RULE-BASED EXPERT SERVER SYSTEM DESIGN FOR MULTIMEDIA STREAMING TRANSMISSION

ZHOU XIAOFEI
B.Eng.(Hons.), HUST

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

RULE-BASED EXPERT SERVER SYSTEM DESIGN FOR MULTIMEDIA STREAMING TRANSMISSION

ZHOU XIAOFEI

NATIONAL UNIVERSITY OF SINGAPORE

2008

Acknowledgements

I would like to express my deepest gratitude to all those who have directly or indirectly provided advice and assistance for my research work and study here in the National University of Singapore. This project would not have been possible without the support of those people.

Many thanks to my supervisor, A/Prof Ong, who has led me to the proposal of this project, for his valuable guidance, suggestions and support throughout the research. During times of difficulties, he always showed much understanding and patience and gave me warm encouragement. I have learned a lot from him and I am really glad that I have come here and became one of his students.

Thanks to Mr. Pan Yan for his valuable advice and friendly help. His occasional discussions around my work and interesting suggestions in operations have been very helpful for this study.

Thanks to my parents, for their invaluable love.

Finally, thanks to my friends who endured this long process with me. Thanks for their support and help.

Table of Contents

List of Tables and Figures

List of Tables:

List of Figures:

Abstract

The presented research is a pioneering work that applied an expert system into the multimedia streaming server and evaluated the server performances. The purpose is to make current streaming servers more powerful on streaming, flexible on control, and reliable on maintenance.

In this thesis, we presented the detailed design and theoretical analysis of the expert server. The time complexity of inference procedure was analyzed and the real time characteristics of the server were discussed. Although the server performances depend largely on the effectiveness of the rules, which is a knowledge database that linked to the main body of the server, we can make reasonable estimations on the performances by studying parameters like inference complexity and request response time. Based on these estimations, server capacity was deduced with respect to the maximum number of clients supportable.

The expert streaming server performance was evaluated with a group of congestion control algorithms on a local area network, compared with Apple's QuickTime Streaming Server (QTSS). Results showed that the expert system can perform effective admission control and distribute traffic reasonably among servers according to the server load and link parameters. It could automatically chops the movie and inserts advertisements based on client profile and content provider's profile. For high-definition movies, it could deliver smoother streams with around 60% reduced throughput oscillations when compared with the basic QTSS. The saved 60% bandwidth could be used for supporting more users. The expert control could also switch the playing movie between different devices without interruption. Such an intelligent handover is a promising technology when nowadays terminal devices become more various

and smart. If congestion happened, the expert system reacts based on the severe of the congestion and conducts cooperative steps to response. Afterwards, the congestion related information was recorded and referred by future congestion avoidance of the stream. Attractively, these enhanced performances were achieved by taking less than 10% of the CPU time for the execution of the expert control program.

To enhance the completeness of the knowledge base in our expert server system, we also designed a client oriented rate control scheme by solving the Discrete Linear Quadratic (DLQ) regulator problem under disturbances. Our study showed that DLQ was superior to conventional rate control schemes especially in maintaining high level and stable client buffer utilization. Besides the basic DLQ method, we also investigated the performance of DLQ under delay.

The limitations and the future development directions are given in the conclusion part of this thesis. We are expecting that the expert server would become a practical, flexible, and robust platform of multimedia streaming transmission. If it is developed as described in the conclusion, it could be a valuable model for future integrated multi-function media server.

Chapter 1 Introduction

In this chapter, we will concentrate on the background information of the rule based expert system and the multimedia streaming. They are two major scopes of the study presented in this thesis.

Being a practical artificial intelligence (AI) approach, expert systems try to imitate the intellectual behavior of human beings. Specifically, the expert system intends to emulate the problem-solving ability of a human expert. To achieve this, it maintains a knowledge database of heuristic and theoretical knowledge for the computer to perform a reasonable inference. A rule-based expert system is a significant branch of the expert system family. The knowledge database in the rule-based expert system is realized using the rules. Each rule represents a piece of expert knowledge to a particular problem and all rules are grouped and linked in a logical order to form the rule base.

Streaming media spares the end-user devices from preparing large buffer space for the whole movie and saves the users' time for downloading the movie before they can watch it. Accompanied by the great flexibility, streaming media has its exceptional characteristics and requires more for the delivery system on resources and technologies.

We will explain the inspiration, the rationale, and the design purpose of our work at the end of this chapter and the general contents of consequent chapters.

1.1 Rule-based Expert System

The first section in the Chapter 1 will provide a quick review of the expert system and rule-based system. The characteristics of the rule-based expert system shown in this section will

support the rationale in section 1.3 about using such an AI field technology in media transmission servers.

1.1.1 Artificial Intelligence

The Artificial Intelligence (AI) can be divided roughly into two categories: *Conventional AI* and *Computational Intelligence (CI)* ([1]). Conventional AI most concentrates on development of algorithms and techniques that allow computers to "learn" and react according to its acquired symbolic represented knowledge. The expert system mentioned in this thesis belongs to this category. CI involves iterative development or learning based on empirical data. The knowledge in it is not explicitly stated but is represented by numbers and will be adjusted as the system improves its accuracy. Typical methods included are neural network and fuzzy system. Both *conventional AI* and *CI* have been used extensively in areas like control, planning and scheduling, diagnostic, speech and facial recognition.

During the 1980s, in a project of performing chemical analysis of the Martian soil [2], researchers at Stanford University initially used rules-of-thumb (heuristics) to exclude numerous structures that are unlikely to account for the data. Their work was the first program that focused more on domain information about the problem to be solved, rather than the complex search techniques. It revealed a truth that the domain knowledge of the problem is more powerful than the reasoning methods in achieving intelligent behavior. The revelation eventually created the epoch of Knowledge-Based Systems (KBS), which is also called Expert System.

1.1.2 Expert Systems

An expert system ([3], [4]) formalizes some of the subject-specific knowledge of one or more

human experts into its database and performs reasoning on it for the solution. The structure of an expert system is different from that of a conventional program (figure 1-1). Conventional programs take numeric data as input and execute a set of pre-decided instructions. The solution is given in the form of exact numbers, a pointer, or a logic judgment (True or False). Usually there is no such sequential procedure for an expert system.

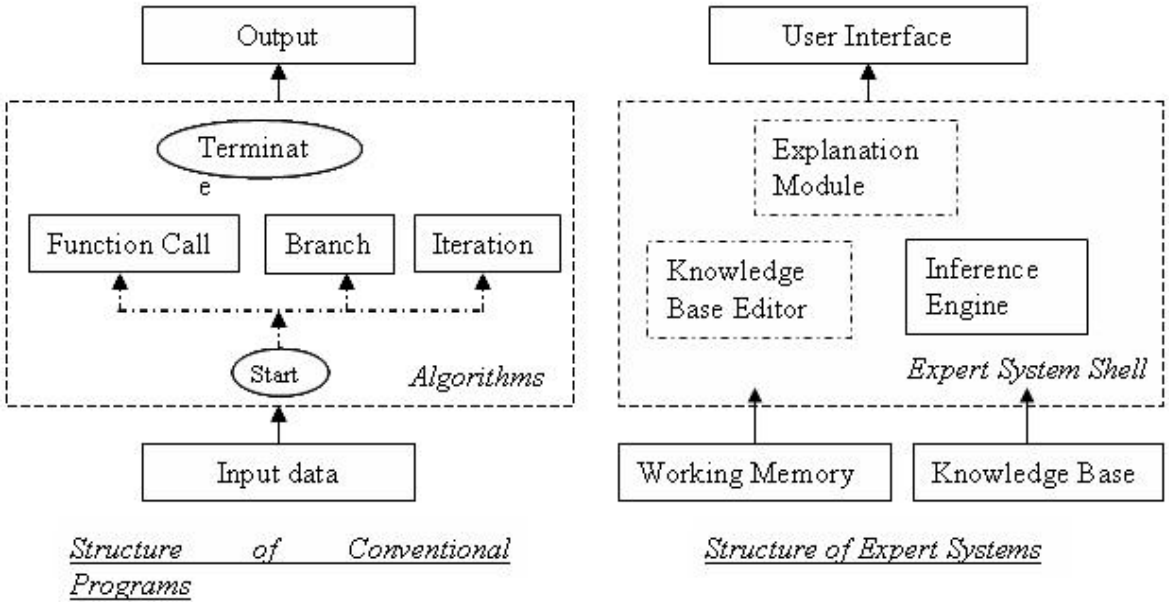


Figure 1-1 Comparison of structures of conventional program and of expert system

In Figure 1-1, the two basic components in an expert system are the knowledge base and the inference engine. The inference engine analyzes the problem and refers to the knowledge base to deduce a solution. Problem related data are saved in the working memory and severs as runtime parameters to record the current state of the system. The knowledge base editor and the explanation module are extra frills that make the whole system easier to use.

The expert system is popular in AI-related research partly due to its flexibility. Since the knowledge base is separated from the reasoning procedure, it is very easy to perform modifications to keep the knowledge base updated. Additionally, the expert system can solve with incomplete data using a large amount of heuristic knowledge. It is a very important

property since the accurate and complete information for a given problem is rarely available in the real world. Yet it also has some major disadvantages like the solutions may not always be correct and its knowledge only limited to a specific domain. Nevertheless, it still gets fast development in many areas. In modern applications, designers borrowed latest database or web techniques for the knowledge representation. S.J.Jang et al ([6]) designed an XML-based expert system that can automatically prescript individual exercises. In their design, the parameters in working memory were obtained from tests on users for their cardio endurance, muscular endurance, etc, and the knowledge base was configured as information frames and saved in an XML file. The inference searches and matches the parameters in working memory with the frames in the knowledge base. The matched pattern was organized and proposed to users.

Unlike the frames used in this example, most expert systems use rule for their knowledge representation, as introduced in the next sub-section.

1.1.3 Rule-based Expert Systems

Rules are used to represent knowledge. It follows the nature of people expressing a piece of knowledge. That is, providing the causes, followed by a conclusion. Therefore, the rules structure is IF-THEN clause pairs:

IF < condition > THEN < assertion/action >

Rules express the associations between input and output. Thus it is suitable to represent procedural knowledge. Using rules, the inference could be performed. When the condition part of a rule is satisfied, an action will be carried out or an assertion will be made. This progression produces new facts. The newly derived facts may cause the conditions of other rules satisfied. Thus one or more rules will be fired consequently. Based on this inference

chain, the reasoning can be performed using two methods: forward chaining and backward chaining. Forward chaining is a data-driven strategy (figure 1-2), in which rules are applied in response to the changes of current working parameters (facts).

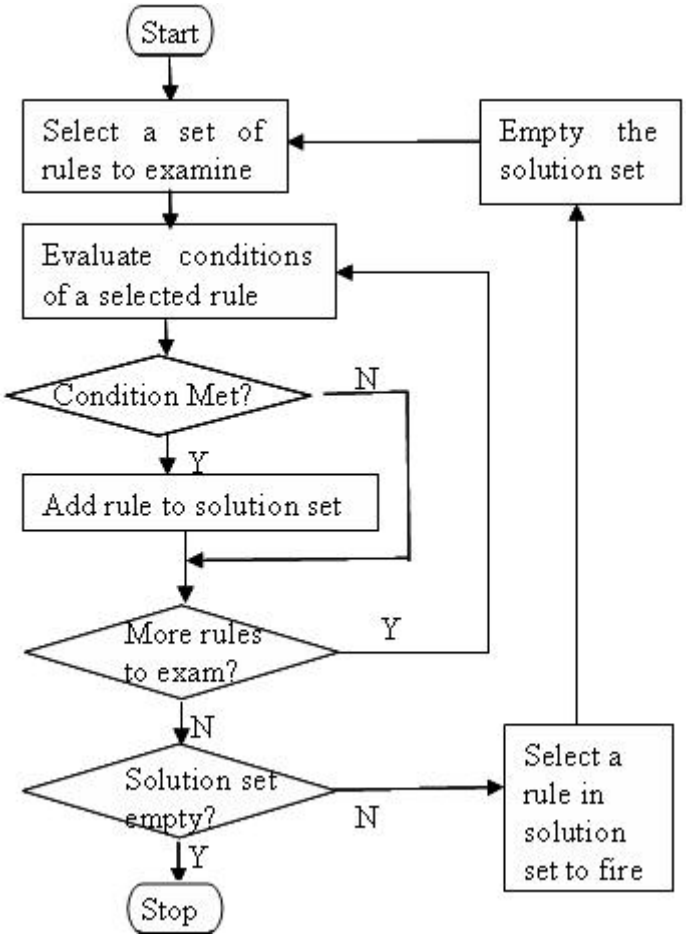


Figure 1-2 Flowchart of forward chaining

Backward chaining, on the reverse order, starts from a goal (G1). If G1 is not satisfied based on current working parameters, the inference engine goes to check is there a rule whose effect part matches it. Upon finding such a rule, its cause part is set to the new goal (G2) and the chaining procedure continues repeating until the goal Gn is verified true by working parameters. Backward chaining is not used in our study, so the detailed flowchart is omitted here. Of course these two types of reasoning can be combined in the real system.

The structure of rules meets the natural format of heuristic knowledge and thus easy to search

and fire. Moreover, the presentation using rules achieves great extendibility on that each piece of knowledge is highly modularized. This enables the developers to start from a small group of rules and extend it step by step to a complete database. However, rule-based systems also have potential problems. We point out the major ones here and make possible amendments.

1) Infinite chaining. If rule A caused the fire of rule B and the rule B, in return, causes the re-fire of rule A, there is a potential infinite loop problem. For this problem, a patent called *Loop Detection in Rule-based Expert Systems* ([7]) was issued by US Patent Bureau on 4th October, 2005. The patent detects the existence of overlapping rules or inconsistently interacting rules that cause the potential problem of infinite loop and prevents those rules to be involved in real execution. In our study, each rule is eligible to fire only once in a round of reasoning.
2) Contradictory among rules. When rule base size becomes larger, it may have two or more rules with the same condition parts but contradicting solution parts. For example:

   Rule 80: IF < network congested > THEN < decrease sending rate >

   Rule 16: IF < client buffer underflow > THEN < increase sending rate >

   In this example, rule 80 suggests to decrease the sending rate to alleviate congestion while rule 16 asks to increase sending rate when client buffer encounters underflow during streaming. However, client buffer underflow often happens together with the congestion because of the jammed packets in intermediate networks. The expert system would encounter a dilemma whether to increase the sending rate or not. Possible solutions will be adding more assertions to the condition part of the

conflicted rules to differentiate their scope, or adding extra meta-rules to handle the conflictions.

3) Inefficient reasoning. With the increasing size of the rule base, the inference time will degrade the performance or even cause the system useless. The efficiency depends on many factors like the characteristics of the problem domain, the length of a single rule, the binary structure of the rule base, and the complexity of the search algorithms. These factors are taken care of throughout the thesis, especially in Chapter 3, 4, and 5.

Now cite an example where the rule-based expert system is used to diagnose problems in circuit simulation. C.W.Lehman and M.J.Willshire implemented an expert system called SOAR (Simulation Output Analysis and Recommendations) to assist in failure-tracking from gate-level circuits to full chip architectures ([8]). In this system, a large number of heuristic rules were used to direct the inference process to converge and to identify the source of the problem. The inference starts from using the failure as a fact, the system performs reasoning by matching the fact with the assertion part of rules. If the assertion describes the symptom of failure successfully, then the condition part of the rule would be added into the solution set as a new indication to be verified. Their experimental results showed that this rule-based expert diagnostic system gave 100% accuracy in their test cases. The diagnostic times (inference time) was only slightly longer when circuit node expanded by two magnitudes.

1.1.4 Summary of the rule-based expert system

At the beginning of this section, we locate the position of our study within the AI area. The comparison between the expert system and conventional programs in sub-section 1.1.2 differentiated their criteria and structures. Then we revealed that most heuristic knowledge is suitable to be expressed using rules.

In sub-section 1.1.3, we introduce the rule structure and the forward chaining process. The major three problems of a rule-based expert system were illustrated with examples and possible solutions. Using the natural advantages of rules, the rule-based expert system has become a dominate branch in the expert system family nowadays. It is powerful for environment dependent problems like planning, task scheduling, decision making, and process monitor and control.

Notice that media streaming is an environment depend application, we were considering whether it would perform better if it is controlled with a streaming expert. To answer this question, we should first study the characteristics for typical media streaming applications.

1.2 Multimedia Streaming

The name of streaming media refers to the delivery method of the medium rather than to the medium itself. It is the multimedia that is continuously played by the end-user while being delivered from the provider. Applications like web TV, distant learning, and P2P systems are all based on media stream delivery technologies. Media streaming plays a more and more important role in commercial society and in our daily life.

1.2.1 Multimedia Streaming System

In the figure 1-3, we demonstrate a typical media streaming system. The components of the server will be introduced in Chapter 2. In the following sub-sections, we discuss the major characteristics of media streaming and network specifications.

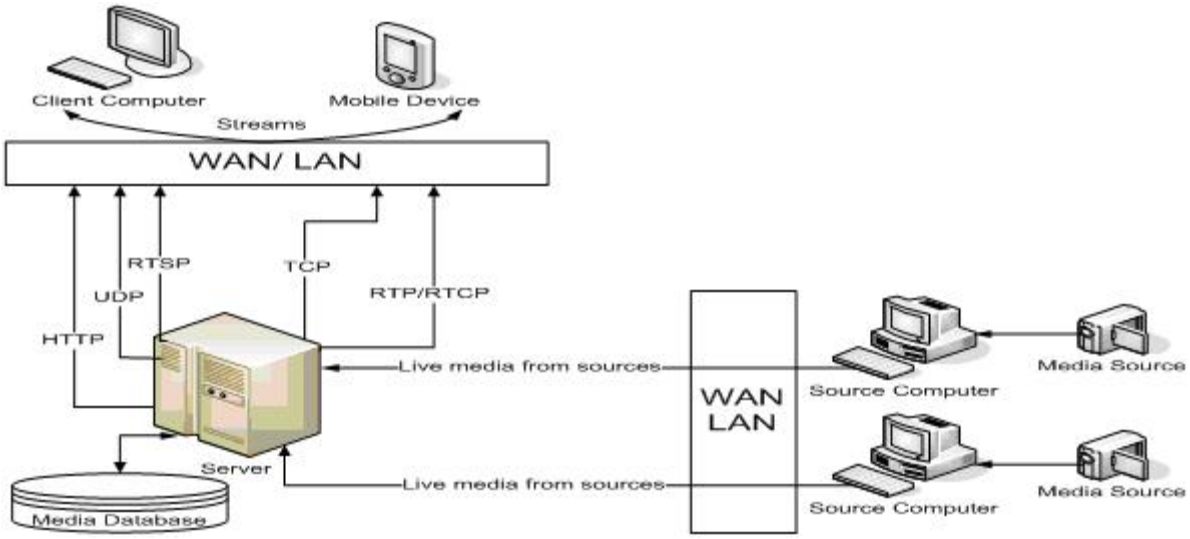


Figure 1-3 Conventional media streaming system

1.2.2 Multimedia Streaming Characteristics

To meet different requirements of qualities of transmission, MPEG standards code the video into several layers (figure 1-4). The performance of a media streaming server will be greatly improved if it can differentiate the frames of each video. In this sub-section, we introduce the transmission steps of a media streaming application and the traffic demands for the streamed media.

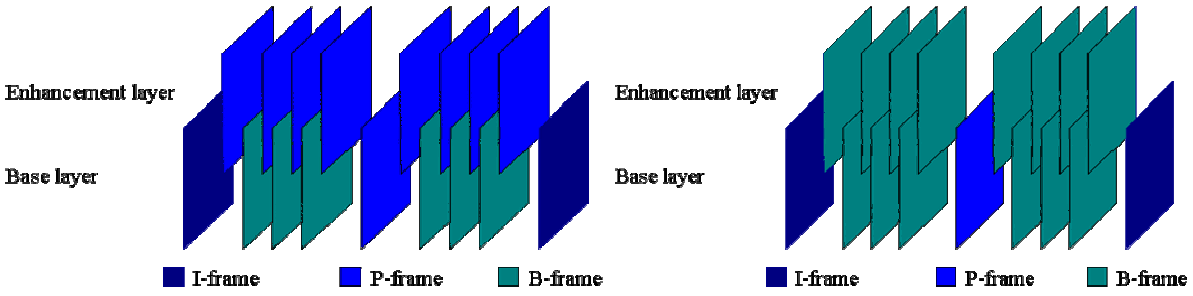


Figure 1-4 Different MPEG2 video layers

A multimedia streaming session would experience three key phases: setup, transmission, and teardown (figure 1-5, left). The online control during transmission is the decisive step for streaming performance. The traffic speed of different frames is shown in figure 1-5 right side.

