Volume 2, No. 1, Jan-Feb 2011
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

The Service Plan for Qos Mangment with The Help of Mobile Middleware and Video Sterming.

Supriya Agarwal
Electronics deptt.
SRMSCET
Bareilly,India
ersupriyaagarwal@gmail.com

Gaurav Agarwal*
Computer Science Deptt
Invertis University
Bareilly,India
gauravagarwal95@gmail.com

Saurabh Singh
Information Technology Deptt
Invertis University
Bareilly,India
saurabh.iiet@gmail.com

Mayank Kumar
Electronics and Instrumentation Deptt.
Invertis University
Bareilly,India
mayank865@gmail.com

Abstract: As middleware and component technologies lack support of QoS management, so we tried to propose a solution for how a mobile middleware can take up support the QoS management. In order to identify' the QoS mechanisms, that the middleware must manage on behalf of the application, we use a Video Streaming scenario and demonstrate the feasibility of our solution for this scenario. The key concept of our work is the service plan that specifies the service and the QoS of components and compositions. These service plans are user recursively, so that the QoS aware middleware platform can configure and reconfigure applications, based on user requirements and resource availability.

Keywords: Qos Management, Mobile Middleware, Video Streaming

I. INTRODUCTION.

In order to meet the increasing performance and scalability requirements, mechanisms that adapt to various workloads and requirements must be designed. It is also develop strategies for dynamically combining components into a high-performance service. The adaptation mechanisms are needed to maintain the Qos.The traditional way of handling this is to integrate QoS mechanisms with the application logic. This makes the application complex and hard to manage, and the implementation of QoS mechanisms cannot be reused in other applications. We follow an approach to separate the application logic from the domain specific QoS management, so that it becomes easy to reuse both application components and QoS mechanisms and also ensure safe reconfiguration (i.e. the service plan can be reconfigure according to the availability of resources , user requirements etc. safely) at runtime .

A. Introduction to Video Streaming to Mobile Terminal.

The Video servers are accessed by clients from different types of terminals like Laptops, Home Theatres etc, which are connected to the internet over access networks[1] like fixed LAN, Wireless LAN, GPRS etc. So, the main challenge lies in giving the users a high-quality playback in different contexts: home theater-LAN, laptop-LAN, laptop-WLAN etc.

B. Introduction to Dynamics and Constraints.

In the video streaming scenario, there are several challenges One is the traditional streaming requirement, timely delivery of high data rate streams. Another, more complicated, is the handling of the different combinations of access networks and terminal types.The QoS requirements like frame rate, resolution and color depth may differ from user to user. So, streaming the same content to users using the same technology may result in streams with different characteristics and requirements [3]. Hence, the application must be adapted according to the user's QoS requirements and the capabilities of all the resources along the data path from server to client.

C. Introduction to Middleware.

Middleware is computer software that connects software components or applications [4, 5]. The software consists of a set of services that allow multiple processes running on one or more machines to interact across a network. This technology evolved to provide for interoperability in support of the move to coherent distributed architectures, which are used most often to support and simplify complex, distributed applications. Middleware sits "in the middle" between application software working on different operating systems. It is similar to the middle layer of three-tier single system architecture, except that it is stretched across multiple systems or applications. Examples include database systems, telecommunications software, transaction monitors, and messaging-and-queuing software.

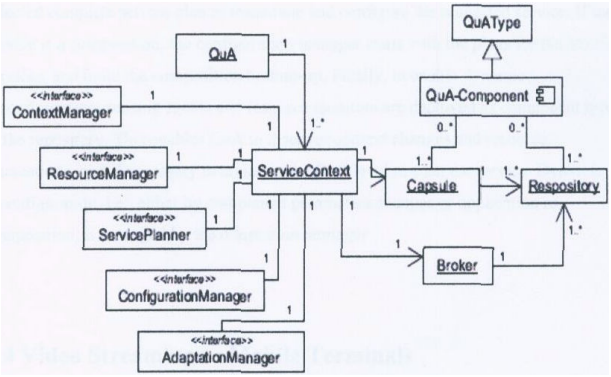

Figure 1 Middleware overview.

D. Introduction to QoS.

In the field of computer networking and other packet switched telecommunication networks, the traffic engineering term quality of service (QoS) refers to resource management mechanisms rather than the achieved service quality. Quality of service is the ability to provide different priority to different applications, users performance, or data flows, or to guarantee a certain level of performance to a data flow. For example, a required bit rate, delay, jitter, packet dropping probability and/or bit error rate may be guaranteed. Quality of Service (QoS) for networks is an industry-wide set of standards and mechanisms for ensuring high-quality performance for critical applications [4, 5].

