The Effects of Mobile Agent Performance on Mp3 Streaming Applications

Binh Thai[1], Aruna Seneviratne[2]

[1] School of Electrical Engineering and Telecommunications,
University of New South Wales, Sydney, Australia
binh@ee.unsw.edu.au

[2] School of Electrical Engineering and Telecommunications,
University of New South Wales, Sydney, Australia
a.seneviratne@unsw.edu.au

Abstract. This paper considers the use of software mobile agents as proxy agents, to provide seamless mobility and cater for heterogeneous devices and network characteristics. We specifically consider the use of mobile agents within the Mobile Aware ARCHitecture (MARCH) framework [8]. We show how a *mobile proxy agent* can be used within the March Framework and we try to evaluate the impact of agent migration when it is used as a relay proxy for an mpeg layer 3 audio streaming application. We first try to isolate the agent migration contribution to the audio loss and we propose a possible solution to minimise this audio loss.

1 Introduction

Mobile Agents are an increasingly popular technology in many research areas such as information retrieval, distributed computing, and network management. This increase of interest could be due the basic characteristics of mobile agents – *autonomous* and *mobile*. Studies that exploit such characteristics include De Meer et al's QoS management [3] and Krulwich's user surrogates [11]. In this paper, we propose to use mobile agents in Internet environments, as content transcoding proxies. We show how a *mobile proxy agent* can be used in a proxy based network architecture such as March [8], and we try to evaluate the migration cost when used in a mp3 audio streaming application scenario.

The March framework addresses the problems associated with client and access network heterogeneity, and the static nature of most content available on the Internet. While the trend is to have *all* content format (eg. html, real media, mp3) available on *all* devices types (eg. desktop, PDA, mobile phones), there still exists many unresolved issues. Indeed not only the media presentation capabilities of these terminals differs widely, but also the network access technologies have various capacities. We believe that maintaining several copies of the media at the server to cater for each situation is impractical. The March framework attempt to address this issue by using proxy agents dynamically placed in the network.

We believe using mobile agents as the code mobility foundation in the context of the march framework has many advantages: first, mobile agents could perform resources management functions by deciding *autonomously* to move from one machine to another, when resources (CPU, network) requires it to. Another is to provide *local mobility*: when a mobile node moves from one network to another, the first-hop proxy can help the mobility by providing content-caching and follow the movement of the mobile host. In this paper, we do not describe the mechanisms mobile proxy uses to decide *when* to and *where* to move. We rather try to evaluate the limitations of using agent technologies, in the context of an audio streaming application.

To investigate this cost, we devised two experiments, both mirrors the situation at which the client is changing its network access point while receiving a mp3 stream from the content server. The first experiment involved using one mobile proxy agent, the second experiment involved using a technique of agent cloning to handle the migration process. We show that the quality of the audio only degrades slightly during the migration period.

The paper is organised as follows: Section 2 briefly introduces the March framework. Section 3 reviews related work which measures the performance of various mobile agent platforms, and Section 4 describes our experiment and presents our findings. Section 5 presents our conclusion and describes our future work.

2 The March Framework

Clients today can have different devices with different capabilities. They can also have access to different type of network access, which have different characteristics. It is difficult for a content provider to provide contents that can cater for all the clients on the network. The March framework address this issue by using application-level proxies that performs functions such as caching, protocol conversion, and content adaptation. There are various studies which utilise such an architecture, Fox et al’s work on quality transcoding [4] and De Silva et al's TOMTEN [5] are just two examples.

A benefit of using network proxies is that they permit clients or servers, or both, to remain unchanged or to be changed very little when placed in new network environments. The proxy should look like a server to the client and as a client to the server. This makes a proxy solution far easier to deploy than a solution requiring changes to clients, server or routers. Proxies are likely to be found near the edges of the network, close to performance discontinuities such as the entrance to a wireless link. Then they could provide additional functionality for the wireless hop. The viability of proxy based solutions has been demonstrated by commercial products such as Web-on-Air [12].

The classical proxy based solution requires that clients are configured manually or at installation time, and it is static. The solution is client centric, i.e. the client (a) makes the decision about which proxies to use, (b) installs and (c) configures them. The server, at best is semi active, dealing with changes in coding and/or rate. However, especially in the client/server architecture, the server has significantly more knowledge about the application and the operational environment than the client.