In the figure, the transmission speed varies between 32kbps and 600kbps. I frames have the largest size, and thus requires the highest transmission rate.

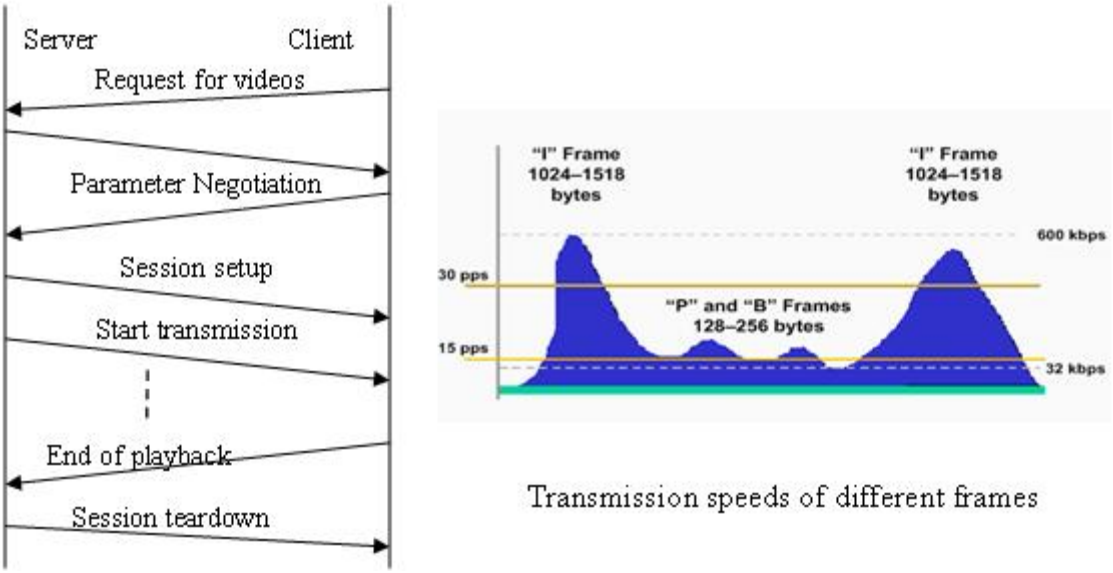


Figure 1-5 Transmission steps and speeds of streaming media

Compared to other applications, multimedia applications generate a large amount of digital information in each second, especially the video part. Streaming applications are very demanding with respect to the overall throughput, loss of packets, frame delay, and jitter problem which is the variation of delay. Delay and jitter problem will consequently affect the synchronization of frames on client side.

However, multimedia streaming is not an intimidating application despite the above mentioned characteristics. It transmits moderately less than FTP applications. It is not so sensitive to delay and jitter as VoIP (Voice over IP) applications. For packet loss, with the development of modern coding and correction techniques, some movies can tolerate up to a 40% packet loss with only slight degradation. Actually, streaming media is not primarily about quality, it is about access. So the video quality of a streamed file is usually much lower than that of an HDTV (High Definition TV). The detailed review of multimedia streaming

technologies will be presented in Chapter 2. In the next sub-section, we will discuss the issues for the proposal in this thesis.

1.3 Rule-based expert media streaming server

After introducing the background information of rule-based expert system, we are now in the stage of investigating the possibility of using it into the media streaming server design. In this section, we will explicitly answer the following questions in order:

- What inspired us to the proposal of this study?
- Is there any related work done with the same scheme?
- Is it feasible to use rule-based expert system in media streaming servers?
- Is using rule-based expert system the best way to solve the problems?
- What are the main goals of such a rule-based expert server system research?

1.3.1 The Inspiration

The original inspiration of this proposed work came from the propensity of optimizing congestion control algorithms to make them better to support media traffic. We found that the algorithms used in current commercial servers, for example Reliable UDP in QTSS, are sufficient to perform high-quality streaming media under light load but somewhat simple for heavy load or unstable environments. Those complex congestion control algorithms only take one or two QoS parameters instantaneously for its decisions. The usually taken parameters are network loss rate, packet round trip time (delay), or previous sending rate. These parameters can represent the variation of transmission environments but they can not provide the whole picture of the situation individually. Moreover, the same change may be caused by different reasons. If the algorithms ignore the related information and the historical trends but

merely take the result as indications for adjustments, it may easily be overactive or not responsive. Even if a congestion control method is designed perfectly, it usually needs the cooperation of other mechanisms. For example, a congestion control algorithm needs to sample the network condition in 5 milliseconds for best performance but the task scheduler always delay the sampling procedure to a period of 500 milliseconds. The information given by the sampler is always lacking in consistency and inevitably harms the control process. Therefore, only those mechanisms that could cooperate with each other well should be selected and work together for the optimization of performance.

From the above investigation, we realize that the solution for congestion problems does not only rely on the control algorithm design itself. It is an overall contribution from every element supporting the transmission. The streaming server is like an active entity that makes decisions, carries out actions, adjusts its behaviors, and learns from experiences. All components inside the server are related. The streaming procedure needs much cleverness to handle those components for the problems continuous appears without explicitly predictive reasons. The solution turns out to be an intelligent streaming server.

1.3.2 Related works

As early as 1988, AI researchers have paid their attention to use the expert system on network control. E.J.Zakrzewski and R.Quillin ([9]) employed the expert system to perform network wide control decisions with only local or sector system status information provided. The system could support network monitoring, fault isolation and system adaptation in degraded modes. Their system could only focus on service assurance in a communication network. It is more concerned with the overall topology of the network rather than the applications.

Later, expert systems were used in network capacity planning. The work in [10] is similar to the study presented in this thesis in that both of them are designed for server resource allocations. Yet this paper focused on ISDN network and all possible applications on the server. In our study, we do not have any specified network architecture but the application must be media streaming. In this paper, applications were classified based on their burstiness and time constraints. Rules were employed for the control. The bandwidth capacity was partitioned into N channels for N types of applications divided by their QoS requirements. The CPU was shared among three categories of tasks: signaling and control, voice and delay-sensitive traffic, and delay-tolerant bursty traffic. Their priorities were assigned in an incremental order. The work realized the upper level server capacity control for multi-service system, but the control effort looked too coarse.

Nowadays, attention is turned from network planning to more specific areas like traffic prediction, task distribution, and active queue management. In [11], M.M.S.Rao et al applied a rule-based expert system for short-term traffic prediction in the power supply system. They classified the factors that influent the system load into four categories and analyzed their behavior through experiments. The analytical results were translated to rules and used for future load prediction.

Other works used the expert system for task distribution after predicting the traffic. Calleja and Troost ([12]) implemented a rule-based expert system model into their naval command and control system to handle the workload segments and to deal with uncertainty and fuzziness. In the system, traffic prediction was handled by another module. The rule-based expert task distribution module took the predicted results for its decision on workload balance among operators. The attractive feature of this module was that it may modify or

reassign tasks for excessive workload situations to recover an operator. This module simulated the behavior of a team leader doing task assignment.

Active queue management (AQM) is an important research area developed recently to support network scheduling and congestion control mechanisms. J.Wu and K.Djemame ([13]) designed a new AQM algorithm that used an expert system for buffer management. In their system, related issues like cost at the switch node, congestion avoidance, traffic policing and delay price were considered for the control decisions. The results obtained from NS-2 simulation showed that this expert system based AQM algorithm achieved significant performance elevation on queue occupancy and throughput compared to other AQMs recommended by IETF (The Internet Engineering Task Force).

The domains involved in previous works were too broad to be specialized within a single expert system. They did not specify a target application. Thus the heuristic rules were hardly proved to be effective considering the variety of requirements from different applications. As a result, the outcomes of early attempts to control the network applications or task scheduling lack domain related significance and seem ambiguous in the problems they attempt to solve. Due to this drawback, the inference procedures were not convincing since the rules in those systems were designed for multiple applications that may have different or even contradicting requirements.

Theoretically, nothing prevented rule-based expert systems from being used in media streaming control areas and there exists no related works done in the literature. In the next sub-section, we will investigate this gap and analyze the feasibility of applying rule-based expert system for media streaming applications.

1.3.3 Rationale of the Proposal

Historically, most expert systems are designed for business planning, manufacturing process control, and disease diagnosis. Although there were works that applied them into the network control or task planning, it needs further investigation on whether the rule-based expert system is really suitable for media streaming applications or not.

Firstly, let us recall the characteristics of a rule-based expert system and the kind of problems that is suitable to it. An expert system simulates the skills of a human expert to solve a problem based on current data, past experience and appropriate reasoning. Unlike conventional computer systems that usually repeat the algorithmic routine work, an expert systems need to find the solution themselves first before taking any action. The decisions it made based on knowledge base, in which large amount of heuristic and theoretical knowledge is coded. Rather than giving out a strictly optimal solution, an expert system first offers a sub-optimal solution and takes such a solution into consideration for further reasoning, getting closer and closer to the final decision. The final solutions are not fixed. They are obtained by reasoning the current situation through some inner principles represented by rules. They may not be optimal, but must be feasible and correct in most cases. In summarize, the performance of an expert system depends largely on the correctness of the knowledge base, the precise of working parameters, the efficiency of the inference procedure, and also on the system capacity of executing the decisions. Thus expert systems are mostly suitable for high level controls that have different patterns of solutions and the decisions are made from the overall picture of the problem.

Examining the multimedia streaming servers, there are two types of work need to be handled for a successful transmission. One is the routine work, like receiving and analyzing client

requests, sending media data based on standard protocols, and maintaining session states. The second type is control work like adjusting session parameters, selecting the schedule strategies, or response to congestions. Conventionally, these control works were fixed coded like the routine procedures. The same procedure of control is repeated whenever the control function is called. Although a single control algorithm has to be realized by sequential programs, the decisions of when and how to use it is difficult to formalize in the conventional way. Those decisions depend on many issues during a transmission with some kind of uncertainty. Therefore, an alternative way to perform these control works is extracting the control-related information and control principles from the main body of the server program and organizing them in a separate supporting database. The server intellectually selects and regroups cooperative methods according to characteristics of problems and adjusts their parameters for the best performance. This architecture matches the expert system very well, especially for rule-based expert systems that natural in presenting heuristic principles.

There may be other mechanisms that can fulfill the requirements. For example conditional branches in conventional programs can perform similarly. However, they are only similar on format but different intrinsically. The conditional branches list all possibilities of a situation and the control flow is fixed. No matter which branch is selected, the execution of it is very unlikely to influent the later entrance of other branches. On the contrary, the rules in the knowledge based are reasonably related to each other. The control flow is set up during runtime. That is, a rule will not be fired if it is not selected by the current control flow even its condition is satisfied. Additionally, the fire of a rule usually will cause the activation of other rules. These consequent results make the progress of inference possible. In summary, the conditional branches are independent choices listed in the program without inference

procedures for the decisions while a rule-based expert system organizes the problem related knowledge logically and performs reasoning based on the knowledge. To decide which form is better, we only need to determine whether the server system needs intelligent reasoning or merely more choices for selection. The answer is obviously the former one.

For all these reasons, we believe that the rule-based expert system is the most suitable choice to solve the problems encountered in current media streaming servers. We hope the system designed from the proposal will achieve the purpose listed in the next sub-section.

1.3.4 Purpose of Research

The aim of our research is to investigate whether adding the expert control could make the traditional server smarter on the whole media transmission. How much will be the system performance improve? This could be reflected by the incoming session distribution strategy and optimal route selection ability of the server, the cooperation among the servers during congestion, and per session evaluation parameters like throughput, delay, jitter, and bursty rate. We also want to investigate what size of knowledge base is required to achieve such performance enhancement, and will the overhead brought by the expert control significantly decrease the number of supportable clients. The new platform aims at fulfilling the following requirements:

A. Able to perform effective admission control and traffic distribution.

B. Provide user level playback scheduling.

C. Parameters are adjusted dynamically during runtime.

D. Carry out smooth traffic shaping and buffer management.

E. Perform knowledge-based congestion control under various environments.

F. Implement failure detection and recovery mechanisms.

G. Control the overhead within a reasonable range

To accomplish the above mentioned purpose, we change the conventional server structure and its decision making procedure; we introduce a forward-chaining planning expert system to perform streaming control; we built a completed rule base to handle the streaming problems; we also design a client-oriented rate control algorithm to strengthen the knowledge base. These accomplishments will be illustrated in detail in the following chapters.

1.4 Organization of the thesis

The thesis would be organized as follows. Chapter 2 reviews the common technologies in media streaming servers. We reviewed the technologies from three perspectives: server, network, and client. For network and client sides, the streaming related protocols and parameters are introduced. For server side, detailed classified methods will be explained with literature review. At the server side, the most relevant technologies to this thesis are scheduling strategy, congestion control, and buffer management. Chapter 3 gives the detailed design of the rule-based expert server system. In this chapter, we are going to introduce the representation of knowledge base, search algorithms, and the expert server layers. We will also explain the expert server modules and dynamic inference procedures. The final consideration will be the communication model between modules. After presenting the design, the server system performance and capacity are analyzed in Chapter 4. In this part, the server computational complexity is quantified. The average response time of requests and tasks will be analyzed based on the computational complexity. Other real time characteristics of the system are also considered, followed by an estimation of the system capacity. A simple throughput analysis under multicast situation is also mentioned. All implementation strategies and the corresponding experimental results would be provided in Chapter 5. We

will explain in depth the state of art when the system is realized. The experiments are conducted on a test bed and public network using QuickTime Streaming Server with and without the expert system. They are carefully designed to demonstrate the research goals listed in sub-section 1.3.4. Chapter 6 is a comparatively independent chapter, in which we will present our efforts on designing a client oriented rate control method used in the expert server knowledge base: DLQ Rate Control. Finally, Chapter 7 concludes the whole thesis on the achievements and limitations of the rule-based expert server we designed, and also indicates the potential future developments.

Chapter 2 Review of Media Streaming Technologies

In section 1.2, we have introduced the general structure of media streaming system and the streaming characteristics. However, the information is not sufficient to design or even understand a streaming server. In this chapter, we focus on more specific server-design related streaming evaluations and support technologies. These technologies are used in current commercial servers and will be scheduled and handled by the expert control.

QoS is the major evaluation criteria. In this chapter, we are going to introduce the QoS parameters that will be used in our experiments, like throughput and jitter. For supporting technologies, we will introduce the most related ones like traffic prediction, admission control, server buffer management, congestion control, and traffic shaping. At the end of introduction of each technology, we will introduce its relationship with others and how it is going to be handled in our expert server. Only through familiarizing these criteria and technologies, can we understand the underlying mechanism and major improvements of the expert control.

The key issue for a streaming server would be resource allocation. Streaming application needs fast CPU response, large network bandwidth, low delay, and low loss rate. All these qualities obtained from proper allocation of the server and network resources. This chapter will give the review on major resource distribution technologies, together with some commonly used commercial servers.

2.1 Technologies for Media Transmission

The technologies will be reviewed with reference to the components in figure 2-1.

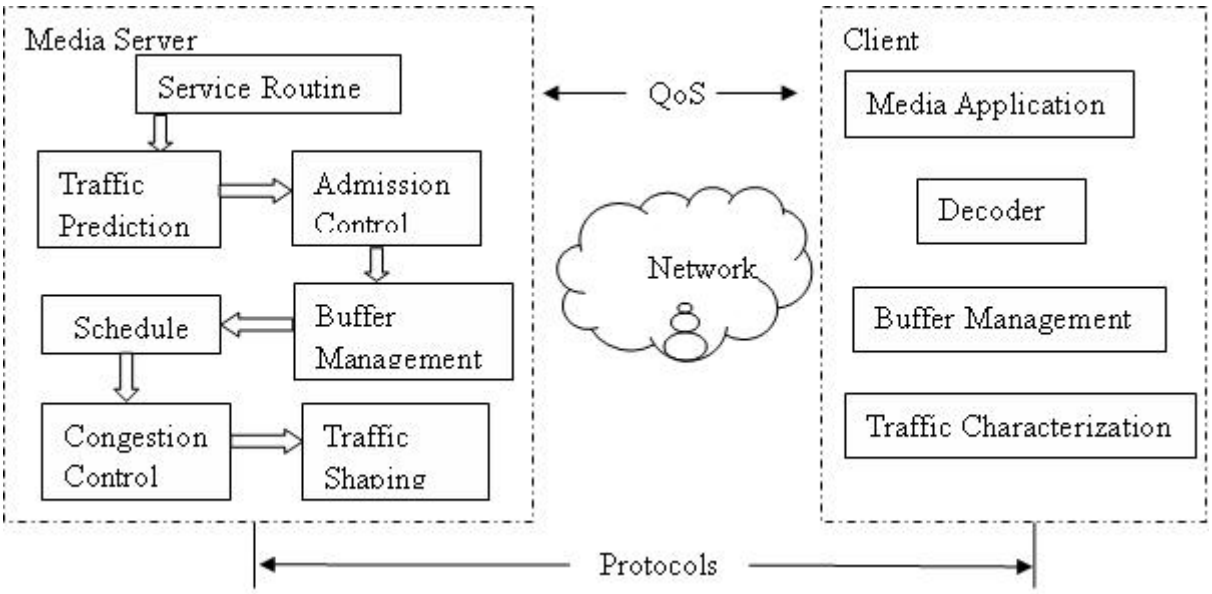


Figure 2-1 Components of multimedia streaming system

2.1.1 Service Quality and Protocols

❖ QoS, the criteria of quality measurement

Quality of Service (QoS) is a magic word that appears frequently in literature but without an explicit definition. Usually, the QoS for media streaming includes:

a) *Guaranteed use of bandwidth.*
b) *Limits on cell loss / packet loss.*
c) *Limits on latency (one-way or round-trip).*
d) *Limits on jitter (delay variation).*

Other parameters in multimedia streaming like transmission reliability, synchronization, and throughput can also be used for QoS measurement. Since QoS represents the combination of quality factors, it needs the support of overall systems. Thus researches related to QoS management either put their effort on global control and structural tuning ([14][15]) or design the whole transmission system with QoS awareness ([16][17]). Our expert control, also targets at improving the general QoS, and does a similar global

contro.. It has QoS maintenance modules and rules, implementing the standard or heuristic QoS adjustment ideas. The QoS criteria we concern are traffic distribution, throughput, loss, jitter, multi-channel scheduling, and congestion response. These QoS parameters will be tested in Chapter 5.

- Protocols used in streaming

  Most streaming applications nowadays use HTTP for content browsing, RTSP or SIP for session initialization, RTP and RTCP for real time control, and UDP for data forward. For some applications with restrictive firewalls, HTTP has to be used to carry the media data. It is not efficient and only suitable to webpage plug-in streaming. The QuickTime streaming server has all these protocols, which will be introduced later for our experiments in Chapter 5.

  Among these protocols, the only one that is control related and QoS related is RTCP protocol. RTCP performs three major functions: feedback on the quality of the data distribution, persistent transport-level identifier for an RTP source called the canonical name or CNAME, and rate control of RTCP packets. It is designed for general purpose and targets at a single flow. Therefore when specific requirements are needed for a stream or the overall performance is considered by the server, RTCP control seems lack of global view and cooperative handling ability. This is why we still need to implement the expert control although we have had this control protocol. The effects of RTCP control will be shown in our experiments in Chapter 5, together with the results of expert control. Many innovative protocols for streaming transmission were proposed these years (e.g. [18][19]) but we prefer to use standard ones.

2.2.2 Server Technologies

- Traffic Analysis and Prediction

  For multimedia streaming, traffic analysis completes three kinds of work. One is to differentiate media streaming from other non real-time applications [20]. The other is providing movie parameters like peak bit rate, frames dependent ratio, average frame size, and bandwidth requirement [21]. The third is bandwidth prediction [22]. The analyzed statistics help the system decide server buffer size and sending speed.

  Primarily, there are two methods to conduct traffic prediction: measurement or traffic model ([23][24]). Measurements have higher accuracy but higher computational complex. Using the traffic model does not waste CPU time for measurement but the accuracy depends largely on the quality of model. Due to the variation of movie characteristics, it is very difficult to have a uniform model; the distortion brought by the model will more or less harm the performance of the whole transmission. In the design of our DLQ scheduler in Chapter 6, we use a measurement method for traffic prediction.