E. QoS Aware Mobile Middleware.

Middleware and component technologies lack support for Quality of Service (QoS) management [2]. Application developers, therefore, integrate QoS mechanisms into the application itself. In this paper, we propose a solution for how a mobile middleware can take on the responsibility for QoS management- We use a video streaming scenario to identify the QoS mechanisms that the middleware must manage on behalf of the application, and we demonstrate the feasibility of our solution within this scenario.

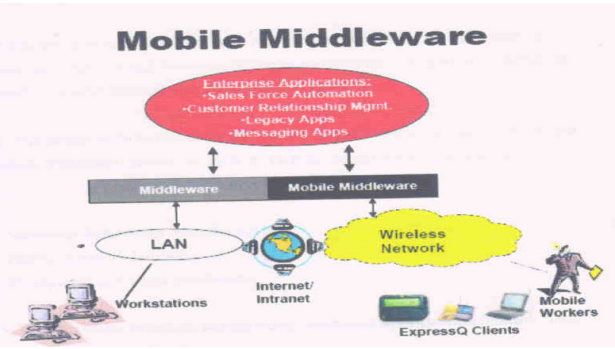

Figure 2 Mobile Middleware

F. Introduction to Video Streaming to Mobile Terminals.

The Video servers are accessed by clients from different types of terminals like Laptops, Home Theatres etc, which are connected to the internet over access networks like fixed LAN, Wireless LAN,GPRS etc. So, the main challenge lies in giving the users a high-quality playback in different contexts: home theater-LAN, laptop-LAN, laptop-WLAN, PDA-WLAN, PDA-GPRS, when network conditions are changing and users roam between access networks. Streaming video is content sent in compressed form over the Internet and displayed by the viewer in real time. With streaming video or streaming media, a Web user does not have to wait to download a file to play it. Instead, the media is sent in a continuous stream of data and is played as it arrives. The user needs a player, which is a special program that uncompressed and sends video data to the display and audio data to speakers. A player can be either an integral part of a browser or downloaded from the software maker's Web site.

II. PROPOSED SOLUTION

It is already defined that the middleware should select and combine the most appropriate components for a given context (e.g., access network technology, execution environment, and terminal type) and available resource capacity (e.g., CPU, storage, and network). If a terminal stays connected to the same network, the initial configuration will remain fixed during the whole session. In order to maintain the best possible QoS. In general, there are two adaptation types that must be supported

[a] changing parameter settings of individual components
[b] changing the application composition

Each application variant, resulting from performing one of these types of adaptation, is called an application configuration. We also recognize that some QoS mechanisms are so tightly coupled with the corresponding functionality that separating them out would result in a very complex design.

One example of such a mechanism is data traffic control, which constantly monitors round trip time and packet error rates, to adjust the transmission rate. Consequently, our approach to handle the dynamics in the scenario is to let the middleware control QoS to the extent feasible, and to allow component self-adaptation where necessary. This requires that the QoS characteristics is specified for each component, which then are used by the middleware to assess the suitability of each component when assembling a composition. For self-adapting components, it is important that the ability to self-adapt is expressed and quantified in the QoS characteristics. The designing phase consist of

[i] Defining components.
[ii] Specifying compositions.
[iii] Specifying parameter configurations.
[iv] Service modeling.
[v] QoS modeling.

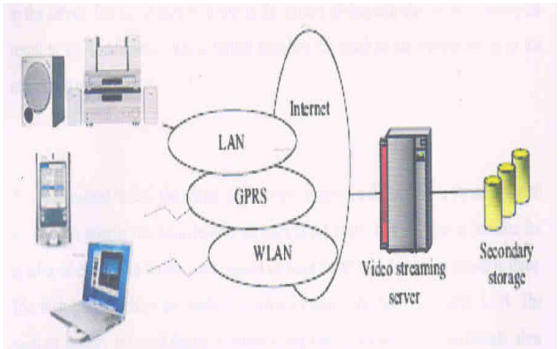

Figure 3 System overview

A. Model to be simulated.

Through dynamic reconfiguration of the model better results and an even more optimized scenario for the same can be obtained [2]. Therefore, we have tried to safely reconfigure the initial model for three clients, using Omnet++. Omnet++ is a tool that enables simulation with Visual C++ as a background tool. OMNeT++ is an object-oriented modular discrete event network simulator. The simulator can be used for:

[a] traffic modeling of telecommunication networks
[b] protocol modeling
[c] modeling queuing networks
[d] modeling multiprocessors and other distributed hardware systems
[e] validating hardware architectures
[f] evaluating performance aspects of complex software systems

B. Simulated Model.

In our simulated model, we have intended to depict the working, wherein; there are clients, sending requests to work on a network. The middleware acts as a mediator. As the name suggests it takes input in the form of requests from the various clients, and passes the requests to the classifier for the determination of the network available for the particular client, according to the priority. From the middleware the request is then passed to the classifier, which on the basis of the priority assigned to the client selects a network for it. The choice of network lies from the very high speed wireless LAN ,to the lower speed GPRS ,to the slowest LAN.Now, through the internet the request of the client is passed from the particular network to the server.