Thus the objective of our MARCH project framework is to investigate the possibility of using information available at the server to make more informed choices about customisation of proxies and to automate the process. More importantly, the classical proxy solution introduces serious end-to-end security problems, which MARCH avoids.

Many proposals in the active network area such as [1], [7] amongst others, investigate dynamic proxy placements systems for content distillation or protocol enhancement. However, MARCH differs as it is the only proposal that uses knowledge at the server to make the decisions as to what functionality should be used for a particular situation, and where the proxies should be placed.

Our framework is illustrated in Fig. 1.

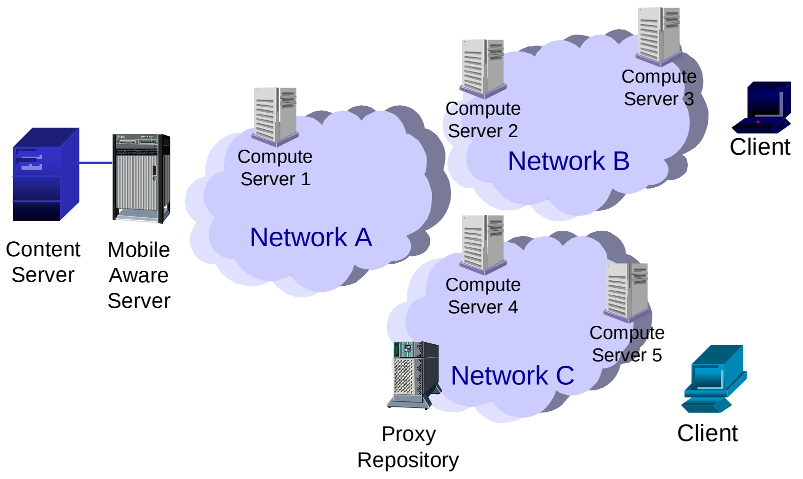

Fig. 1. The Mobile Aware ARCHitecture.

The Mobile Aware Server (MAS) is the entity in which performs most of the decisions. The client devices incorporate a Mobile Client Entity (MCE), which is responsible for transmitting the client device capabilities, user's preferences and network type to the MAS. In some cases, the MCE could be optional or located elsewhere, e.g. if the client is a WAP device, the terminal characteristics could be obtained at the WAP gateway. Another example is to use the *user-agent* feature of common web-browser to gather more information about the client's capabilities.

Before a session is being initiated by the MAS there has to be a client-server interaction to obtain information about the performance status of the network, the client terminal characteristics and to collect user preferences. The MCE will provide details about the terminal, where to find user preferences and the characteristics of the access network the client device is attached. MAS will use this information together with the information it has about intermediate network domains and its knowledge about the application to choose the proxy functionality to be used along the data path and decide where these proxies should be installed.

MAS then need to signal the compute servers that will house the proxies to retrieve, to install and to configure them.

The MAS will request the chosen compute servers to obtain the required proxies modules from the Proxy Repositories (PREP). Proxies at the server and the client will be installed by the MAS. The concatenation of all proxies between server and the client will form the enhanced session, customised to fit the characteristics of interconnecting networks, the client type, and the requirements of the user.

In case of a secure session, e.g. a banking application, the PREPs and compute servers may be situated at bank branches in order to create "secure enclaves". At these enclaves, secure data can be decrypted, transformed to another format and encrypted for further delivery to the client or server.

Different technologies could be used to implement the dynamic proxy placement function of the MARCH framework. One of them is to use mobile agents. Using mobile agents allow extended functionalities and optimisations. For example once the session is established, if the network conditions degrades, or the CPU resources on the compute server become scarce, a mobile agent could autonomously take the decision to move to a different nearby compute server. This also brings more robustness in the system as it delegates the responsibility for *local issues*, to the proxies themselves.

3 Related Work

There are various studies that concentrate on the performance of mobile agent platforms, and experiments were devised to measure such quality. Silva et al. [17] extensively tested 8 different mobile agent platforms and compared the performance and robustness of each one: Aglets, Concordia, Voyager, Odyssey, Jumping Beans, Grasshopper, Swarm and James. 12 experiments were performed on the above mentioned mobile agent platforms. The experiments include the measuring execution time of a sample application with varying agent size, the caching capabilities, and the amount of network traffic generated during agent migration.