- Admission Control

  Admission control ([25]) is usually implemented between network edges and core to control the traffic entering the network. It is also used in media server to control the user population ([26]). Precise admission control needs the support of traffic analysis and prediction, together with proper acceptance criteria, and a proficient control algorithm. This mechanism must be realized for a server to control the load level. Our expert control is built up on top of the traffic prediction and admission control. It analyzes the traffic of servers in the server cluster and refers to heuristic rules to decide the admitting of a new session and the distribution of this session among all servers.

- Buffer Management

  Buffer management techniques offer fundamental support for data manipulation. Besides conventional buffer management methods (RED [31] FRED [32] XRED [33]) designed for general traffic, several multimedia transmission oriented methods are proposed these years. Some of them are end-to-end management method ([34]) that trade off random loss for controlled loss of visually less important data. Some are user-oriented fair buffer management ([35]) that focus on user expected video quality. Another possible method are look-head buffer management ([36]) that set up a virtual buffer to prevent loss.

  In our expert server, these buffer management schemes are selected and tuned by the expert control, working in cooperation with other scheduling and congestion control methods to enhance the overall server performance.

- Task/Packet Scheduling

  Task scheduling performs low level scheduling that manages hardware resources like CPU time and I/O bandwidth [37]. Our work does not handle OS level scheduling directly but register the expert control process at top priority in the Linux kernel.

  As for packet scheduling, the simplest method would be round robin (RR) that serves each flow in turn. General purpose scheduling methods used in current networks are WFQ (proposed by John Nagle in 1987), WF2Q ([38]), BSFQ ([39]), and DiffServ ([40]). Besides them, many media transmission oriented scheduling schemes are proposed:

  - Window based scheduling. Dynamic window-constrained scheduling method ([41]) guarantees no more than x packet deadlines are missed for every y requests by adjusting the window size according to loss constraint. It bounds the packets delay at an acceptable level and promises the minimum bandwidth utilization. It is a

developed method based on WFQ, EDF, and conventional IP window scheduling technique. However, window based scheduling usually performs slower than rate based scheduling. For fast speed large volume media data transferring, window based scheduling would be the second consideration.

- Multi-layer transmission. This method divides the media transmission into multiple layers and applies different strategies at each layer ([42]). Actually this method has been extensively used in streaming servers, including the server used in this thesis.
- Multi-path scheduling. Some muti-path scheduling methods ([43]) transmit key frames through reliable networks and less important frames through best effort service. The expert control will also consider multi-path scheduling but our target is to choose the most suitable path for the whole stream so that the synchronization problem at the receiver side is avoided.
- Wireless media transmission. The wireless channel has distinctive characteristics like low bandwidth, high loss rate, fading problem and an unstable environment. Therefore, assured transmission is the key consideration for scheduling in wireless channels ([44]). We use TCP like window control in our research for the wireless streaming.
- Heuristic scheduling. Unlike previous methods with mathematical formula or exact algorithms, heuristic scheduling works on experienced knowledge or statistical information. It is quite powerful in solving problems with unpredictable or vague input. The method proposed in ([45]) checked the fail ratio of previous packets and raised the flow priority level if there was a loss. It could prevent continuous loss for a flow. Our expert server will not use heuristic schedule methods but will embed some

well-defined heuristic rules for meta-level control of these schedule methods.

- Receiver-driven BW sharing. Contrary to conventional BW sharing scheduling methods that focus on the capacity of network, this kind of scheduling method allocates bandwidth among TCP flows according to user references ([46]). It offers a different aspect to providing QoS and boosts the completeness of scheduling strategy design. We were enlightened by this idea for our client-oriented rate control method.
- Coordinate CPU and BW scheduling. The work [47] combined the two types of scheduling methods. This could be a future direction of the expert control.

In Chapter 5, our experiments will be conducted on QTSS (Quick Time Streaming Server), in which Reliable-UDP is adopted. Reliable-UDP is very similar to the window based TCP control. It requires feedback from the clients, and adjusts the sending rate based on feedback information. The expert control takes advantage of it and adds rules to control the window size and the sending rate adjustment policy. The expert control also considers the multilayer scheduling and the multi-path scheduling. It sends out only the necessary layer of stream, and it could select the least congested route during congestion. These heuristic rules are all designed according to the scheduling algorithms introduced in above paragraphs.

- Congestion Control (Rate Control)

Congestion control can be a separate module or implemented into protocols. Since UDP does not perform any congestion control itself, people developed some revised versions of UDP with congestion control ability. For example, the Apple's QTSS uses reliable-UDP that accepts feedback from client to support its flow control module. Other non-protocol congestion control methods collect loss and delay information at the end systems

and determine a TCP-friendly transmission rate during the streaming ([27][28]).

Another similar issue, rate control, often appears together with the congestion control. Since both are used to adjust the sending rate, it is easy to mix them up. Actually the aim of rate control is to control the speed of a flow for certain QoS requirements. It is necessary even with light traffic. Congestion control, with the aim of avoiding jam, focuses on the overall load level of the network and regulates the sending speed without concerning much on the application characteristics. It is turned on only at the time of congestion. Two types of congestion control are classified here.

- Window based congestion control. In this group, window size is used to determine the number of packets eligible to be sent. The control effort is performed by adjusting the window according to the receivers' acknowledgements. Window based methods have the advantage of accuracy and effectiveness, especially for wireless channels. But the control steps are discretely executed and sometimes cause window size oscillations. Usually it acts slower than rate based methods.
- Rate-based congestion control. This kind of method ([29][30]) changes the sending rate by adjusting the interval of consequent packets. It is often used with faster UDP flows like media streaming.

The expert control will take care of both types of congestion control algorithms. It has rules designed according to the advantages and disadvantages of them, decide when to use which one and adjust the parameters of these algorithms. The detailed discussion about congestion control methods will be carried out in Chapter 5 in the case study of our rule-based expert server system.

- Traffic Shaping

Traffic shaping is a preliminary method of QoS control to prevent the performance of streaming degraded by jitter or lost. The sender or the routers modulate outgoing packets so that they appear to be more periodic at an appropriate speed. Refer to the figure 2-2, the shaping procedure delays excess traffic using a buffer, or queuing mechanism (a priority queue (PQ), a custom queue (CQ), or a FIFO queue), to hold packets and shape the flow when the data rate of the source is higher than expected. Our expert server does not implement traffic shaping mechanisms separately, coupling the work with rate and congestion control.

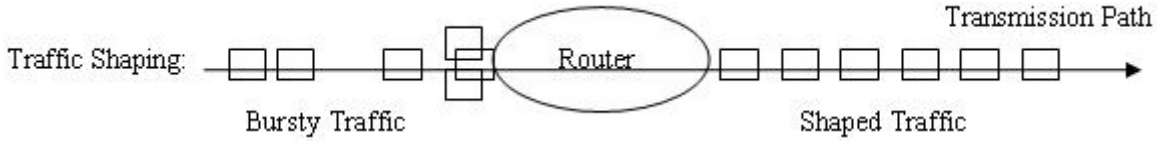


Figure 2-2 Traffic shaping

2.2 Current Multimedia Streaming Servers

Many commercial multimedia streaming servers are currently available in the market. The popular ones are Microsoft Windows Media server, RealNetworks Realserver and Apple QuickTime Streaming Server (QTSS).

Microsoft Windows Media Server ([48]) works in conjunction with Windows Media Encoder and Windows Media Player to deliver audio and video content to clients over the Internet or an intranet. The clients can be other computers or devices that play back the content using a player, or they might be other computers running Windows Media servers (proxying, caching, or redistributing content). Clients can also be customer applications that have been developed by using the Windows Media Software Development Kit (SDK). It provides some new features like fast Streaming, real time monitoring, and IPTV support.

RealServer ([49]) is a member of the RealSystem G2 family of software tools. Similar to Microsoft Windows Media Server, RealSystem G2 makes up of three components:

1) *Production tools*: like RealProducer Pro or RealProducer Plus that creates media.
2) *RealServer*: Media streaming server.
3) *Client software*: for example, RealPlayer.

Similar to the Microsoft Media Server, it streams both pre-recorded and live media over the networks to real time watching. In the latest version RealPlayer7.0, the view source feature allows users to view the source code for SMIL presentations or media clips. The user can also browse the on-demand content available to the RealServer.

Although powerful and popular, the Windows Media Player and the RealPlayer are commercial streaming servers without source code opened to the public. Thus developers can not work on them for their own research. In the study, we make use of the open source QTSS (also called Darwin Streaming Server) and implemented our expert server. The detailed introduction of QTSS will be provided after the expert system implementation is described.

2.3 Summary of the multimedia streaming server technologies

In this section, we introduce the streaming technologies that developed rapidly in recent years. In section 2.1 the server components were divided into three groups: server side components, network components, and client side component. Client side and network side components are not controllable for a server design, so we only introduce the parameters associated with them. The server side components include traffic analysis, admission control, congestion control, buffer management, task/packet scheduling, and traffic shaping. Each of them carries out a dedicated function. Their cooperation determines the overall streaming performance, which is mainly gauged by four QoS parameters: throughput, delay, jitter, and

loss rate.

With the increase in network speed and computer capacity, the commercial streaming servers become more and more powerful. However, many problems still exist. For example, if delivered at a low cost, the streamed media are often interruptive. Even for premium paid streaming contents, the video quality is not satisfactory during peak hours. When talking about the components of a streaming system, we see many innovative methods proposed to solve the problems of media transmission. Yet they only focused on a special component, for example traffic analysis or packet scheduling. The methods designed are attractive but sometimes they need fixed settings as assumptions for the desired performances. When considering these potential problems, we realize it is necessary to find a flexible way to meet the requirements of current media streaming applications and to make it extensible with the fast changing future of streaming applications. The detailed design of such a system will be given in the next chapter.

Chapter 3 Rule-Based Expert Server System Design

In this chapter, we present the complete design of the rule-based expert server system. The design related background information is introduced in the first section. The information includes the representation of the knowledge base, search algorithms used in the server, the expert server layers, and the general server cluster structure. Then we will explain the server components comprehensively. With the support of these components, the inference procedure is illustrated with an example. In the last part of this chapter, we give the communication model between modules of the server.

3.1 Introduction

The presented rule-based expert server, which targets streaming applications, has its unique way of representing the knowledge base. Here we will introduce the format of rules, followed by some search algorithms as background information. The expert server layers are also presented in the last part of this section.

3.1.1 XML

The Extensible Markup Language (XML) is a general-purpose markup language. An XML document contains markup and character data. The markup contains the meaning, such as "variable name", and is held in tags and other XML elements. The character data is the content. An example would be:

<variable name>Number of Client </variable name>

In above example, notation '<' and '>' delimits the tags. The character data, which is the

variable name, is put between the starting and end markups. Users can define their own set of tags suitable for the application. A single tag pair is defined as a root element. All other elements, also in pairs, are nested within this pair. Sub-elements are nested within their parent elements, forming a hierarchical data structure. An XML document looks like ([50]):

```
<variables>
        <variable>
                <name> Number of Clients </name>
                <value> 1000 </value>
                <type> integer </type>
        </variable>
</variables>
```