The server then responds to the request of the particular client. It passes the result to the middleware, which further transfers the result to the corresponding or the requesting client.In the simulated model, the clients depict either a laptop, a desktop, or a personal digital assistant. A priority has been decided for each of the client. It is possible to increase the number of clients, but for our convenience we have limited the number of clients to three. The high priority client are enabled to access through the fastest Wireless LAN. The medium priority personal digital assistant is required to access the comparatively slow GPRS, whereas the common pool or the lowest priorities are enabled to use the slowest LAN. This classification of network is done by the classifier. After the classification has been done the request travels through the overall network or the internet.

The server is where the requests are received and processed so as to produce the final or rather the desired results as demanded by the client. The working for the various applications is stored in the middleware.

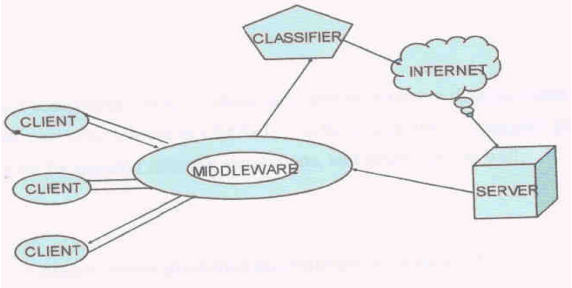

Figure 4 Simulated models

C. OMNeT++ simulation model look like.

OMNeT++ provides a component architecture. Models are assembled from reusable components, modules. Well-written modules are truly reusable and can be combined in various ways like LEGO blocks.Modules can be connected with each other via gates (other systems would call them ports), and combined to form compound modules. Connections are created within a single level of module hierarchy: a sub module can be connected with another, or with the containing compound module. Every simulation model is an instance of a compound module type. This level (components and topology) is dealt with in NED files. To give you an idea, a component named Ether MAC would be described in NED like this:

```
//
// Ethernet CSMA/CD MAC
//
simple EtherMAC
        parameters:
        address : string; // others omitted for brevity
gates:
        in: phyln;   // to physical layer or the network
        out: phyOut; // to physical layer or the network
        in: llcln;   // to EtherLLC or higher layer
        out: llcOut; // to EtherLLC or higher layer
endsimple
```

And it could be used in the model of an Ethernet station like this:

```
//
// Host with an Ethernet interface
//
module EtherStation
        parameters: ...
        gates: ...
                in: in;   // for connecting to switch/hub, etc
        out: out;
submodules:
        app: EtherTrafficGen;
        llc: EtherLLC;
        mac: EtherMAC;
connections:
        app.out --> llc.hlln;
        app.in <-- llc.hlOut;
        llc.macln <-- mac.llcOut;
        llc.macOout --> mac.llcln;
        mac.phyln <— in;
        mac.phyOut —> out;
        endmodule
```

Comments are useful in documentation generation via opp nedtool; see an example here). Simple modules which, like Ether MAC above, don't have further sub modules and are backed up with C++ code that provides their active behavior are declared with the simple keyword; compound modules are declared with the module keyword. To simulate an Ethernet LAN, you'd create a compound module Ether LAN and announce that it can run by itself with the network keyword:

```
module EtherLAN
        ... (sub modules of type Ether Station, etc)...
endmodule

network etherLAN : EtherLAN
endmodule
```

III. CONCLUSION

In the paper we have discussed about the Video Streaming scenario and demonstrate the feasibility of our solution.The concept of our work is the service plan that specifies the service and the QoS of components and compositions.

IV. REFERENCES

[1] Charles Perkins. IP Mobility Support for IPv4. IETF RFC 3344, August 2002.

[2] Amundsen, K. Lund, C. Griwodz, and P. Halvorsen. Scenario Description- Video Streaming in the Mobile Domain. http://www.simula.no:8888/ QuA/2/techV ScenAl.pdf,2005..

[3] OMG. UML Profile for Modelling Quality of Service and Fault Tolerance Characteristics and Mechanisms. OMG Adopted Specification, June 2004.

[4] R. Staehli, F. Eliassen, and S. Amundsen. Designing Adaptive Middleware for Reuse. In Proceeding from 3rd International Workshop on Reflective and Adaptive Middleware, 2004.

[5] S. Amundsen A. Solberg, J. 0. Aagedal, and F. Eliassen. A Framework for QoS-Aware Service Composition. In Proceedings of 2nd ACM international Conference on Service Oriented Computing 2004.

[6] Simula Research Laboratory. QuA documentation. http://www.simula.no:8888/QuA/55,2004

[7] S. Amundsen, K. Lund, F. Eliassen, and R. Staehli. QuA: Platform-Managed QoS for Component Architectures. In Proceedings from Norwegian Informatics Conference (NIK)**,** November 2004