Silva el al. discovered that Concordia, Voyager and Jumping Beans have problems in relation to stability. They also showed that James provided the highest performance, as the authors suggested that this particular mobile agent platform was designed with various optimisation mechanisms.

From our perspective, we can use this study as a guideline on which mobile agent platform has the best performance. However, we cannot use this study to decide the viability of the use of mobile agents in our framework, because the scenarios used in their experiments do not bound to any particular frameworks.

Less extensive performance experiments were performed by Narasimhan [13], [14], [15]. The author concentrated specifically on the migration performance of IBM Aglets. She suggested that there are three factors that can affect the time that an aglet requires to migrate from one host to another: network congestion, geographical location of servers and the size of the aglet.

The first experiment Narasimhan performed was measuring the round trip time of an aglet with various sizes. The second experiment was measuring the round trip time of aglet as it travels around the world via 4 different hosts located in USA, Italy and Japan. Both experiments reflect a typical scenario of which a mobile agent travels around the world to gather data for the user. The results of the experiments can be found in [15] and [14].

Due to Narasimhan's assumptions, and the methods the experiments were conducted, the results obtained from the experiments cannot provide us with any insight on the real migration performance of IBM aglets.

Moreover, the study of Silva et al and Narasimhan. were aimed at measuring the performance of mobile agents without considering the specific application of the technology. Therefore the results they presented are only the performance of the mobile agent in a general case. How this performance affects a streaming application when a mobile agent is acting as a proxy remains unknown.

A more analytical approach of investigating the performance of mobile agents was performed by Kotz et al [10]. They developed a mathematical model based on different data streams arriving to a common gateway, which are then transmitted to the mobile devices via a wireless channel. Kotz et al proposed the use of mobile agents, originating from the client's device, to migrate from the device to the common gateway. The mobile agent then performs data filtering at gateway as the data streams arrive from the Internet. This proposal was compared against the technique of performing data filtering at the mobile device.

4 Mp3 Streaming Experiment

Two experiments were devised to determine how the migration of the mobile proxy agent affects a mp3 streaming audio session. The equipment we used for the experiments are as follows:

- 5 Pentium II 400 PC's. Each with 128Mb of RAM, running Linux 2.2.x. These PC's are connected together on a 10Mbps Ethernet.
- IBM aglet version 1.0.3b [9] was used as our mobile agent platform.
- Blackdown Java Linux port JDK 1.1.7b ver 3 [2] was used as our JVM. All the PC's had the software installed in exactly the same way.

One PC was chosen as the mp3-streaming server over RTP. The software we used to perform the streaming was Obsequieum version 2.1.7 [16]; however, we had to modify the original source code to cater for our prototype, since Obsequieum was designed to perform non-stop multicasting over the Internet.

Another PC was assigned as the client. The client software we used to listen to the mp3 stream was freeamp version 2.0.7 [6]. We chose this particular server-client application because unlike other streaming mp3 applications, which use TCP as their streaming protocol, Obsequieum streams mp3 audio using RTP.

The remaining 3 PC's are assigned as follows: one was assigned as the proxy repository. This is where the mobile proxy agent resides. The other 2 PC's are assigned as compute server 1 and 2 respectively. All of these nodes use *Tahiti* as the aglet server. This is the default aglet server.

The complete system is illustrated in Fig. 2.

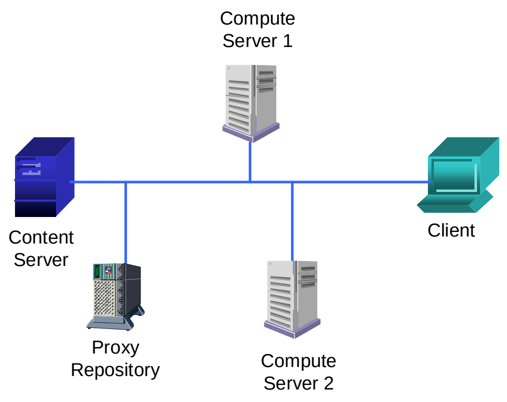

Fig. 2. The prototype framework used for the experiment.