The first line is the root tag of the *variables* element. In the second tier, *variable* is a child element of *variables*. It represents a specific variable. Below it are the child elements of *variable*: name, value, and type. All these markups are defined by users as attributes in the DTD (Document Type Definition) file. A typical DTD in XML1.0 defines like:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root-element [ doctype-declaration... ]>
<!ELEMENT element-name content-model>
   <!ATTLIST element-name attr-name attr-type attr-default ...>
   <!ATTLIST element-name attr-name attr-type attr-default ...>
   … …
```

During the rule base realization in Chapter 5, we will provide the DTD, the binary structure, the parser, and the linking of the expert server rule base.

3.1.2 Search Algorithms

The decision making procedure is a process of searching the rule base. There are numerous search algorithms for different kinds of requests and criteria. Now we give a quick review on those adopted by expert systems.

The Breadth First Search (BFS) and the Depth First Search (DFS) are two basic search algorithms. Yet these two search methods perform complete search and become less efficient with the increase of search space. To solve the problem, the heuristic search algorithms become popular in AI applications. The heuristic search algorithms use heuristics (rules or functions) to narrow down the search space or branches, directing the search procedure to final solutions effectively and fast. Solutions vary with different heuristics, and they may not be optimal. Commonly used methods are hill climbing search, beam search ([51], [52]), breadth-first heuristic search ([53]) and A* search. The A* search takes global sum of the cost to arrive at the current point and the cost to reach the final goal to truncate the searching space. It does not only focus on the goodness of the next step. The intelligence of A* search actually relies on the evaluation functions, not on the searching strategy itself. In the consequent paragraphs, we discuss the former two methods since they are used in our work.

Hill climbing (best first search) is a mix of DFS and heuristics. Instead of randomly select a child under the current parent node to further the depth first search, hill climbing method uses the heuristic to select the best one from all candidates. The unselected paths usually are discarded in order to save time. This search procedure has the advantages of being informed on each step and modeling human reasoning. However, the solution is not guaranteed to be found. That is, if the search path is not directed properly, the search will reach the local optimal point and no way to return to the global optimal one. This is the main search algorithm used in our study.

Beam search is another popular heuristic search algorithm. It is like a mix of BFS and heuristics. The beam number is used to narrow the solution set size (width) of child nodes to save time in the next round of search. The survived lucky child nodes are selected by

heuristics. Beam search is also adopted in our study but it only used for fine level adjustment after the first round of inference.

3.1.3 Expert Media Streaming Server Layers

Before introducing the design comprehensively, it is necessary to clarify the position of the expert system in a server. In Chapter 2, we have introduced the conventional multimedia server architecture. Expert system can be treated as an embedded part in these servers like a control middleware. Figure 3-1 shows the corresponding layers between server and client and communication protocols at each layer.

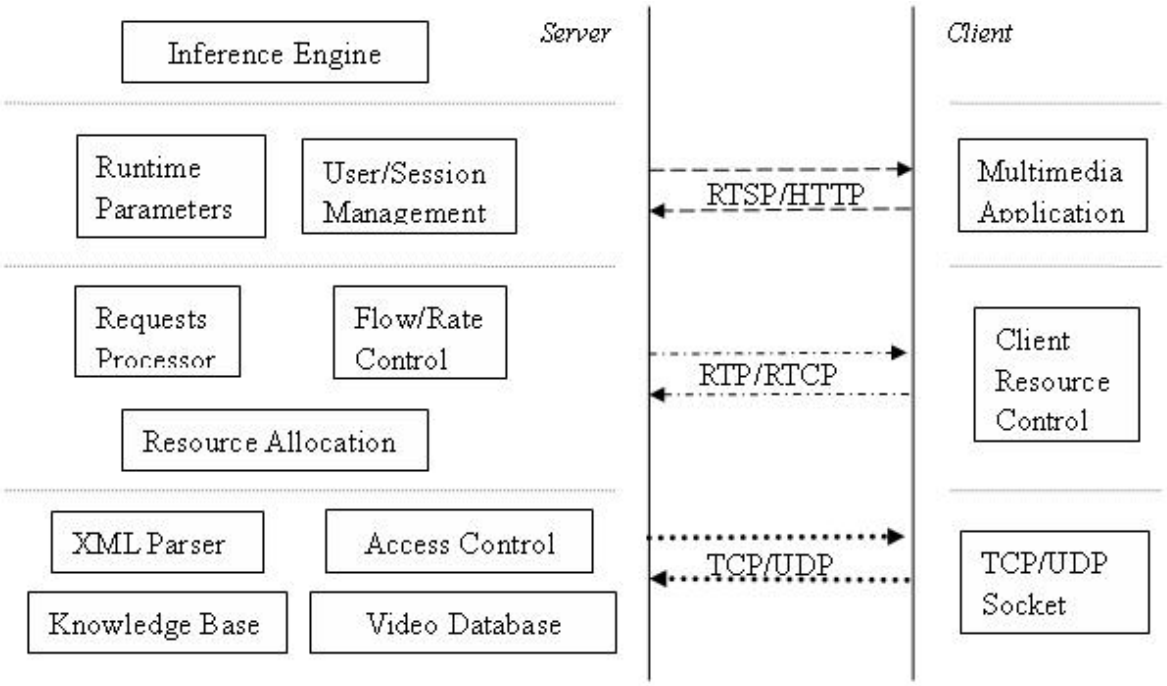


Figure 3-1 Expert media streaming system layers

Comparing to Figure 2-1, the inference engine in highest level and the XML parser and Knowledge base in lowest level are new figures. Although only three parts are added, the whole control procedure is changed. It is these additional features that enable the expert system to allocate resources that are more reasonable and flexible. In the following sections,

we will explain the detailed design of the expert server, especially these added features.

3.1.4 Topology of Distributed Server Network

Servers located at a facility are grouped into a server cluster (Figure 3-2). Each server has its own decision making mechanism. That is, each server is equivalent in functionality. Working parameters are periodically broadcasted among servers in the same cluster and also among clusters. A client can send the request to any server station. The server will make a global decision based on latest working parameters and forward the request to the most suitable station for processing.

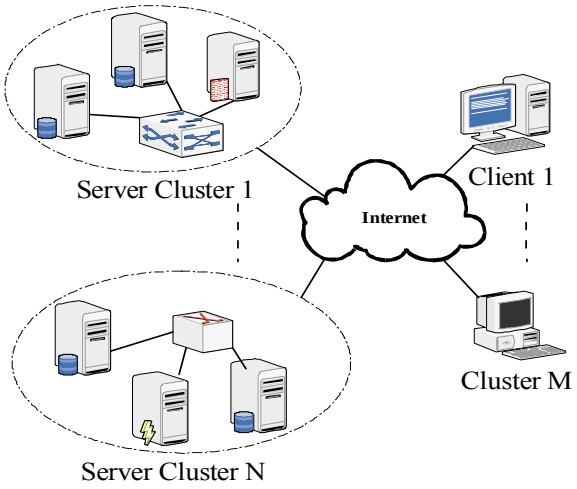


Figure 3-2 Expert server system topology

3.2 Server Design

In this section, the design options and tradeoffs are listed and studied. Then the detailed server structure is provided and explained. After that, we follow a memory allocation request to see the decision making procedure. Finally, the communication mechanism among modules is shown with a diagram.

3.2.1 Design Options and Tradeoffs

Recall the QoS parameters introduced in Chapter 2, the QoS we want to achieve are:

a) *Guaranteed use of bandwidth.*
b) *Limits on cell loss / packet loss.*
c) *Limits on latency (one-way or round-trip).*
d) *Limits on jitter (delay variation).*

To realize the above performances, we consider many options during the design. We list some important ones here, followed with the analysis and tradeoffs for our decisions.

Server oriented or client oriented

The first and most important thing to be decided is which part is the focus of optimization. In other words, what is the target system along the streaming path that going to be optimized, the server system, the route system or the client system? If server system is targeted, all designs should concentrate on server parameters, and the ultimate goal is to give the service providers (who own the server) the most flexibility, the best performance, the most efficient management cost. If the router system is the concentration, the design will focus on the selection of shortest path, the choice of most reliable route, or the minimization of network cost. If client system is the target, providing a user-friendly interface and power-saving features in the client device are appropriate design topics. In our research, we concentrate on the server system. That is, we use the current available network and client technologies to make the streaming server achieve higher reliability and more flexibility for service providers. The performance of an expert server would be better with the cooperation of intermediate routers or client terminals, but will not rely on their cooperation.

Centralized or distributed

At the beginning of design, there are two choices to realize the system, centralized control or

distributed control. Both of them can support the expert system. A centralized system is easy to establish and maintain, and the messages exchanged among stations are scalable. However, the centralized control is not reliable. If the central server breaks down, the whole system is useless. On the contrary, distributed control is difficult to establish and maintain, and the messages exchanged among stations will be overwhelming as number of stations increase. It has the advantage of reliability, that is, a single station failure will not impact the performance of other stations.

To balance the cost and performance of these two types of controls, we finally deployed a hybrid system. The servers on an area (usually a city size) are organized using centralized control system, and a most powerful server is selected as the connector to the outside. The connector servers in different areas are organized with distributed control system, and they share information periodically. If the connector server in a server cluster is down, another backup server will take over the work.

Global control or local control

In the literature regarding streaming transmission, most researches focus on a single scenario or a single algorithm. We also faced the problem of whether to design a local control mechanism or a global control system. After thorough investigation, we found the global control to be a big gap in current server research area. Although a lot of algorithms have been designed for various situations and various traffic, there is no mechanism to integrate these algorithms into a server and make them cooperate with each other.

Furthermore, with the development of modern protocols, the streaming under RTP and RTCP control is already good enough for standard movies. Those algorithms designed for marginal cases are too complex to be used than simple heuristic rules. Thus, we selected some useful

streaming control algorithms into our method base and translate the complex algorithms to simple heuristic rules to form the expert server. It combines the power of the latest streaming technologies and reduces their complexity. The implementation of a global control system will definitely consumes more CPU time and will influence the transmission of movies. Therefore our work must reduce this overhead to make it acceptable.

Unicast or multicast

Multicast could save intermediate bandwidth, ISP load, and client burden when a large number of clients in the same area are demanding the same movie. This requirement is not easy to be satisfied for a VoD application. Even if this requirement is satisfied, multicast requires efficient algorithm to establish the multicast path. If the path is not well established, the signaling messages and the inefficient transmission will greatly degrade its performance to be worse than unicast. Unicast is easy to handle and maintain, but it is not efficient when a large amount of client behind the same ISP demanding the same movie at the same time.

Since our target is to design a powerful and flexible server, not to design an efficient multicast path, we choose unicast in our design. This decision is reasonable under the fact that seldom does a VoD system has great number of clients request the same movie at the same time. Additionally, the expert system is a global control system that could be extended to be as multicast-enable platform in future development, as long as the intermediate routers support the multicast.

3.2.2 Server modules

Figure 3-3 gives the module structure of our expert server. In the left top corner of the figure, monitor is used to listen to the network notifications or client requests or feedback. Breakdown or recovery information of other servers is also sent to the monitor. Further more,

monitor records the current resource situation and calculate some statistical parameters to manage sessions.

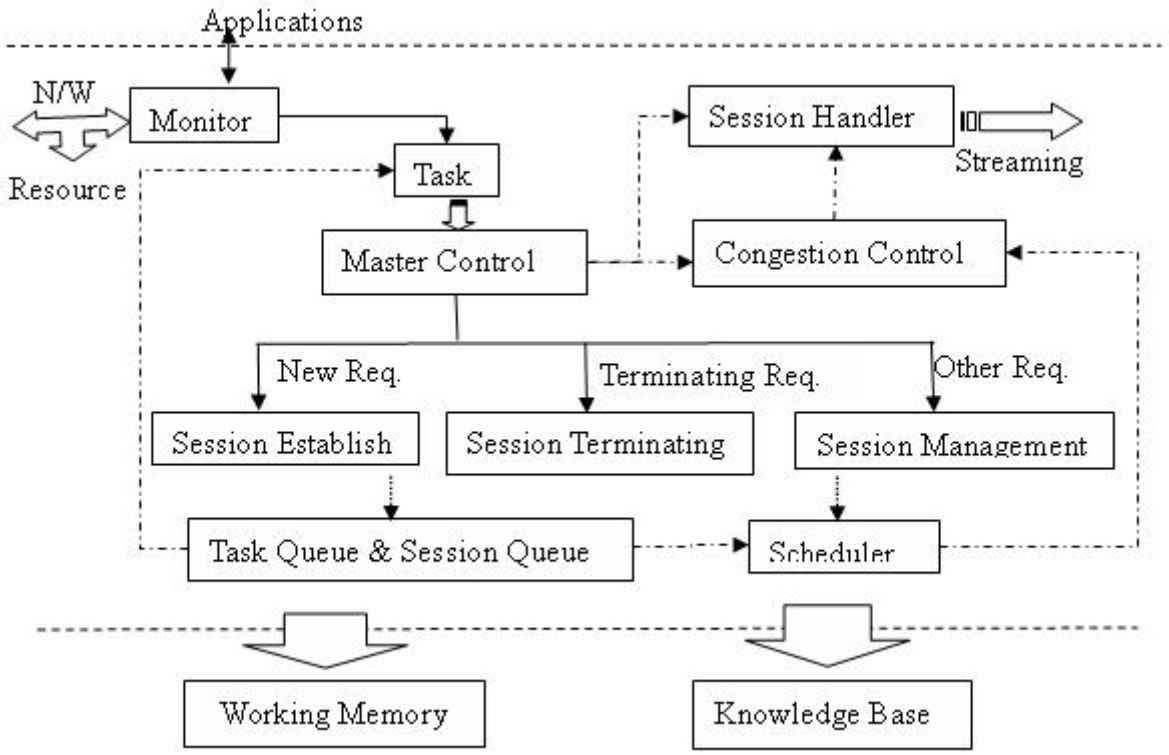


Figure 3-3 Expert system modules structure

Master control is an independent event driven routine. It receives requests from the monitor, which executes periodically every 100ms. Then it differentiates the request type and calls suitable sub-functions to serve the task. There are mainly three types of tasks: session establish or termination, media transmission, and session management (QoS). Media transmissions are controlled by the session handler. If the packet failed to be served, the master control module records the failure information. If the failure happens too frequently, the QoS management module will be called.

The session establishment module, one level down from the master control, must be called by the master control procedure. It is used to perform admission control and establish a new

session. If requested movie does not locate on the current station or the load of the current station is too high to serve more clients, this module is in charge of transferring the request to other suitable stations. This kind of job could be pre-coded into server main routine or may be supported by the rule base with station allocation and resource allocation rules.

The session termination module is used to terminate a session. Terminations may be caused by four reasons: resource shortage, no response within TIME_OUT, client requested, and normal finish. For terminations motivated by resource shortage, if there are resources reserved for the just terminated session and the server load is high, immediate adjustment is needed. For the other reasons, the server only releases resources without disturbing other sessions.

Session management is another dependent module called by master control. It performs QoS adjustment, parameters tuning, schedule table management, session states maintenance and congestion control.

Following is the detail introduction of the rule base. We divide the rule base into six groups. Each group is described with a simple example. The program structure of a rule and the corresponding procedures of condition examination and decision execution will be given in sub-section 5.1.2 rule base implementation part.

a) ***Meta rules***. These are special rules used to make upper level decision or to decide which group of rules is the starting point for searching. It may also contain rules to decide the search method based on time constraints.

   Example: IF *New Subscription* THEN *Search Session establishment/termination rule set.*

b) ***Session establish/termination rules***. These rules used to accept or reject the requests from clients, and set QoS level based on client buffer size, network delay, etc.

Example: IF *Termination Reason = Resource Shortage* THEN *Perform Online Monitor* AND *Execute Resource Reallocation Procedure*

c) ***Station Allocation rules***. The rules are in charge of selecting a proper server station from distributed server system for the new subscription request.

Example: IF *Server Load = High* AND *Movie cached at Station x* THEN *Forward Request to Station x*

d) ***Resource Allocation rules***. They are used to allocate and de-allocate the resources like CPU, memory (for packet queuing), and bandwidth.

Example: IF *Current Memory Usage < Low Threshold* THEN *Check Req. Arrival Rate*

e) ***Real-time Monitor rules***. The rules are used to monitor and update parameters of CPU utilization, memory usage, BW availability, and received notifications from network. They are also responsible to detect inactive/dumb session.

Example: IF *No response from a session for TIME_OUT* THEN *Report it a DUMB session*

f) ***Real-time QoS management rules***. These rules are responsible of process management and congestion control. They help the server to react on any violation of resources by adjusting transmission control parameters or changing delivery strategies.

Example: IF *Congestion Detected* THEN *Select Suitable Congestion Control Scheme*

With the above group segmentation, the inference engine starts from the meta-rules and searches only the request-related rule group according to the decision of meta-rules.

3.2.3 Decision Making Procedure

Here we use a potion of memory allocation rules as an example to illustrate the basic forward chaining decision making procedure in our system. The rule base inference procedure would be based on the following memory allocation rules in *Resource Allocation* rule set:

IF *Current Memory Usage* ∈*[Low Threshold, High Threshold] (i.e. Moderate)*

THEN *Check Requested Movie Bursty Rate*

IF *Current Memory Usage* < *Low Threshold* THEN *Check Req. Arrival Rate*

……

IF *QoS Level* = *Premium* THEN *Buffer* = *2*Average Sending Rate*

The logical relations of these rules could be illustrated in figure 3-4.

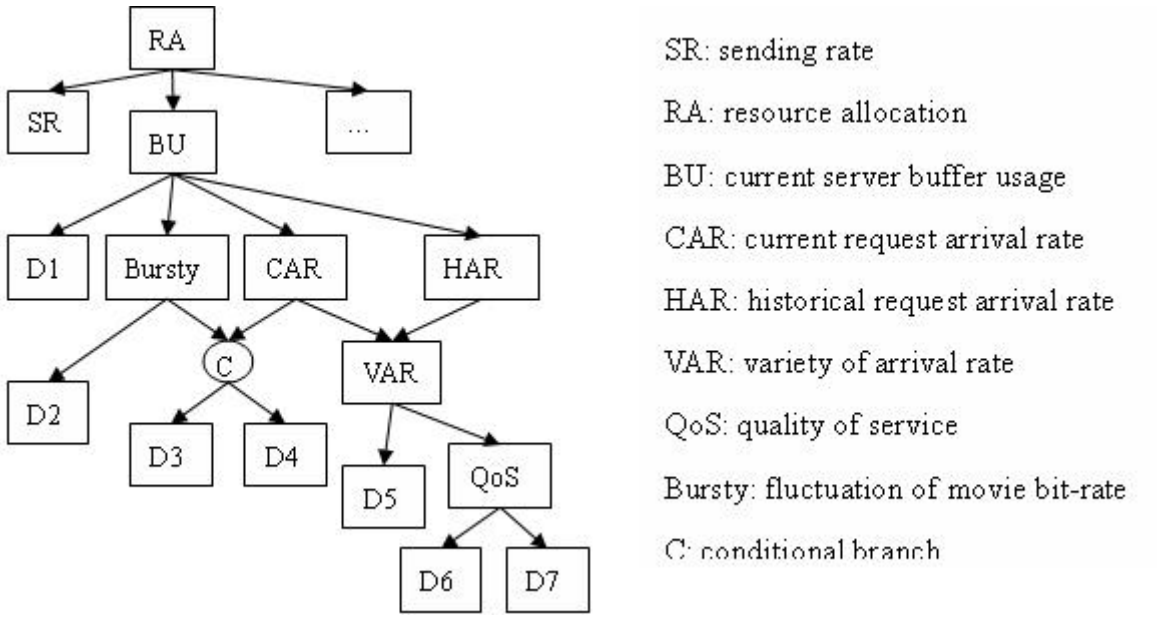


Figure 3-4 Rule relations for a resource allocation request

After searching, the inference tree is built as in Figure 3-5. To make the figure easy to read, we use full names for each rectangle. In the decision tree, regular rectangles are used for the rule call or function call, and the decisions are represented using circular rectangles. Conditions for branches are shown on arcs. When a resource allocation request is issued, the inference engine performs depth-first search. The searching sequence of branches is decided by meta-rules.

In the provided example, the search starts from the left-most branch, that is, from deciding the sending rate. In each branch, the inference engine searches the rule base using hill

climbing algorithm. Heuristics (conditions on arcs) are used to select the best child to trace further. The sending rate decided would be written into the corresponding session handler; meanwhile, the session related information in working memory is modified. Then the search process continues to decide the memory allocation. It checks current memory usage level and branches to the child nodes. If current buffer usage is moderate, it checks the trend of arrival rate. If arrival rate of new session establishment requests increases during the past monitored period, the decision should consider leaving more spaces for the coming users. If the arrival rate is stable in the monitored history, the resource is allocated merely according to the required QoS level. This solution is also written into the session handler and working memory. After that, the search process goes on to perform other resource allocations by repeating the same search algorithm.

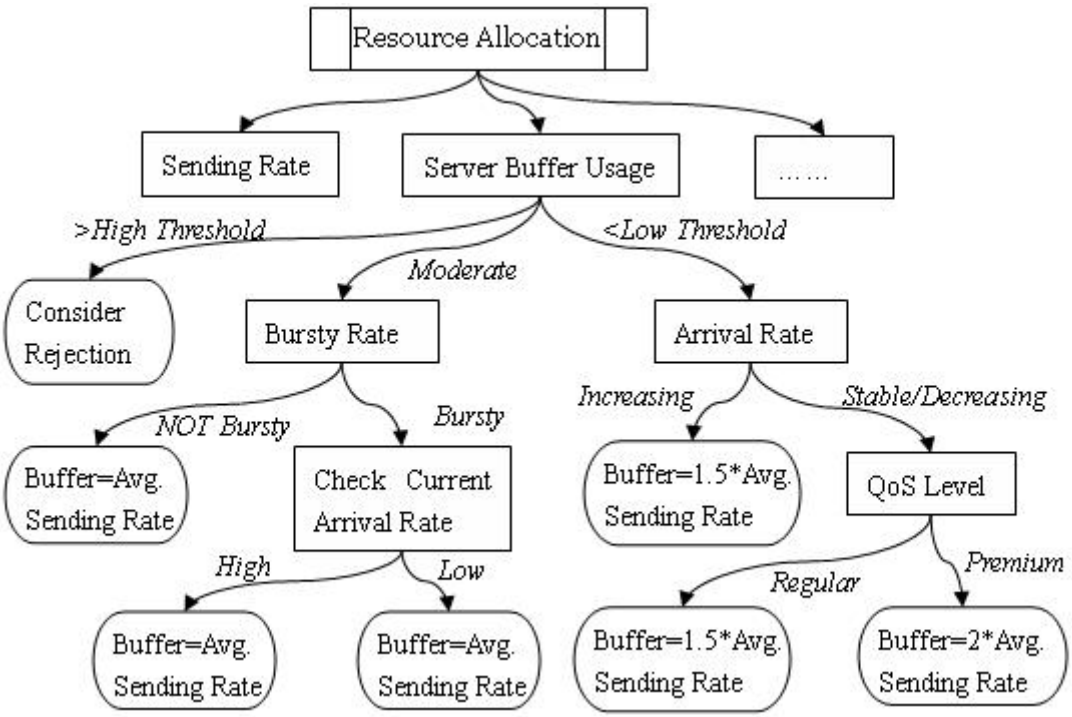


Figure 3-5 Decision tree for buffer allocation

The tree is set up automatically during the search process. The hill climbing search

terminates whenever a solution (circular rectangle) is reached. After all initial parameters for a session are set, the search process would be repeated to check whether modified parameters in working memory would cause other rules in related rule groups capable to be fired. This is a fine-grain adjustment on the final solution and we adopt beam search algorithm for it. For example, if sending rate is adjusted during the search, the corresponding buffer allocation would be fine-tuned accordingly immediately. The whole inference procedure ends until no rules can be fired under current situation.

From this example, we can see that the knowledge base and the inference procedure of an expert media streaming server are quite different with other recognition or planning expert systems. Those systems have large amount of loose related parameters and shadow edges on branch conditions, which need substantive heuristic rules to narrow the searching scope and direct to the solutions. Media transmission, on the contrary, requires apparent types of resources and the information of these resources are closely related to each other. In the example, the buffer allocated depends largely on the disk reading bandwidth and the sending rate; while the initial sending rate depending on the playback rate of the movie and the QoS level requested by the client. The inner relations among the rules made the inference procedure complete much faster with smaller fluctuations comparing to conventional expert system applications.

It may be argued that since the types of resources in media expert server are clearly defined and the links among parameters are close, why not pre-coding all IF-THEN clauses into server programs. Although the comparison of these two types of similar methods has been preliminary illustrated in the second last paragraph in section 1.3.3, we want to add the following points to make the explanations clearer.

Firstly, some heuristics are difficult to be mathematically modeled and sequentially coded, although they are practically helpful under heterogeneous networks. Additionally, the set of heuristics needed for a decision is not always the same. For these reasons, a more effective way would be coding heuristics as rules separated from the main program.

Secondly, the performance improvement and the overhead brought by the expert system are balanced. Compared to pre-coded search, the only additional searching overhead brought from using knowledge base comes from the online translation of parameter numbers into their actual values in the working memory. Since the knowledge base is parsed and linked in binary form beforehand (details are given in Chapter 5), searching it would not require much more time than searching IF-THEN clauses pre-coded in the server program.

Lastly, the expert system encodes all problem related expertise in data structures only; none are in programs. This organization enables great flexibility on knowledge base updating and system maintenance.

3.2.4 Communication among Server Processes

Figure 3-6 shows the relations among processes in the server program. We set up a packet queue for receiving requests from the network and clients; a task queue for information from current server station; a session link to manage active sessions on the current station. Five processes, task processor, packet processor, session handler, monitor, packet receiver, will work on these three queues as demonstrated in the figure; semaphores are applied to each queue for mutual exclusion. All global runtime parameters and resource tables are stored in working memory. The rule base is edited off-line. The line connecting the rule base to the working memory means that rules can modify the working parameters if necessary. The searches on the rule base could be initiated by the master control module, the modules called

by the master control, or by the real time monitor module. So we do not specify the source of the search arrow connected to the rule base in figure 3-6. Consequently, the decisions made by the rule base would be responses to those modules that initiate the search. Since the source is not specified in the figure, the returned decision arrow in the figure also does not have a specific destination.

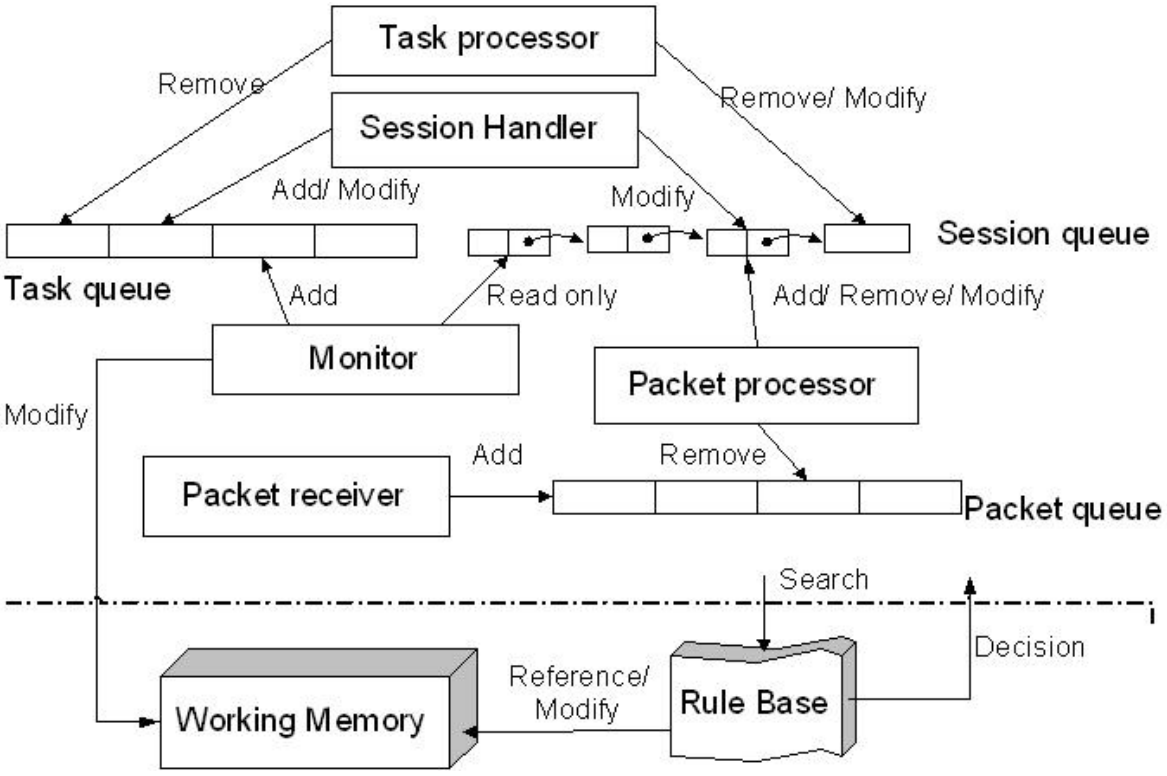


Figure 3-6 Communication among server processes

3.3 Summary

In this chapter, we first introduced the XML tool and searching techniques useful for our design. The design options and tradeoffs are explained. Then we indicated the level of our rule-based expert control system and the expert control components added in a conventional server. It is these added components that change the whole pattern of control for a conventional server to form a more powerful and flexible solution. The knowledge base is the

most important part for the intelligence of an expert system. We used explicit examples to illustrate the division of rule groups. Although search strategy is not the decisive factor for the intelligence, it is a significant factor for the overall performance. Usually the search procedure in planning expert system is conducted within a large amount of unrelated information and rules, and the subsequent searching time is unpredictable. However, the memory allocation example given in this chapter showed potential logic relations for its supporting rules. When directed properly, the search process converged very quickly.

The server structure diagram, the rule groups, the decision making procedures, and the communication model shown in this chapter are fundamentals for the consequent analytical and the experimental parts of the thesis. In the next chapter, we would first analyze the performance and estimate the theoretical server capacity before any real implementation. This will further exam the feasibility and scalability of the expert server system.

Chapter 4 System Performance and Capacity Analysis

In this chapter, we analyze the performance of the expert system. Methods and assumptions for analysis are introduced first. Then the server computational complexity is quantified. The average response time of requests and tasks will be analyzed based on the computational complexity. Other real time characteristics of the system are also considered. Finally the system capacity is estimated followed by a short discussion regarding the analytical results.

4.1 Introduction

Before analyzing the expert system, we first thoroughly examine factors that would impact the system performance and discuss their significance. Then the related notation and mathematical theories are introduced. The assumptions for the performance evaluation are given in the last part of this section.

4.1.1 System Performance Influence Factors

In general, the following six groups of factors are critical to the system performance:

1) Network parameters (BW, delay, loss rate, etc)
2) Client parameters (client buffer size, client requested QoS, new client arrival rate, average session duration, etc)
3) Disk bandwidth (Access bandwidth and data block transmission speed)
4) OS level task scheduling (Real time and non-real time tasks sharing the server resources)
5) Packet scheduling (Packet service sequence and session rate control)
6) Movie characteristics (Average playback rate, frame size, movie traffic bursty level, etc)

The former three factors are decided by the intermediate network, the client, or the supporting database hardware, which are not controlled by the expert system. Although uncontrollable, they influence the decisions made by the expert server and the overall transmission performance. We take them as given parameters for the system analysis. The task and packet scheduling methods are selected by the expert server during execution. They decide the effectiveness of the transmission, and therefore are factors concentrated in this chapter. The movie characteristic is negotiable by the server. Movie quality and coding strategies vary according to the requirement of clients and the available network bandwidth.

Due to the unique character of the expert system that all control decisions are made through inference on the rule base, the complexity of inference procedure should be analyzed first before investigating any other procedure. Inference procedure could be divided into five stages: test, match, activate, act, switch. The *Test* is to get the runtime parameter value in accordance with its parameter number and test the IF clause in a rule. The *Match* is to evaluate the corresponding THEN clause true or false. The *Activate* would activate the decision made by a rule if the condition is satisfied. The *Act* will perform the action decided by the rule. In our server, it represents the execution of a selected function. The *Switch*, as an equivalent action with the *Act*, stands for that the inference path branches to another rule.

In summary, this chapter focuses on task and packet scheduling algorithms analysis based on the given network, database, client and movie parameters. The inference procedure analysis operates as the key role within the whole analysis.

4.1.2 Theories for analysis

The foremost notation would be O(n) for time complexity analysis ([54]). O(g(n)) gives the upper bound of change speed for f(n) in the following notation.

Notation: We write f(n) = O(g(n)) if there exist constants c > 0, n_0 > 0 such that 0⩽f(n)⩽ cg(n) for all n⩾n_0. E.g.: $2n^2 = O(n^3)$ (c = 1, n_0 = 2)

The second important theory would be the queueing theory for service time analysis. Before introducing it, we will first take a look at the classification of delays along the media streaming transmission path.

A. Communication Delays

The delays experienced by packets in a transmission system can be illustrated in figure 4-1:

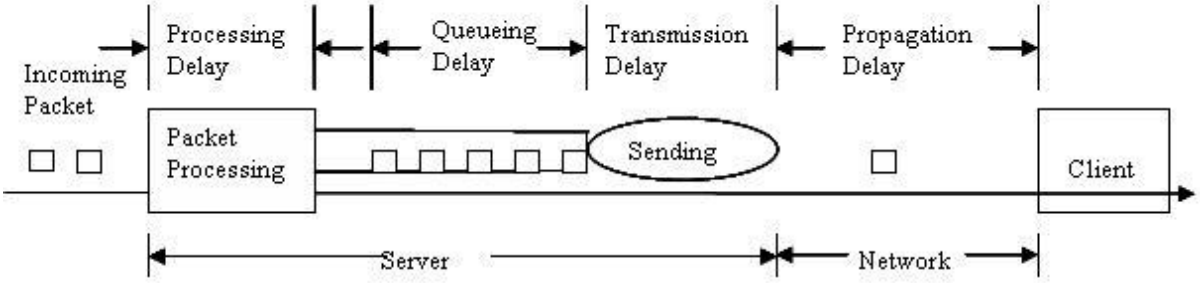


Figure 4-1 Communication delays

All mentioned delays will influence the transmission performance. The round trip time (RTT), a measurement of total delay after a packet is sent, is the decisive parameter for many rate control schemes. For example in QTSS, the RTT is re-estimated whenever an RTPStream object is called. The revised value is used to decide the timeout for feedback packets and regulate the sending speed. In most congestion control methods, delays are used together with the loss rate to decide the level of congestion and the necessary reactions.

