications.” 1 June 2003. IEEE Journal on Selected Areas on Communications 22.5 (June 2004): 889-903.
20. Diot, Christophe, and Laurent Gautier. “A Distributed Architecture for Multiplayer Interactive Applications on the Internet.” IEEE Network 13.4 (July-August 1999).
21. Meier, René, and Vinny Cahill. “STEAM: Event-Based Middleware for Wireless Ad Hoc Networks.” Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW’02) (2002).
22. Eugster, P.Th., P. Felber, R. Guerraoui, and S.B. Handurukande. “Event Systems: How to Have Your Cake and Eat It Too.” Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW’02) (2002).
23. Hinze, Annika, and Sven Bittner. “Efficient Distribution-Based Event Filtering.” Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW’02) (2002).
24. Aarhus, Lars, Knut Holmqvist, and Martin Kirkengen. “Generalized Two-Tier Relevance Filtering of Computer Game Update Events.” NetGames 2002 (April 2002).
25. Sun Microsystems, Inc. “Java Technology.” Sun Developer Network. 2005. Available at http://java.sun.com (17 May 2005).
26. Borella, Michael S. “Source Models of Network Game Traffic.” Computer Communications 23.4 (February 2000): 403-410.
27. Färber, Johannes. “Traffic Modelling for Fast Action Network Games.” Multimedia Tools and Applications 23 (2004): 31-46.
28. “Ethereal: A Network Protocol Analyzer”. Available at http://www.ethereal.com/ (14 February 2005).