A mobile proxy agent was developed using IBM aglet. The proxy function is to pass the received UDP packets from the mp3 streaming server, extract the timestamp from the RTP packet header, display it on the screen, and then relay this packet to the client. The extractions of the time stamp from the RTP packet header enables us to determine the amount of audio is loss during the migration of the mobile proxy.

In term of the size of the mobile proxy agent, the mobile proxy agent consists of 3 classes – a class which extends aglet, a class which extends thread and a class derived by ourselves to handle all the signalling of the system. The size of the bytecode for this mobile proxy agent totals to 10207 bytes.

The system is completed with various middleware at the server and client sides to allow for the migration of the mobile proxy agent. We did not modify the application itself to suit our framework, as it contradicts the philosophy behind any proxy based network architectures.

The quality of mp3 audio we used was 44.1kHz music streaming at 128kbits/s. Higher bit rate was possible; however, 128kbits/s is a common bit rate at the moment. Since there are no transcoding of audio involved in our experiment, we defined the degradation of Quality of Service (QoS) as the loss of audio. As mentioned previously, the proxy agent is designed to display the timestamp of each RTP packet it receives. The amount of audio loss during its migration is calculated by taking the difference between the *first* timestamp *after* migration and *last* timestamp *before* migration. This time loss includes the aglet's suspension of execution, object serialisation, bytecode transfer across the network, object deserialisation, resumption of the aglet, and the signalling to the server's middleware to redirect the mp3 stream to a different compute server.

Since freeamp also has the capability of buffering streams, we also investigated the affect of different buffer size at the client application.

4.1 Experiment 1: Simple Mobile Agent Migration

With this first experiment, we tested the amount of audio loss with the most primitive approach. As the session begins, a mobile proxy agent resides in compute server 1. After a period of 20 seconds, the proxy agent migrates to compute server 2. The process continues back and forth between the two compute servers for a total of 200 times.

Initially the amount of buffering of the stream at the client was set to 3 seconds – this is the default value for freeamp. We repeated the experiment with buffering set to 0 seconds, ie. no buffering. The results are presented in Table 1.

Table 1. The results of Experiment 1.

freeamp buffering	av. RTP packets dropped	s.d. RTP packets dropped	av. audio loss	s.d. audio loss
3 sec	3.89	1.22	248.78ms	78.21ms
0 sec	3.71	0.82	237.53ms	52.63ms

From the results shown in Table 1, it appears that the amount of buffering has no significant effect on the amount of audio loss as the mobile proxy agent migrates from one compute server to another. Indeed, since the amount of audio loss we could measure is *the amount of audio loss on the network*. In terms of the degradation of quality, this loss equates to a very short skip of audio.

With buffering turned on, as the mobile agent proxy migrates from one compute server to another, the buffer is emptied due to the loss of audio on the network – about 250ms worth. If the buffer is not empty, the audio loss is translated to a short skip. However, if the buffer becomes empty, the application stops playing the audio altogether, and refills its buffer before it continues. When this situation occurs, the user experiences a silence of 3 seconds. Compared to the 250ms of audio interruption as mentioned previously, this 3 seconds silence is less acceptable from the user's perspective.

4.2 Experiment 2: Mobile Agent Migration Using a Cloning Scheme

In the second experiment, we introduce a handover mechanism using the mobile agents cloning ability, to reduce the audio loss when migration occurs. Instead of migrating a single agent between two compute servers, the mobile proxy agent clones itself while maintaining the data stream to the client. The clone migrates to another compute server, and signals to the server's middleware to change its data stream. The original proxy agent then destroys itself.

Once again, different buffering sizes were tested – 3 seconds and 0 seconds. The results of this experiment are presented in Table 2.

Table 2. The results of Experiment 2.

freeamp buffering	av. RTP packets dropped	s.d. RTP packets dropped	av. audio loss	s.d. audio loss
3 sec	1.27	0.45	81.38ms	28.56ms
0 sec	1.24	0.51	79.26ms	32.81ms

From the results shown in Table 2, we can see that with the use of the cloning technique, the number of RTP packets dropped is significantly reduced compared to the results obtained from experiment 1. Since the amount audio loss is so small, the effect of buffering was not noticeable – freeamp did not have to refill its buffer at all throughout the entire experiment.

5 Conclusions and Future Investigations

In this paper, we have reviewed the concept of our MARCH framework, which provides dynamic application-level dynamic proxy placement into various compute servers on the network. Since the server has more knowledge about the characteristics of the content, our framework places the intelligence at the server, with the server making the decisions on the customisation for proxies and automating the processes.