For analysis without a measured RTT value, a tiny portion of delays could be ignored. Transmission delay varies with the packet length. As a usual UDP packet size is only several KB, the transmission delay of UDP packets are much less than processing and queueing delays and therefore could be neglected. The propagation delay is even smaller and the

retransmission is rarely happened. Hence only processing delay and queuing delay are considered in our analysis.

B. Queueing theory

Queuing theory is used to solve the system queuing status based on statistical characters of clients and servers. Statistical characters of a queueing model are described using following parameters (figure 4-2).

1) Arrival Process. Probability density distribution (λ) determines the request arrivals.
2) Service Process. Probability density distribution (μ) determines the request service times. In our server, the service time refers to the decision making time for a request.
3) Number of Servers. This is the number of servers (n) available to service the customers.

Using the short form of Kendall's notation, the common queuing systems can be represented as M/M/1, M/G/1, M/G/n, M/D/n, G/G/n, etc.

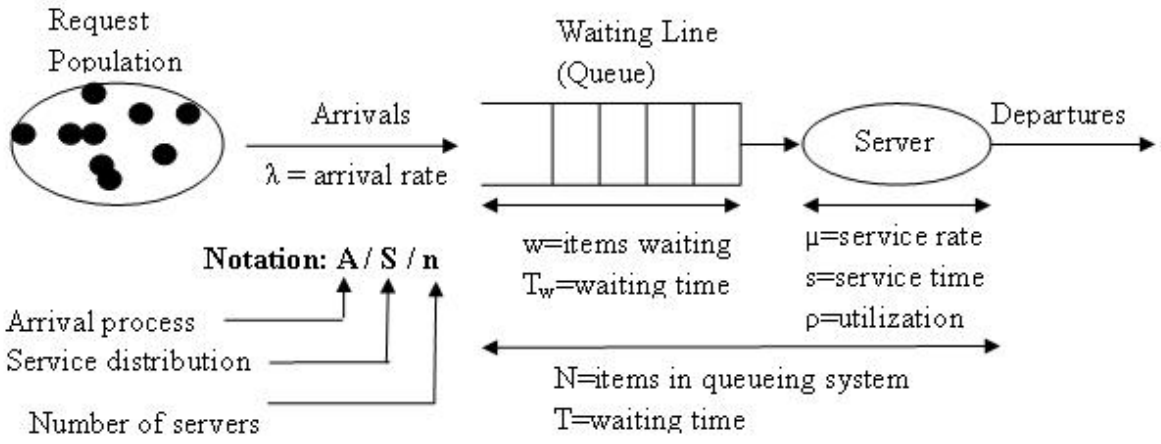


Figure 4-2 Queueing model

The fundamental of queueing theory relies on the Little's Theorem, which states that:

The average number of requests (N) in system can be determined from $N=\lambda T$.

Here λ is the average requests arrival rate and T is the average service time for a request. All queuing status parameters are calculated using Markov status chain based on this simple rule.

Take the common M/M/1 queue as an example. With reference to Kendall's notation, M/M/1 means a queueing model with both exponential distribution of customer arrivals and service times, and there is a single server.

Use P_0 denotes the probability that the system is idle. Then the utilization, the system busy probability is $1-P_0$. In steady state, the average arrival rate equals to the average departure rate. That is: $\lambda = 0P_0 + \mu(1-P_0) \Rightarrow P_0 = 1-\lambda/\mu$

Then the utilization factor ρ would be $1-P_0$, which is λ/μ.

The ratio $\rho = \lambda/\mu$ is also called the *traffic intensity* with unit Erlangs. Under the steady-state, it must be less than 1 for a single server queue. The probabilities of system with N requests in it can be solved using Markov status chain (figure 4-3).

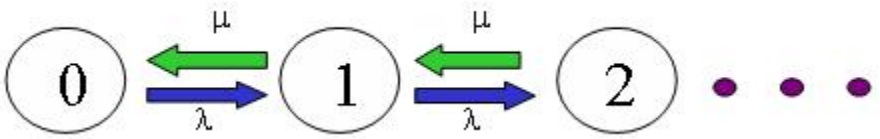


Figure 4-3 Markov chain for M/M/1 queueing model

In steady state, the probability for the system leaves state i must be the same as the probability of the system enters state i. That is:

$$\begin{cases} \lambda P_0 = \mu P_1 \\ (\lambda+\mu)P_1 = \mu P_2 + \lambda P_0 \\ \quad\quad\quad ..... \\ (\lambda+\mu)P_n = \mu P_{n+1} + \lambda P_{n-1} \end{cases}$$

Solve the above functions, we have $P_n = \rho^n P_0$. Thus the following parameters can be deduced.

Average number of requests in the system: $N = 0P_0 + 1P_1 + ... + nP_n = \rho/(1-\rho)$.

According to Little's theorem, the average in-system-time is: $T = 1/(\mu-\lambda)$.

Consequently, the waiting time $T_w = T - 1/\mu = \rho/(\mu-\lambda)$

The results will be used to calculate the capacity of the expert system in section 4.2.

4.1.3 Assumptions

We make the following assumptions for the analysis.

(1) All real time tasks in the server belong to media streaming applications.

(2) Unicast is considered.

(3) Clients follow a Poisson Process.

(4) The server has a sufficient buffer size for incoming requests. The requests can only be discarded once the deadline is exceeded.

4.2 System Performance Analysis

In this section, the system level performance analysis is conducted based on the theories and assumptions introduced in section 4.1. The analysis is performed in a progressive manner. First we analyze the computational complexity of each module and real time characteristics of an individual expert server, through which the scheduling delay are estimated. Then we calculate the maximum and the average service time (μ) for a request. Afterwards, queuing delay (Q) and response time (T) for a task or a packet with respect to different arrival rate (λ) is analyzed. Finally, we predict the number of clients a server could theoretically support.

4.2.1 Complexity and Computation Time

To balance the computational overhead and system accuracy, the master control program of expert system is set to execute every 100ms. The execution time of the expert control can be divided into two parts. One is the sequential request handling process used to respond to requests and tasks; the other is the rule base searching process used for decision making. In execution, the rule-base searching process is embedded into the sequential request handling

process. That is, requests are served by searching the rule base. If taking the searching process as a single statement, the decision making process would run purely in a sequential manner. The complexity is *O(m)*, where m is the number of sessions. Thus, the main execution complexity for the expert system program comes from searching the rule base. Now we focus on calculating the complexity of rule-base searching process.

Basically, the inference procedure uses best first search or beam search. The best first search selects the best child at each branch for further searching. Thus its complexity depends on the tree depth. For a rule base contains n rules, the highest depth is *n*. The complexity of best first search, as a result, is *O(n)*. The beam search is a truncated width first search. Its complexity depends on the searching depth and the beam width. The number of searching node spread geometrically as the tree level goes deeper. Thus the complexity can be written as $O(w^h)$, where w is the beam width and h is the search depth. Limited by the number of rules, the deepest depth can be reached is $1+w+w^2+w^3+...+w^h=n$. We can deduce the $h=\log_w[1+(w-1)n]+1$. Substitute it back into $O(w^h)$, the final solution will be $O((w^2-w)n+w)=O(n)$. Thus we get the conclusion that both two searching algorithms implemented in our design have the complexity of $O(n)$.

Now we can further the discussion to inference procedures. In the expert system, the firing of one rule may cause in chain the firing of other rules. Under this case, several rounds of searches are needed for an inference on the rule base. Here we present the worst case analysis for two typical conditions using best first search. The first condition, called C1, is that only one rule is fired during a search, and this fired rule is always the last rule searched. Another extreme condition, named C2, is that one rule is fired for every round of searching and finally all rules are fired in one round of inference. The fired rule in each round, similar to C1, is still

always the last one searched. For example, when we search a rule base with four rules inside, the sequence is r1→r2→r3→r4 in the first iteration. In the first round, r4 is fired, and it causes r3 ready to be fired. In the second round, inference engine searches r1->r2->r3 in sequence and fires r3. As a result of firing r3, r2 is satisfied, and similarly r1 is ready to fire after firing r2. The final situation is that all rules are fired after **n** iterations of searching. Of course, C2 is unlikely to happen during real execution because the conditions of some rules are contradictive to each other. In a set of these rules, if the condition of one rule is satisfied, the conditions of other rules inside the set will not be verified true, thus the other rules cannot be fired together for current inference. Here we use C2 as the worst case bound.

Using a test program, we get the average time t_s for searching one rule is approximately 0.01us and the time t_e for firing one rule is around 0.04 us. If there are n rules in rule base and the server need to search all rules to make a decision, then the worst case searching time for C1 and C2 are:

C1: $T = n \times t_s + t_e$

C2: $T = [n \times t_s + t_e] + [(n-1) \times t_s + t_e] + ... + [1 \times t_s + t_e] = n \times (n+1) \times t_s / 2 + n \times t_e$

From the Figure 4-4, worst case searching time increases linearly ($O(n)$) for C1, and bounded below 100 us with less than ten thousand rules. For C2, the worst case searching time increases exponentially ($O(n^2)$) to half a second as rule size goes to ten thousand. Practically, the searching time should follow the curve of C1 if rule base is properly organized. We will discuss the influence of complexity to system performance in the next subsection, considering the real time characteristics.

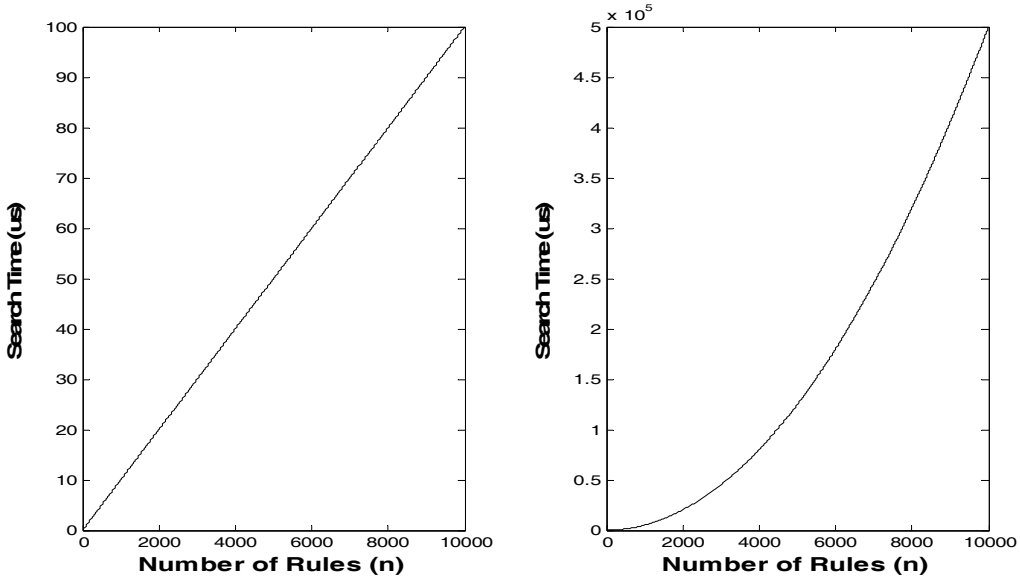


Figure 4-4 Worst case searching time for C1 and C2

4.2.2 Real time characteristics

In a media server, decisions are made within a specific deadline. For example, a feedback packet should finish processing within feedback intervals. The server system cannot break down at any time. These are the real time characteristics. Such a system needs the support of a real-time operating system. In analysis, we assume that the server computer OS uses a real-time kernel that can provide timing, preemptive thread scheduling, and fast interrupt response. Most deadlines in the server are soft deadline, which may be violated lightly without serious effect. So even without real time OS support, the system is still usable with somewhat degraded performance.

The key issue in analyzing a real time system is to evaluate the scheduling process. The efforts in finding workable solutions for real time scheduling problems have been progressed for many decades yet still no optimal method established. Even if only basic round robin or EDF scheduling is considered, the mutual exclusion and preemptive constraints make the analysis difficult or even impossible. Therefore, the thesis merely provides the offline

analysis based on static characteristics of the server code for a general prediction of the server performance.

4.2.3 Service time for tasks / packets

Based on the complexity analysis in the previous section, we can analyze the service time for tasks and packets. Tasks and packets are differentiated only because they are generated from different resources. Tasks come from the server whereas packets come from clients and networks. They are treated the same in expert control. Here we refer to them uniformly as requests.

Most services provided for requests need the search of rule base. However the complexity analyzed in sub-section 4.2.1 is a high level complexity that considers only the stage of test, match, activate, and switch. It did not count in the complexity of function (the act stage as introduced in 4.1.1) called by the rule. Besides that, there may have non-streaming traffic in the server. For non streaming requests, the service time is unpredictable since they are preemptive by real time tasks. Therefore it is impossible to give a uniform distribution of service time of streaming requests and background traffic under run-time uncertainties. However, the service time has an important characteristic that it is memoryless. That means the service time of current task is not influenced by the service time of previous tasks. With this feature, the expert control service time could be approximately modeled as an exponential distribution and the mean value is calculated in the next paragraph.

Taking C1 as the condition and setting the rule size as 2000, the time of each inference takes 20.04 us. If one decision is made from 5 iterations of inferences, it takes 20.04*5 = 100.2 us. Suppose the server needs to make 20 decisions for each round of control, and every decision must be made by inference on the rule base. The overall time taken is 2000.4us. Considering

the monitor interval of 100ms, the time portion for making decisions is $2000.4/10^5 = 2\%$. This is the best case mean service time and the calculation does not consider communication overhead among processes. To guarantee that the expert system takes no more than 10% of the CPU time to perform global adjustment, the monitor interval could be adjusted using:

*Monitor Interval > Avg. time for a decision*Avg. number of decisions for each monitoring / 10%*

The rest 90% of CPU capacity is dedicated to the real streaming transmission.

4.2.4 Queuing delay and response time

With a server cluster contains n servers, the incoming requests can be forwarded and served by any server in this cluster. The expert system will distribute the requests among themselves. So the requests come from every server node can be combined as a single queue. And the expert server system can be approximately described as an M/M/n/∞ queue. Consider a simple case first that all servers are selected equally for the coming requests. The mean request arrival rate is λ and the mean service time is $1/\mu$. The coming requests are classified as real time (class 1) or non real time (class 2) requests. They have different priorities for services and the system is preemptive. In sub-section 4.1.3, we assumed all real time tasks belong to streaming applications. The state diagram is shown in figure 4-5.

In the state transition diagram, green arcs represent the arrival and departure of media requests. The ellipses represent the probability of staying at a state (a,b), where a, b are the current number of requests of class 1 and class 2 in the system. From the state transition diagram, the streaming tasks would be served as if the server is dedicated entirely to them. Whenever they come, they preempt resources for services and this preemption may cause non-real time tasks suffer from starvation. Therefore, as we mentioned in the assumption part that the whole server is dedicated to media streaming applications, all requests submitted to

the system would be considered to be real time tasks with different deadlines. The schedule of tasks according to their deadlines is too complex to be uniformly modeled. We use the basic FIFO queue as an approximation.

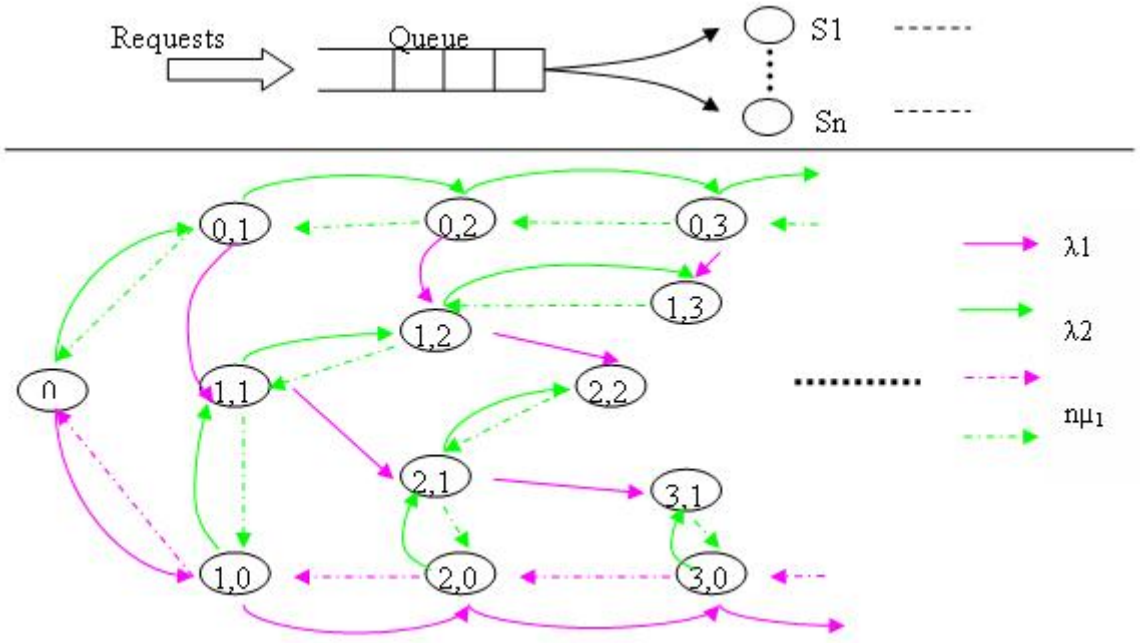


Figure 4-5 Queueing model and state transition diagram for a 2-priority M/M/n queue

Consider the real time tasks only, such an M/M/n queue model has been well investigated in literature. Here we list the formulas directly. Detailed deductions could be found in related analytical books ([55]).

(1) Probability that a customer has to wait: $P_Q = \dfrac{P_0(n\rho)^n}{n!(1-\rho)}$ ---(Erlang C Formula)

Where $\rho = \dfrac{\lambda}{n\mu}$ and $P_0 = \dfrac{1}{\sum_{k=0}^{n-1}\dfrac{(n\rho)^k}{k!} + \dfrac{(n\rho)^n}{n!(1-\rho)}}$

(2) Average time waiting in the queue: $T_w = \dfrac{\rho P_Q}{\lambda(1-\rho)}$

(3) Average time spends in system: $T = \frac{1}{\mu} + T_w$

(4) Average queue length: $N_Q = \frac{\rho P_Q}{1-\rho}$

(5) Average number of requests in system: $N = n\rho + \frac{\rho P_Q}{1-\rho}$

Suppose there are ten servers with utilization factor ranges from 0.01 to 0.99, and the average service time for the requests ranges from 1 to 40ms, the average response time could be plotted according to the results provided above. Figure 4-6 shows that service time increases when utilization factor and average service time goes up. The curve rises sharply under high load especially when the service time is large. This means the service time influences the average response time more than that of the utilization factor. When the average service time is 40ms and utilization factor approaches 0.99, a request needs to wait ten times (400ms) in average before being served.

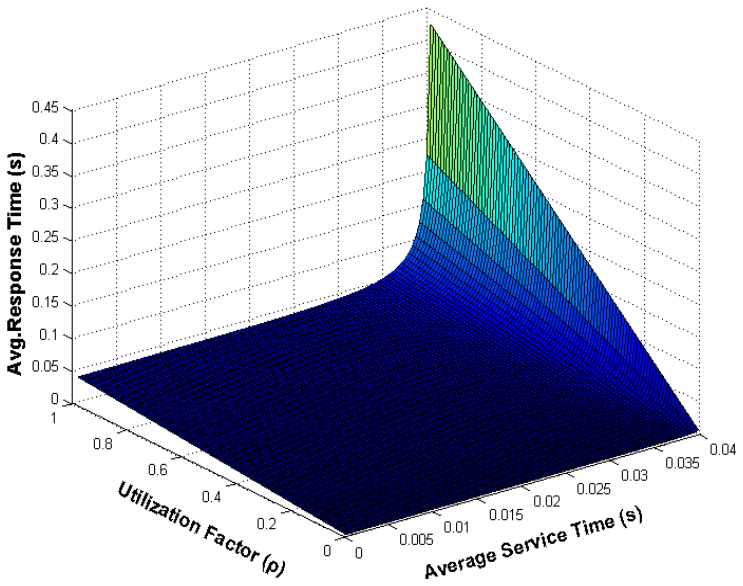


Figure 4-6 Average response time for M/M/10 queueing system

Now we take one server as a case study for some numerical results. Assume the buffer in

system is infinite. In real cases, the requests that were not served within a given deadline are discarded, although they were received. To guarantee all requests in server are valid to be served, each request should have a limited response time. Here we restrict the average in-system-time T within a certain time bound $T_{Threshold}$; then the following inequality function should be satisfied: $T_1 \le T_{Threshold}$

Substitute by the results of M/M/1 queue in subsection 4.1.2:

$$\frac{1}{\mu-\lambda} \le T_{Threshold} \Rightarrow \frac{T_s}{1-\rho} \le T_{Threshold} \Rightarrow \rho \le 1-\frac{T_s}{T_{Threshold}} \qquad \text{where } T_s = \frac{1}{\mu}$$

The arrival rate would be limited by

If each client generates *m* request in average, the supportable client is:

Suppose each session generates ten requests a second and a decision must be made before next frame of this session sent out, the maximum deadline for the current request would be equal to the frame interval, that is 1/25 = 0.04*s*. Hence the supportable number of client is:

N_c <= 0.1/(10*100.2*10-6) - 0.1/(10*0.04) <= 99.55.

4.2.5 Capacity of a Single Server

There are two factors limiting the traffic that a transmission system can support: the server processing speed and the network bandwidth. For a normal CPU, the maximum number of clients derived in sub-section 4.2.4 is around 99. Of course this number is reasonable only when monitor interval is properly set, real-time tasks are scheduled preemptively, and the server buffer is large enough. Yet these 99 users will consume 99x1.25 Mbps bandwidth = 123.75Mbps (normal VCD quality). From the comparison, the capacity bottleneck of the server would be the outgoing bandwidth. Since there is traffic for protocol, feedbacks, and

other applications, we cannot grant total bandwidth to the streaming. Usually only around 70% of bandwidth could be used for streaming data for the system stability. Therefore, we estimate the capacity by:

Server capacity =70% Outgoing BW/ Average Sending Rate.*

According to the above formula, the server capacity depends on the average sending rate of a single stream given a specific outgoing bandwidth. Thus in the following section of case study, we take sending rate (bandwidth usage) as the main criteria to illustrate the rule-based system performance.

4.2.6 Multicast Analysis

Multicast groups users within the same sub-network and deliver a single steam to their ISP. ISP is responsible to forward the stream to all users through prescribed tunnels. Using such a transmission topology, server needs to maintain a multicast tree or group list. For each request, which may be handled by the expert control, the server processes it the same way as in unicast. So multicast will not impact the average service time for a request, only brings overhead on multicast group maintenance and impact the bandwidth utilization. It saves a lot of backbone network bandwidth on data transmission, meanwhile it requires well-designed protocols to protect the network from overwhelmed by those acknowledgements returned by all multicast clients.

There are many types of multicast protocols, like sender-initialized, receiver-initialized, and tree-based protocols. Sender-initialized protocol needs positive acknowledgements (ACKs) to be sent back to the server for every packet correctly received. While a receiver-initialized protocol needs negative acknowledgements (NCKs) that sent back to the server only for lost or corrupted packets. Tree-based protocol collects ACKs/NCKs hierarchically to decrease the

bandwidth waste of acknowledgements. In the bandwidth analysis presented in this sub-section, we consider the most basic and commonly used protocol, which is sender-initialized multicast protocol, and use the method that introduced in [56].

W = BW for initial transmission + BW for retransmissions + BW for receiving ACKs

$$W = W_d(1) + \sum_{m=2}^{M} W_d(m) + \sum_{i=1}^{L} W_a(i) \Rightarrow E(W) = E(M)E(W_d) + E(L)E(W_a)$$

Where W is the bandwidth required for a multicast session. $E(W)$ is the expected bandwidth consumption. $E(M)$ and $E(L)$ are the expected number of retransmissions and acknowledgements. W_d is the bandwidth required for a data packet. W_a is the bandwidth required for an ACK packet. $E(L)$ depends on the value of $E(M)$.

$$E(L) = N * E(M)(1 - p_d)(1 - p_a)$$

N is the total number of clients in the multicast group. p_d and p_a are loss probabilities for data and acknowledgements respectively. So that $(1 - p_d)$ is the probability of a data is not lost and $(1 - p_a)$ is the probability of an ACK is not lost. We have:

Probability of number of retransmission is less than m times = 1- probability of number of consecutive retransmission is m

That is:

$$P(M \le m) = 1 - p_{re}^m$$

$$p_{re} = p_d + (1 - p_d)p_a$$

That means either a data packet loss or an ACK packet loss will cause a retransmission. As the client receive procedures are independent from each other, we have,

$$P(M \le m) = \prod_{r=1}^{N} P(M_r \le m) = (1 - p_{re}^m)^N = \sum_{i=0}^{N} C_N^i (-1)^i p_{re}^{im}$$

$$P(M=m)=P(M\le m)-P(M\le m-1)=\sum_{i=0}^{N}C_N^i(-1)^i\,p_{re}^{i(m-1)}(p_{re}^i-1)$$

$$E(M)=\sum_{m=1}^{\infty}mP(M=m)=\sum_{m=1}^{\infty}m\sum_{i=0}^{N}C_N^i(-1)^i\,p_{re}^{i(m-1)}(p_{re}^i-1)=\sum_{i=1}^{N}C_N^i(-1)^{i+1}\frac{1}{1-p_{re}^i}$$

Multicast needs to cache data until all clients receive it correctly.

Average server side buffer size = current data + Σ(size of sent data waiting for ACK)

$$= \mathrm{E(M)} \times \mathrm{RTT} \times \mathrm{S_r}$$

$$E(B)=RTT\times S_r\times E(M)=RTT\times S_r\times\sum_{i=1}^{N}C_N^i(-1)^{i+1}\frac{1}{1-p_{re}^i}$$

The multicast analytical results could be used to estimate the bandwidth usage of current streams, or to predict the potential server ability of admitting new streams. If multicast is supported by the server, our expert control could use these analytical results to design effective admission control rules.

4.3 Summary of Performance Analysis

In this chapter, we generally analyzed the rule-based expert system and estimated its performance. The server computational complexity is quantified to be within the range of O(n) and O(n^2), where n is the size of the rule base. Considering the configuration of the rule base where rules are written for different questions and some of them can never be fired together, it is very unlikely that only one rule is fired in a round of search but consequently all rules are fired separately during n times of search in a reference. If this situation could not happen, then the exponential bound will seldom be reached. Hence we expect that using the linear bound for future estimation to be reasonable.

Based on this assumption, we gave a formula to calculate the monitor interval. The decision

of monitor interval is influenced by average service time, requests arrival rate, communication overhead, and CPU portion for the monitor. In our design, we suggest to allocate less than 10% of the CPU time for the control procedures to maintain satisfactory time for transmissions. The formula can be applied to set the control interval as long as we know the average service time of requests.

We employed the M/M/n queueing theory for the estimation of average response time, average queue length and blocking probability. Within the two classes of requests in the system, the media transmission requests have the superior priority to other non real time tasks. The analysis of streaming tasks, as a result, could ignore the influence of other disturbing tasks. Thus the system meets the results of M/M/n queue provided in sub-section 4.2.4. The numerical results with one server showed that a server can support around 99 customers simultaneously, if each customer generates ten requests in average. From this number, the bottleneck of the server capacity is revealed to be the outgoing network bandwidth, not the CPU power. So in the last subsection, the overall system capacity was estimated by consuming approximately 70% bandwidth of the outgoing link. Bandwidth requirement under multicast situation and corresponding server buffer usage are also analyzed at the last sub-section.

From the complexity and capacity analysis in this chapter, the added expert control does not decrease the number of client potentially supportable by a normal media streaming server. The expert system structure provides us enough flexibility to adjust the monitor interval, the service time, and the priority level to limit the control overhead within a reasonable range. We will implement the expert server system in the next chapter and conduct a case study to test its performance.

Chapter 5 Implementation of Rule-Based Expert Server System

In the chapter, the detailed implementation of the rule-based expert server system is introduced. All experiments are performed on a test-bed using the parameters obtained from real Internet. We investigate several aspects of the expert control performance with around 1000 rules in the knowledge base. The experimental results are given to show the comparisons between basic Apple's QuickTime Streaming Server (QTSS) and QTSS with expert control.

This chapter is organized as follows. Section 5.1 introduces the test-bed configurations, classification of rules, and basic modules of Darwin Streaming Server. Section5.2 gives a thorough study of the methods and algorithms used in expert control, followed by a test scenarios map and their evaluations in real Internet. The experiments and discussions are shown in Section 5.3, followed by a short summary.

5.1 Introduction

This section introduces the computer specifications in our test-bed and network topology of the experiments, the file format of the rule base and its binary structure after parsing and linking, the details of rules in the rule base, and the background information of basic QuickTime Streaming Server. They are the fundamental configurations of experiments that are going to be presented in the sections afterwards.

5.1.1 Experiment computer configurations

The general structure of designed test-bed is shown in Figure 5-1. There are three DELL

Precision T5400 PCs; each has 19 virtual machines (VMWare) installed. In the figure, it is illustrated as 20 small computers reside on a physical machine. Similarly, two DELL Optiplex 755 PCs are selected with 9 virtual machines installed. All virtual computers have QuickTime Player, and all physical machines have QTSS. Table 5-1 lists the parameters of these PCs. These PCs will be selected and re-configured according to the purpose and requirements of each experiment. We will introduce the detailed experimental configurations separately in the section of Experiments and Discussions.

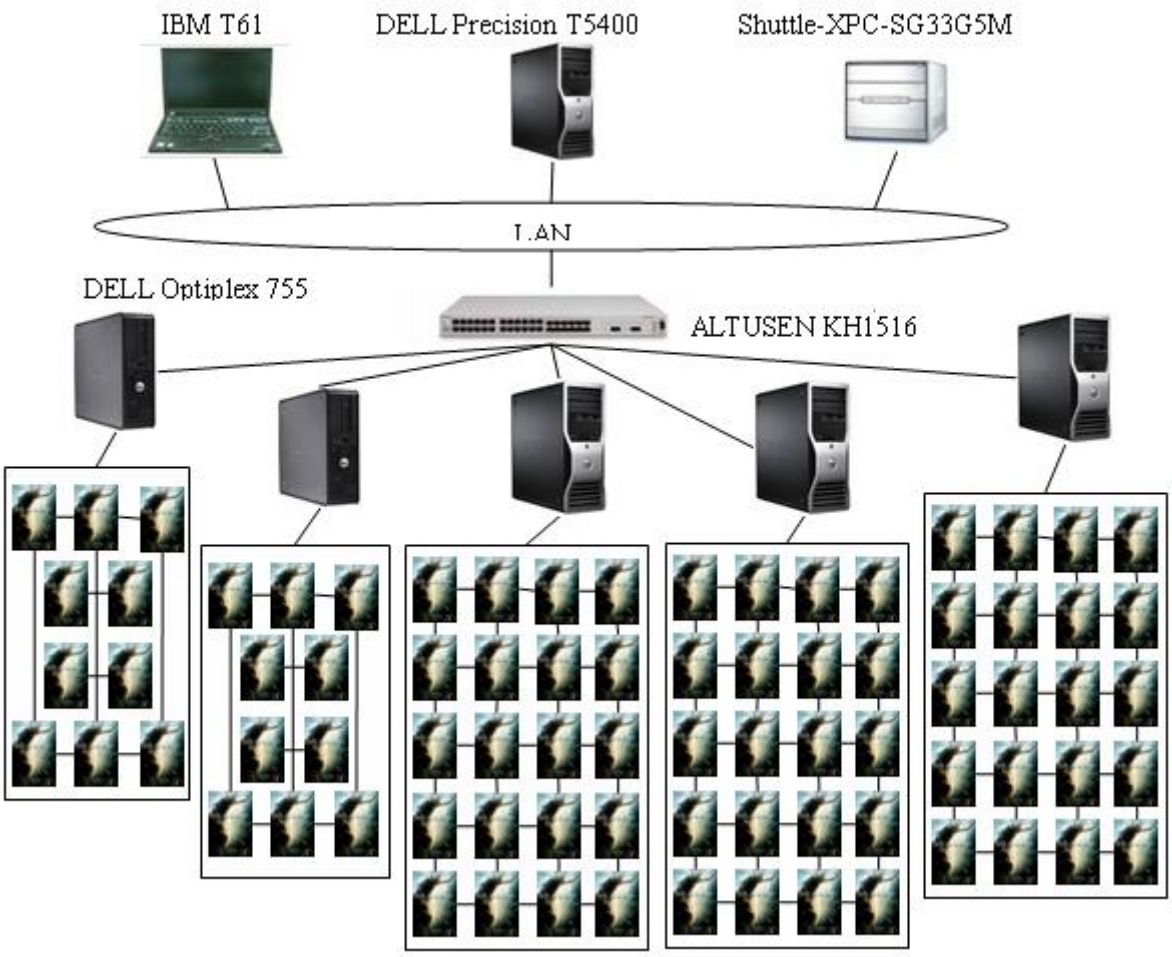


Figure 5-1 Basic structure of test-bed

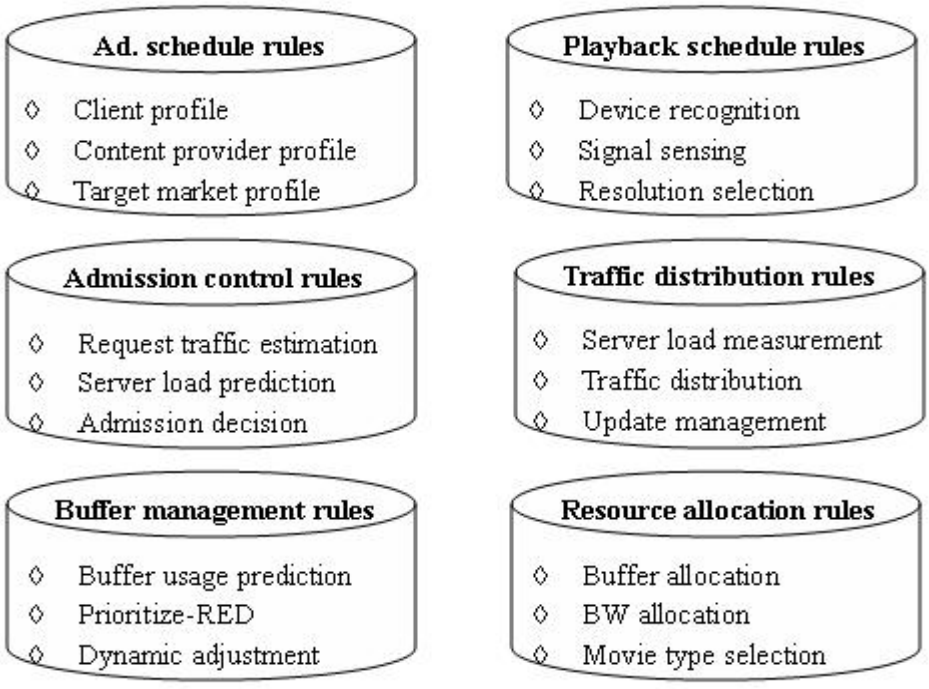
	Processor	**Memory**	**Network**	**Operation System**
DELL Precision T5400	Intel Xeon	7.8GB	1G	Fedora 7.0

	4 CPU@2.83GHz			
DELL Optiplex 755	Intel Quo'2 Quad Q7600@2.66GHz	7.8GB	1G	Fedora 7.0
Shuttle-XPC-SG33G5M	Intel Core2 Quad Q6600@2.4GHz	2*2GB	1G	Fedora 7.0
IBM T61p	Intel Core2 Duo@2.4GHz	2GB	100M	Windows XP

Table 5-1 Device parameters

5.1.2 Rule Base Implementation

This sub-section illustrates the real implementation of rules. The rule base consists of around 1000 rules. Around 40% of them are QoS and congestion control rules. Nearly 10% are advertisement playback schedule rules. Around 25% are session management and monitor rules. The other 25% are meta-rules, admission control rules, traffic distribution rules, resource allocation rules, and buffer management rules. Figure 5-2 shows the rule buckets in our rule base. Meta rules are not shown in a bucket because they only have a single function to direct the search to one or several proper rule buckets.



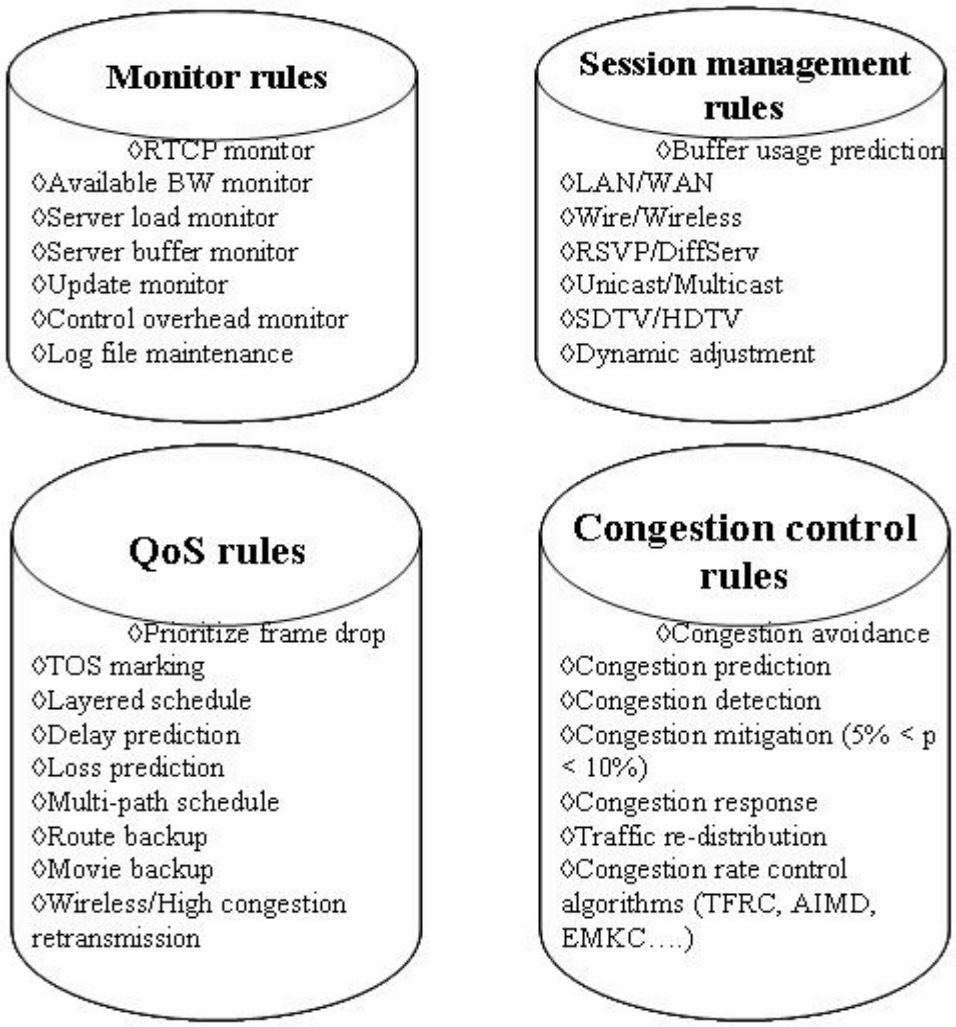


Figure 5-2 Rule buckets in the rule database

The rule base was constructed in XML language. As introduced in sub-section 3.1.1, the DTD file for rule base is defined as:

```
      <?xml version="1.0" encoding="UTF-8"?>
      <!ELEMENT rule_base (rule)*>
      <!ELEMENT rule (condition, (rule_call|func_call)*)>
<!-- Attributes of a rule  rule_no: a unique rule number -->
      <!ATTLIST rule rule_no CDATA #REQUIRED>
      <!ATTLIST                          rule                          type
      (Meta|StAlloc|ResAlloc|Sched|QoSMana|Monitor) #REQUIRED>
<!- - - - - - -Definition for Condition - - - - - - - ->
      <!ELEMENT condition (#PCDATA | var | number | func_chk)*>
```

```
      <!ELEMENT var EMPTY>
      <!ATTLIST var name CDATA #REQUIRED>
      <!ELEMENT number EMPTY>
      <!ATTLIST  number value CDATA #REQUIRED>
      <!ELEMENT func_chk (para*)>
      <!ATTLIST func_chk fc_name CDATA #REQUIRED>
<!-- - - - - - - Definition for Actions - - - - - - - ->
      <!ELEMENT rule_call EMPTY>
      <!ATTLIST rule_call rc_no CDATA #REQUIRED >
      <!ELEMENT func_call (para*)>
      <!ATTLIST func_call fc_name CDATA #REQUIRED >
      <!ELEMENT para EMPTY>
      <!ATTLIST para type (num|var|str) #REQUIRED >
      <!ATTLIST para value CDATA #REQUIRED >
```

According to the DTD, rules are written with the following format.