We considered the concept of mobile agents as one of the possible solutions to implement the dynamic proxy placement function of the MARCH framework. Being autonomous and mobile, the introduction of mobile agents in our framework allows the network conditions and the CPU resources of the computer servers to be evenly distributed. This is possible as a mobile agent can automatically detect the condition of the compute server's CPU resource, or the network condition around the compute server, and then autonomously make a decision on whether it should migrate to a nearby compute server.

To investigate the viability of mobile agents in our framework, we performed two experiments that mirror a typical mp3-streaming session. The results showed that with different migration policies, the resulting QoS could vary.

The viability of using mobile agents as dynamic proxies in the MARCH framework is still in the early stages, and there are a number of issues we have yet to address:

- The placement of proxy – The mobile aware server requires to select a compute server in the network to place a proxy agent. The location of this compute server must be such so that it can provide the user with the optimum performance. The algorithm of which the mobile aware server uses to make such a decision is not yet developed.

- The mobile proxy agent migration criteria – We showed in this paper that the migration of a mobile agent proxy does not affect greatly to the quality of audio in a mp3 streaming session; however, the criteria of which when should migration occurs is yet to be investigated. Since there are several conditions that migration is required, for example, CPU of the compute server is over utilised, a mobile proxy agent should be able to detect the change in its surrounding environment and make a decision on the next compute server of which it should migrate itself to.

Acknowledgments

The authors would like to acknowledge Ericsson Radio Systems AB, Sweden and Ericsson Australia for their financial support.

References

1. Amir E, McCanne S, Katz R, “An Active Service Framework and its Application to Real-time Multimedia Transcoding”, ACM Computer Communication Review, vol. 28, no. 4, pp.178-189, Sep. 1998
2. Blackdown Java-Linux homepage, http://www.blackdown.org
3. De Meer H et al, “Programmable Agents for Flexible QoS Management in IP Networks”, *IEEE Journal on Selected Aeras In Communication*, Vol. 18, February YEAR, p.256-267.
4. De Silva R, Landfeldt B, Ardon S, Seneviratne A, “Total Management of Transmissions for the End-User, A Framework for User Control of Application behaviour”, HIPPARCH, London, 1998.
5. Fox A, Gribble S, Chawathe Y and Gribble E, “Adapting to Network and Client Variations using Infrastructional Proxies: Lessons and Perspective”, *IEEE Personal Communications*, August , 1998.
6. Freeamp homepage, http://www.freeamp.org
7. Fry M and Gosh A, “Application Level Active Networking”, Fourth International Workshop on High Performance Protocol Architectures (HIPPARCH ’98), June 98
8. Gunninberg, P. and Seneviratne A. “Services and Architectures in the next generation Internet using dynamic proxies”, FTF99, Bejing China, December 1999.
9. IBM Aglets homepage, http://www.trl.ibm.com/aglets
10. Kotz D et al, “Performance Analysis of Mobile Agents for Filtering Data Streams on Wireless Networks”, to appear in MSWiM 2000, Boston USA, 2000.
11. Krulwich B, “Automating The Internet: Agents as User Surrogates”, *IEEE Internet Computing*, July/August 1997, p.34-38.
12. Ludwig R, Niebert N, Quinet R, “Radio Webs – Support Architecture for Mobile Web Access”, International Conference Distributed Multimedia

13. Narasinhan N, “Experiments To Evaluate Aglet Latency”, http://alpha.ece.ucsb.edu/~nita/agletsExpt/index.html
14. Narasinhan N, “Latency for Aglet Travel Around The World”, http://alpha.ece.ucsb.edu/~nita/agletExpt/agletsExpt2.html
15. Narasinhan N, “Variation in Aglet Latency with Aglet Size”, http://alpha.ece.ucsb.edu/~nita/agletExpt/agletsExpt1.html
16. Obsequieum, homepage, http://obs.freeeamp.org
17. Silva L M et al, “Comparing the performance of mobile agent system: a study of benchmarking”, *Elsevier Computer Communications*, Issue 23, p.769-778
18. Thai B, Senevirante A, “The Use Software Agents as Proxies”, *Proceedings of ISCC 2000*, p.546-551