```
<rule rule_no="78" type="Monitor">
      <condition>
            <func_chk fc_name="GetSePara">
                  <para value="SeUnderConsi" type="var" />
                  <para value="Status" type="str" />
            </func_chk>
            EQ
            <number value="0" />
      </condition>
      <func_call fc_name="TaskGenerator">
            <para value="0" type="num" />
            <para value="ENDSESSION" type="str" />
      </func_call>
</rule>
```

This rule is used to detect a dumb session. It checks the status of a session. If the status equals to 0, that is, no reply from client within TIME_OUT, a session-terminate task is generated.

The XML rule base is parsed by Expat2.0 and the returned characters are handled by our explanation program attached on the expert server. This explanation program combines the characters obtained from Expat and parses them into the rule structure defined as follows:

```
struct rule_t {
      int   type;           /* The validity of this rule item */
      struct cond_t * condition; /* root of the condition tree */
      struct act_t  * action;  /* link list of actions to take */
};
```

In the binary structure, each rule contains a rule type, a condition root pointer and an action root pointer. The type indicates which group the rule belongs to. The condition part is a binary tree, where operators are parents and operands are leaves. The action part is a link list. The structure of conditions and actions are shown below:

```
/* Node structure for the condition tree */
struct cond_t {
      int   type; /*Type of Node, operator/number/variable/function*/
      int   op_id;          /* ID of the operator */
      int   var_id;         /* variable id */
      double value;         /* number value */
      int   func_no;        /* Function call no */
      struct para_t * para; /* parameters for the function */
      struct cond_t * left;
      struct cond_t * right;
};
/* Node structure for the action list */
struct act_t{
      int func_no;          /* function call number*/
      struct para_t * para; /* parameters for the function */
      int rule_no;          /* rule call number */
      int   type;           /* type of the action, function/rule */
      struct act_t * next;  /* next action */
};
```

Conditions are evaluated by walking from leaf to root when the online value of variables are available during run-time. Actions consist of function call and rule call. All parameters referenced by rules and functions called by rules have their unique identity number. For consistency, we use functions to modify or check the real-time value of server parameters, although these parameters are accessible directly by rules. If a rule needs to be called, the corresponding rule number is given; while if a function is called, the function number and the parameter link head are passed to the function. The parameter link gives the parameters needed by the expected function. Each node in the parameter link is a structure, which contains not only the value of the parameter but also the type of it. Refer to the previous dumb session detection rule example, the rule number 78 and the rule type 'Monitor' is given to activate this rule. At the first line of condition part, a function named GetSePara is called to check a session's status value. The two parameters, session number and attribute name, are passed to GetSePara function through the 'para' link. If the condition is satisfied, the function TaskGenerator is called to generate a task to end the unresponsive session. In our implementation, we allow nested function calls in passed parameters.

To make the inference procedure faster, the translated rules are put into a rule table at a fixed position decided by its unique rule number given in the XML file. Another look-up table maintains the rule numbers belong to each group. All these works finish at the startup of the expert server. The rule numbers in a group will be sorted by their reference sequence during the first inference. The sorted order is considered to be the most likely sequence pattern for future inferences.

Until now, we have introduced the setup and knowledge-base information for our first type of experiment, which is conducted on the local area network. In the next sub-section, we will

introduce the platform for the experiments, that is, the QuickTime Streaming Server.

5.1.3 QuickTime Streaming Server

Apple's QuickTime Streaming Server (QTSS), also called Darwin Streaming Server, is an open source version of the media server technology that allows user to send streaming media across the Internet using the standard RTP and RTSP protocols. Streamed media can be viewed by both Macintosh and Windows users using QuickTime Player or any other application that supports QuickTime or standard MPEG-4 files. The server can be used to delivery live media or videos on demand, or broadcast. In the following paragraphs, the server structure is illustrated using figure 5-3, which is provided in Apple's *QTSS Modules Programming Guide* document [58]. From the figure, QTSS server consists of four parts.

1) The server's own Main thread. The Main thread checks to see if the server needs to shut down, log status information, or print statistics.
2) The Idle Task thread. The Idle Task thread manages a queue of tasks that occur periodically. There are two types of task queues: timeout tasks and socket tasks.
3) The Event thread. The Event thread listens for socket events such as a received RTSP request or RTP packet and forwards them to a Task thread.
4) One or more Task threads. Tasks threads receive RTSP and RTP requests from the Event thread. Tasks threads forward requests to the appropriate server module for processing and send packets to the client. By default, the core server creates one Task thread per processor.

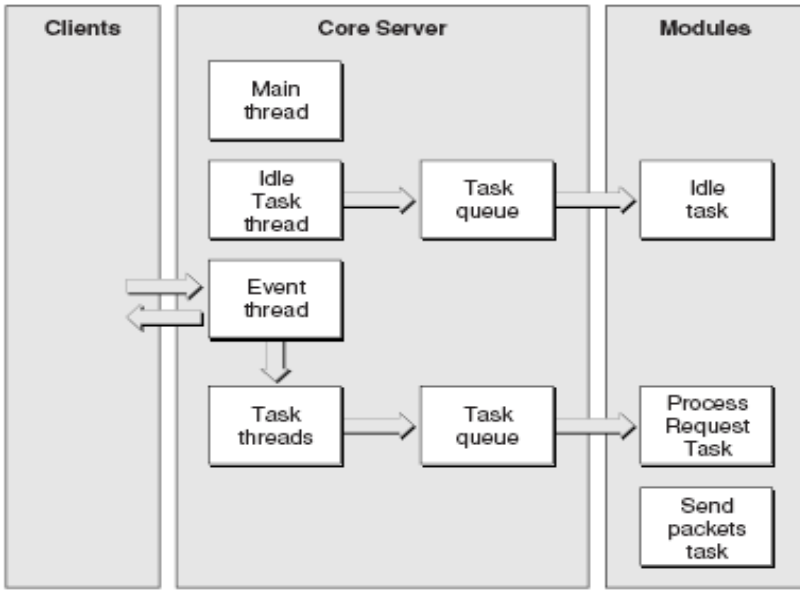


Figure 5-3 QTSS server structure

The Streaming Server consists of one parent process that forks a child process, which is the core server. The whole server runs by event triggered tasks. Each Task object has two major methods: Signal and Run. Signal is called by the server to send an event to a Task object. Run is called to give time to the Task for processing the event. As an asynchronous server, the communication mechanism for events is performed by generalizing Task objects.

QTSS uses a shared buffer for all flows. Video data is moved to the buffer when required and sent out immediately. The scheduling method it uses is basic round robin, which serves each session in a fair and sequential way. The QTSS uses Reliable-UDP and flow control together to perform the function of the congestion control. The so called Reliable-UDP is a modified protocol that imitates TCP to quantify client satisfactory by asking clients to send back acknowledges periodically. Flow control increases or decreases BW allocation by the information obtained from RTCP packets.

Now we compare the QTSS with our expert server on main modules. Obviously, both QTSS and the expert server designed in this thesis are event driven. QTSS Event Thread module is

equivalent to the Packet Receiver module in the expert server. QTSS Task Thread module performs similar functions as Master Control and Session Handler modules in the expert server. Besides these similarities, there are differences between the two servers. QTSS is a single thread server while the expert server we designed is a multi-task server that contains four major threads: Monitor, Master Control, Packet Receiver, and Session Handler. However, the major and most important difference is the procedure of making decision. The QTSS is traditionally programmed while the expert server refers to the rule base for solutions.

In the following case study, we will introduce the selected buffer management methods, scheduling methods, and congestion control methods used in the experiments.

5.1.4 Experiments and Evaluations

We designed five experiments to demonstrate the smart behavior of our expert server in the real Internet.

A. *Effective admission control and load balance*

   With the heterogeneous capacity of servers, there is an optimal distribution of sessions among these servers. However, the incoming requests are randomly issued to a server in the server cluster. It will greatly enhance the whole system capacity if requests could be distributed reasonably among servers. Thus, we design four types of initial request distributions to compare the load balance function of our expert control with the basic QTSS. Admission control is also added to cooperate with QoS management and congestion control.

B. *Playback scheduling*

   Several videos could be displayed in a scheduled sequence and advertisements could be inserted according to content provider's demand and client profile. This function is

enlightened from a project that needs to automatically update the VoD content and advertisement playback schedule to their subscribers. The function could be extended to a large amount of usages. For example, it may be used to automatically control the playback when client device is under different environment and status.

C. *High Definition (HD) streaming rate control*

High definition videos require much larger network resources compare to normal videos. They are killer applications on current IP network. However, more and more applications such as Cisco Telepresence products ([83]) require the scenery from each party to be as clear as possible, like everybody is in the same conference room. For these cases, rate control is crucial to guarantee video quality and avoid congestion. We use rate control methods that will be introduced in sub-section 5.2.4 to realize the rate control.

D. *Streaming handover*

In our expert server, we consider a challenge scenario that the terminal devices are changed during playback and the expert control could perform online streaming handover. Cases we consider are the client device switches from a handphone with slower IP network to plasma TV display with a fast IP network, or vice versa.

E. *Congestion control*

Congestion control is always the most important issue as long as network resources are shared. In our approach, the congestion control is divided into four steps, congestion avoidance, congestion mitigation, congestion response, and traffic redistribution. The four steps are deployed in sequence as the severe of congestion increases.

5.2 Experiments

The performance of the expert server system depends largely on the efficiency of rules and methods. Therefore it is difficult to give out uniform experimental results without a standard knowledge base. Here we demonstrate the performance of expert control compare with basic QTSS server for above designed scenarios. First we introduce the configurations of our test bed, the movies, the target performance parameters, and the critical methods and rules used for the expert control.

5.2.1 Experiment Configurations

Table 5-2 and Table 5-3 list the movie parameters and experimental configurations used in our experiments. Stream-1 to stream-3 in Table 5-2 are the same movie tailored to different resolutions, in which stream-3 (movie 720p) could be classified as a high definition movie. The configurations will be slightly tuned in experiments, which will be future explained in the corresponding sub-section. Performance parameters that we plan to measure are shown and evaluated in Table 5-4.

	320 (Stream-1)		**480p(Stream-2)**	
	Audio	Video	Audio	Video
Codec	MPEG-4 AAC LC	H264 Main@1.2	MPEG-4 AAC LC	H264 Main@3
Average Rate (kbps)	98	200	205	1925
Duration	131.2	131.2	131.2	131.2
Frequency (Hz)	44100	-	44100	-
resolution	-	320 x 172	-	848 x 448
Target Frame Rate (Frames/sec)	-	24	-	24
Size (KB)	4851		34181	

	720p(Stream-3)		**iPhone Keynotes (Stream-4)**	
	Audio	Video	Audio	Video
Codec	MPEG-4 AAC LC	H264 Main@3.1	MPEG-4 AAC LC	H264 Baseline@3
Average Rate (kbps)	457	6026	128	1504
Duration	131.2	131.2	6325.208	6325.198

Frequency (Hz)	48000	-	44100	-
resolution	-	1280 x 688	-	640 x 352
Target Frame Rate (Frames/sec)	-	24	-	30
Size (KB)	103896		4851	

Table 5-2 Parameters of movies

Metrics	Value	Evaluation
Protocols	Session setup: RTSP (RTP & RTCP)/UDP	Used in commercial servers
Router	0~2	Use computers to emulate parallel routes with various settings
Concurrent connections	70~72	Number of nodes in test-bed
Playback rate	300kbps~6Mbps	Standard definition movie
Loss rate	LAN/ WAN/ Wireless (802.11g)	Test on real Internet/ Wireless network
Delay	LAN(0ms~10ms)/ WAN(100ms~300ms)	Test on real Internet/ Wireless network

Table 5-3 Experiment configurations

Measure parameters	Explanation	Evaluation
Connection success rate (CSR)	The ratio of active video streams / the total attempted video streams	To testify the effective admission control and traffic distribution
Join latency	Time to start a session	Startup delay
Throughput	Bps	It reflects the streaming characteristics, the smoothness of the flow, and the bit-rate.
Client buffer occupancy	MB	It reflects the problem of underflow / overflow
Inter-arrival Jitter	(us)	Smoothness of the streaming
CPU taken by expert system	%	Overhead test (run time QoS management overhead, measured every 100ms)
Video presentation quality	Resolution, corrupt frames, frame rate	User side quality measurement

Table 5-4 Measurement parameters

The measurements in Table 5-4 are analyzed using captured packets from WireShark network packet analyzer. In Table 5-3, the loss rate and delay is extracted from RTCP reports. The detailed calculation could be found in RFC 3550. Here we only explain the loss rate

calculation because there are different ways to compute it in the literature.

First, the number of packets expected can be computed by the receiver as the difference between the highest sequence number received (s->max_seq) and the first sequence number received (s->base_seq). Since the sequence number is only 16 bits and will wrap around, it is necessary to extend the highest sequence number with the (shifted) count of sequence number wraparounds (s->cycles). That is:

```
extended_max = s->cycles + s->max_seq;
expected = extended_max - s->base_seq + 1;
```

The number of packets lost is defined to be the number of packets expected less the number of packets actually received:

```
lost = expected - s->received;
```

Since this signed number is carried in 24 bits, it should be clamped at 0x7fffff for positive loss or 0x800000 for negative loss rather than wrapping around. The fraction of packets lost during the last reporting interval (since the previous SR or RR packet was sent) is calculated from differences in the expected and received packet counts across the interval, where expected_prior and received_prior are the values saved when the previous reception report was generated:

```
expected_interval = expected - s->expected_prior;
s->expected_prior = expected;
received_interval = s->received - s->received_prior;
s->received_prior = s->received;
lost_interval = expected_interval - received_interval;
if (expected_interval==0 || lost_interval<=0) fraction = 0;
else fraction = (lost_interval << 8) / expected_interval;
```

The expected interval is calculated in the following way.

1. If the number of senders is less than or equal to 25% of the membership (members), the

interval depends on whether the participant is a sender or not (based on the value of we_sent). If the participant is a sender (we_sent true), the constant C is set to the average RTCP packet size (avg_rtcp_size) divided by 25% of the RTCP bandwidth (rtcp_bw), and the constant n is set to the number of senders. If we_sent is not true, the constant C is set to the average RTCP packet size divided by 75% of the RTCP bandwidth. The constant n is set to the number of receivers (members - senders). If the number of senders is greater than 25%, senders and receivers are treated together. The constant C is set to the average RTCP packet size divided by the total RTCP bandwidth and n is set to the total number of members.

2. If the participant has not yet sent an RTCP packet (the variable initial is true), the constant T_{min} is set to 2.5 seconds; otherwise it is set to 5 seconds.

3. The deterministic calculated interval T_d is set to $\max\{T_{min}, n*C\}$.

4. The calculated interval T is set to a number uniformly distributed between 0.5 and 1.5 times the deterministic calculated interval.

5. The resulting value of T is divided by $e^{-3/2}$=1.21828 to compensate for the fact that the timer reconsideration algorithm converges to a value of the RTCP bandwidth below the intended average.

This procedure results in an interval which is random, but which, on average, gives at least 25% of the RTCP bandwidth to senders and the rest to receivers. If the senders constitute more than one quarter of the membership, this procedure splits the bandwidth equally among all participants, on average.

5.2.2 Buffer management methods

We choose prioritized-RED and layered drop as the buffer management method. RED, as introduced in Chapter 2, is an efficient algorithm on managing routers buffers in public

network. However, it is not suitable for multimedia data transmission because it drops packets randomly without differentiating the importance of frames. In our implementation, we mark packets of different frames as different priority and discard the low priority packets first when necessary. The threshold for starting dropping and the dropping probability is adjusted by congestion control rules.

5.2.3 Packet scheduling methods

Three scheduling methods are implemented in the expert server. They are round robin (RR), weighted round robin (WRR) ([59]), and priority queuing. In the expert system, the weight for WRR scheduler is set during session setup stage and maintained on-the-fly by the QoS management rules and congestion control rules. The priority queuing method divides sessions into a premier and a normal group. A certain amount of bandwidth is reserved for sessions in the premier group while normal sessions receive only a best effort service.

It should be noted that the expert system decides more than merely selecting a scheduling algorithm for sessions. It creates smart combinations of those algorithms to make the transmission efficient. For example we can serve the premier sessions in an RR way for fairness while the normal sessions are served in a WRR way. Furthermore, the weight of each session could be changed depending on the availability of bandwidth and the historical performance of the session.

5.2.4 Rate control algorithms

We select four rate control methods for our case study. They are DLQ ([60]), TFRC ([61],[62]), EMKC ([63],[64],[65],[66]), basic AIMD ([67]). The brief comparison of these methods is given in Table 5-5. The notations are listed below the table.

TFRC and AIMD are window based congestion control while DLQ and EMKC are rate based. TFRC performs satisfactorily under most situations. AIMD is suitable for a network that occasionally encounters sudden parameter change. It is too slow for wired high throughput media streaming. DLQ is a client oriented rate control method we designed (Chapter 6) that can maintain high client buffer occupancy and reduce the jitter problem, but it does not consider the intermediate network conditions. EMKC obtains the overall system optimality by maximizing individual resource utility. The drawback is that it is delay sensitive and not always stable under heterogeneous environments.

	Response Function	Advantages	Disadvantages
TFRC	$\frac{s}{RTT\sqrt{2p/3}+3RTO\sqrt{3p/8}p(1+32p^2)}$	Satisfactory performance for most conditions	Slow start after congestion
DLQ	$u_i^* = -\frac{T_s p_{i+1}}{1+T_s^2 p_{i+1}} x_{i-1} + \frac{T_s^2 p_{i+1}}{2(1+T_s^2 p_{i+1})} l_{i-1} - T_s b_i$	Client oriented. Avoid jitter. Easy implementation	Not consider intermediate network conditions.
EMKC	$R_{i+1} = R_i + \alpha - \beta p_i R_i$	Maximize individual resource utility	Delay sensitive. Not always stable.
AIMD	$I: W_{n+1} = W_n + \frac{W_n}{f(W_n)}$ $D: W_{n+1} = W_n - W_n g(W_n)$	Suitable for N/W with frequently changed parameters.	Too slow for Wired high-throughput media applications.

Table 5-5 Comparison of congestion control methods used in expert server system

p: loss rate	S or s: packet size
C: bandwidth capacity	R: sending rate
W: window size	RTT: round trip time
RTO: request timed out	Subscript n or i: discrete sample points

Notice that most provided congestion control methods in Table 5-5 take loss rate p as the

input, especially for EMKC, who take p as the only parameter of the environment. Actually, loss rate p is the most important parameter that describes the congestion situation. Higher loss rate is usually caused by increasing congestions in the intermediate networks. However, the loss rate carried by feedback packets sometimes are not precisely calculated due to measurement noise and are often delayed by intermediate network. Therefore we add the Kalman Filter ([69]) in the expert system to predict the p value. We move the design of Kalman Filter into appendix A and only give out the simulation result for EMKC in Figure 5-4 since the main purpose of our work is not improve a specific congestion control algorithm. The EMKC with Kalman Filter is called EMKC_KF. From the Figure 5-4, EMKC_KF is more stable than EMKC under violated noise and delay. In the implementation, we use EMKC_KF instead of EMKC.

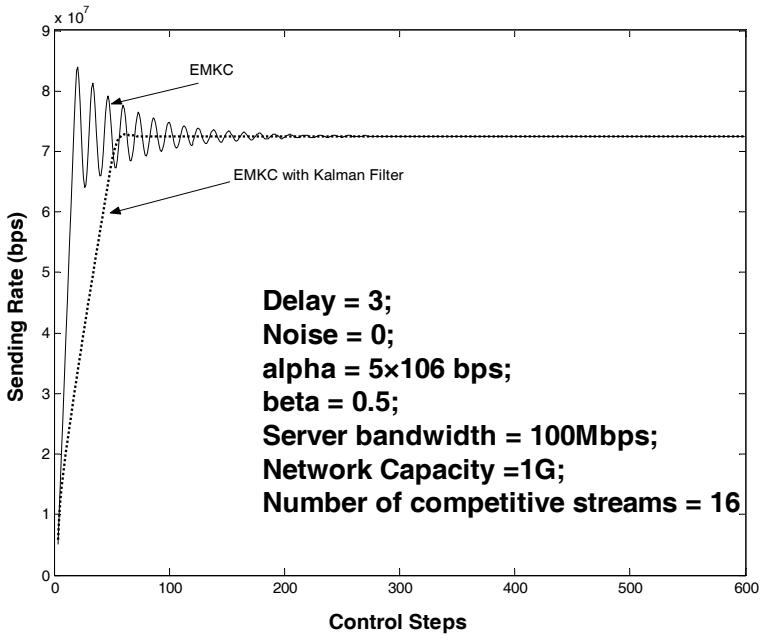


Figure 5-4 Performance of EMKC_KF

These algorithms are used for congestion avoidance. To take the advantage of them and to avoid their shortcomings, the expert system selects them according to the network condition

it encountered and the client information availability. Under the usual wired network conditions, TFRC achieves satisfactory performance. When *p* increases to a higher value, TFRC generates a very small sending rate which may cause the interrupt of media streaming. Now the EMKC_KF could be switch on and takes over the control. If client buffer occupancy could be provided during streaming, DLQ is turned on for an optimal sending procedure. In the case of wireless application where loss problem is seriously, AIMD with retransmission mechanism is a suitable method to guarantee the quality of transmission. In our implementation, only *I* frame packets need ACKs and will be retransmitted. Finally, we should note that the selection and switching among these algorithms are decided from the statistics for a certain time, not by any instant value. Furthermore, these algorithms are applied to only streaming sessions for congestion avoidance. If congestion happened due to background traffic or network/ client jam, expert control has to turn on other mechanisms to handle it. The detail congestion control steps and implementations will be introduced in sub-section 5.3.5.

5.3 Results and Discussions

For streaming applications, the main criteria of performance are high CSR (Connection Success Rate), fast session setup, smooth throughput, small delay, low jitter, and acceptable control overhead. We compare these parameters for QTSS with and without expert control in front of different problems in this section.

5.3.1 Effective Admission Control and Load Balance

Usually, admission control is implemented in QoS-enabled networks and it requires the cooperation of routers and servers. Our target is to enhance the performance of streaming

server, not to improve the routers, so the experiment in this sub-section is to test the expert system can distribute traffic reasonably among servers and to test the startup time when system load increases. Admission criteria are set based on the analysis of available system capability and prediction of the new session traffic requirements. The test-bed configuration is shown in Figure 5-5. Four computers (IBM T40, DELL Precision T5400, two Shuttle XPCs) serve as the streaming server, denoted as S1 to S4. Two DELL Optiplex755 with 9 virtual machines in each and three DELL PrecisionT5400 with 19 virtual machines in each serve as clients. All together there are 80 clients. Four servers are connected to each other through a LAN, and they are allocated with different parameters using Linux TC network simulator (Table 5-6). Eighty nodes send movie requests to the target server with predefined distribution in Table 5-6, one session added every 10 seconds. The delay between S1 and S2, S2 and S3, S3 and S4, are 100ms, 20ms, 50ms respectively.

When a joining new session exceeds the capacity of the server, QTSS still admits the session and degrades all current sessions. In this experiment, we tried four scenarios. For example in the first scenario, 100% requests are sent to server 1. In the second scenario, 50% clients send request to server 1 and the rest 50% send to server 2. We conducted these four scenarios with basic QTSS and QTSS with expert control, the sessions on each server are recorded after stabilization.

Figure 5-6 shows the stable traffic distributions of QTSS with and without expert control for four scenarios. Note that the total number of sessions on all servers may be smaller than the total number of clients requested. For example in the left upper picture in Figure 5-6, altogether only 35 sessions were recorded at server 1 with the basic QTSS (blue column), although all 80 clients issued the request. This is because the adding of the 36^{th} session

caused the previous streams to be discontinuous and thus unable to be watched. So we take the maximum number of sessions supportable by basic QTSS under this scenario to be 35.

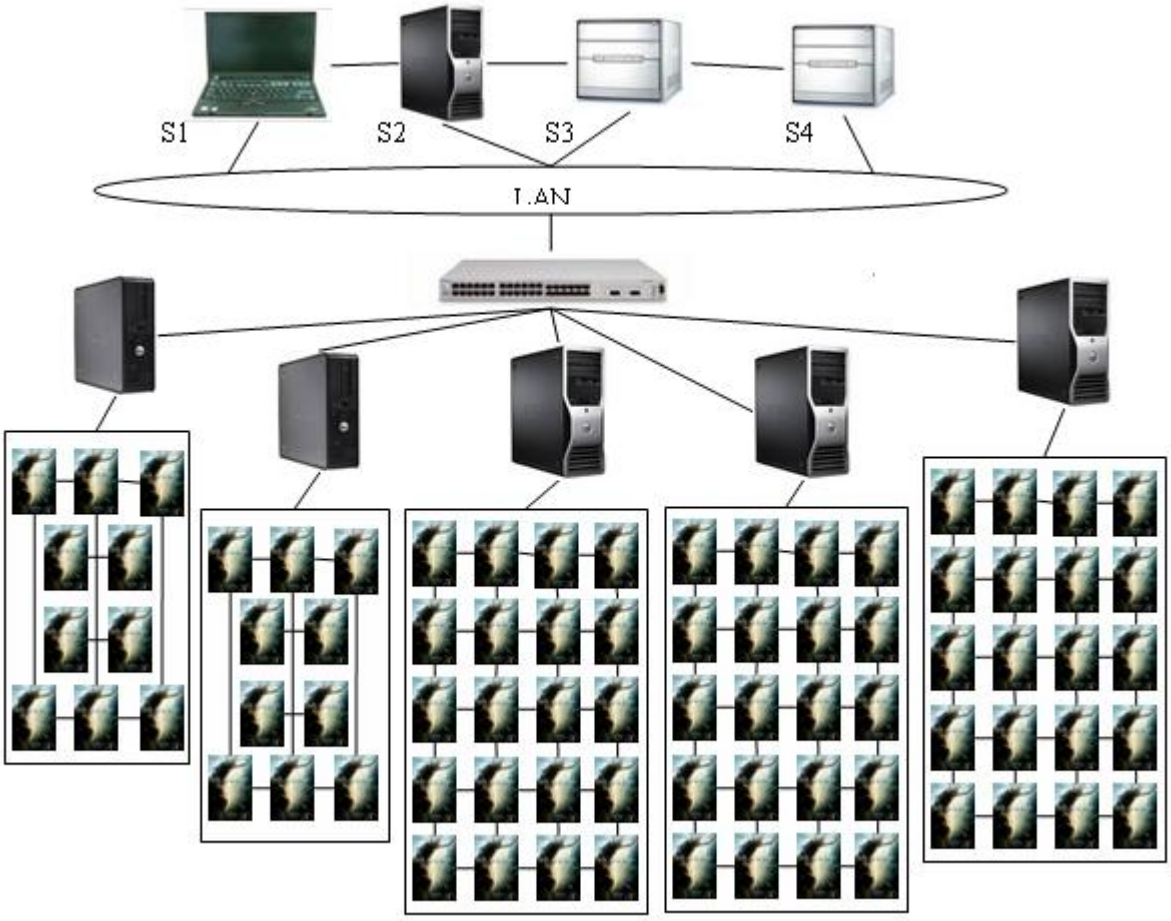


Figure 5-5 Test-bed configuration for admission control and load balance

Servers	Bandwidth	RTT	Loss Rate
S1	*100M*	*200ms*	*0.1*
S2	*100M*	*40ms*	*0.01*
S3	*50M*	*100ms*	*0.1*
S4	*50M*	*20ms*	*0.001*

Scenarios	S1	S2	S3	S4
Scenario1	*100%*	*0*	*0*	*0*
Scenario2	*50%*	*50%*	*0*	*0*
Scenario3	*40%*	*20%*	*40%*	*0*
Scenario4	*25%*	*25%*	*25%*	*25%*

Table 5-6 Server configurations and load distribution

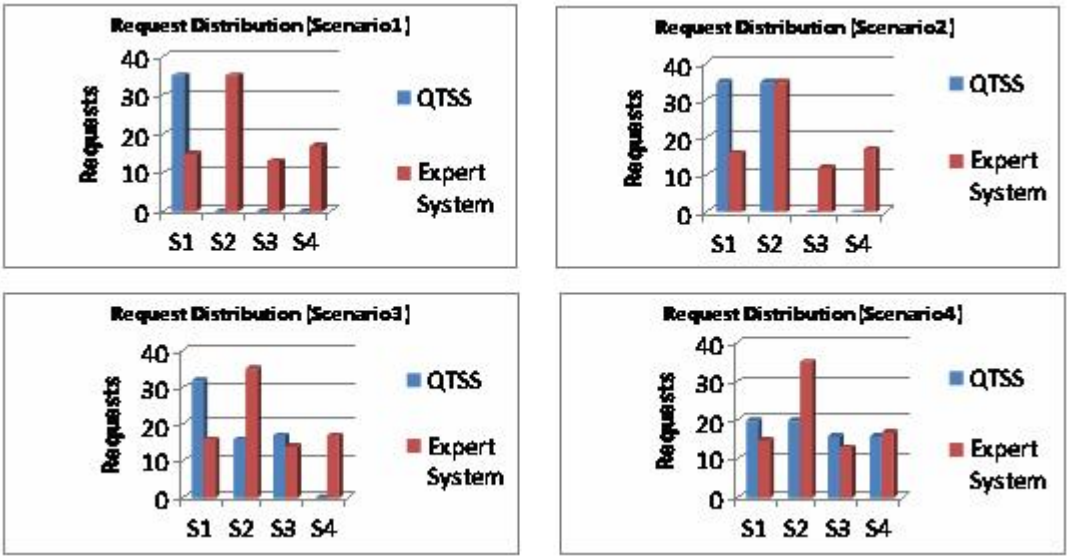


Figure 5-6 Load balance of QTSS with / without expert control

The requests reaching the QTSS server are processed locally and served in a best effort manner. The basic QTSS has no information about other servers in the same cluster, so the four servers are independent of each other, although they are connected. In Scenario 1, all 80 requests are issued to S1 and QTSS can only support around 35 sessions. Its CSR is 43.75%. In Scenario 2, requests are divided into half-half and issued separately to S1 and S2. Because S1 and S2 have the same bandwidth capacity, they accepted nearly the same number of requests. As a result, totally 70 requests are successful, CSR is 87.5%. In the third scenario, 32 requests went to S1, 16 requests went to S2, and 32 requests went to S3. S1 and S2 have enough capacity to accept all coming requests, but S3 could only support around 16 sessions, making the altogether CSR to be 80%. The rest 16 sessions have to wait until current sessions finished, although S1 and S2 have spare capacity to support them. Even in Scenario 4 where requests are evenly distributed, the CSR of QTSS reaches up to 90%, still 8 requests sent to S3 and S4 can not get the service.

On the contrary, no matter which server the requests were initially sent, QTSS with expert control will reasonably distribute them among server according to the server and the

intermediate link parameters. The under-layer execution is like this. Servers, S1 to S4, exchange messages about their available bandwidth, load level, etc. The RTT between two servers are tested with the monitor procedure, which runs every 100ms. Loss rate is initially set to zero and updated with RTCP client report after session established. In our rule base, the traffic distribution rules will first estimate the available bandwidth of all servers from their current load level. For those servers that have enough capacity to support the new request, it calculates a weighted sum for the link delay and loss rate. The lower the value is; the higher chance the corresponding server is selected. After deciding the server, new request will be forward to that server, until an accept ACK message is received. If a Reject message is received other than ACK, it means the target server is busy and cannot accept the new request. For example the server has other background FTP traffic running. In such a case, the expert control will select the second choice and try again. Meanwhile it updates the patient user with messages. If a new request is rejected four times by remote servers and the local server that the request initially sent cannot support it, a reject decision will be made and sent to the user. From the experimental results in Figure 5-6, QTSS with expert control has approximately identical load distribution for four scenarios, and their CSR are all 100%.

Another important parameter to scale the server performance is startup delay. Strictly, the startup delay is defined as the time between pressing the START button and the beginning of the movie play. Our startup time is calculated by capturing the first RTSP request packet out and the first RTP data packet returned on each client. It is shorter than the real startup delay because it does not count the buffering time at client device before playing. The startup delays of these scenarios are shown in figure 5-7. We want to use it to investigate the overhead brought by expert server when performing admission control and load balance.

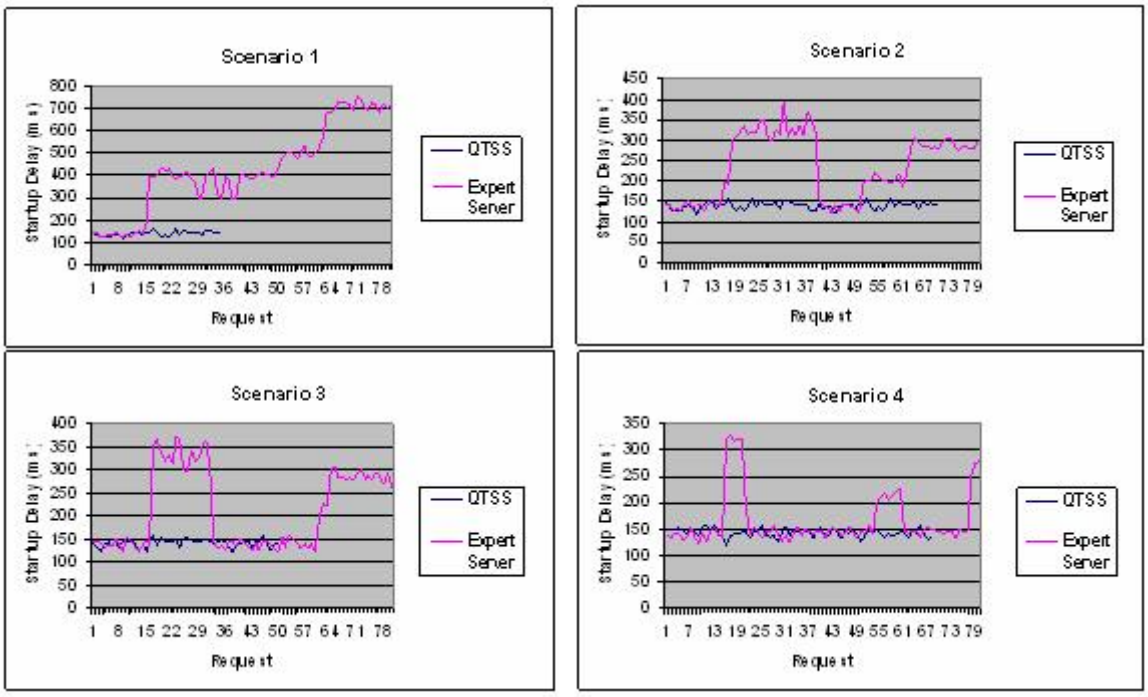


Figure 5-7 Startup delay during load balancing

As we illustrated in previous paragraphs, basic QTSS processes requests locally. When the new coming request exceeds the capacity of the server, it will not be forwarded to the other server that has enough capacity to handle it. Therefore in the four figures in Figure 5-7, the startup delay under basic QTSS control (blue line) is quite stable at 145ms, and most time it is lower than or overlap the one (red line) with expert control. The blue lines are not continuous to the end of the x-axis because we only show the startup time for accepted sessions. Those requests that cannot be supported by the server are not counted into calculation.

For QTSS with expert control, a new request may be transferred to another server when the current server capacity is not the best choice. When such a transfer happens, the join procedure of a new session is lengthened, and the startup time becomes longer. That is the reason for the curves shooting up at certain times in four pictures with expert server control. The more hops required to transfer the request, the longer time it takes before starting the

streaming. In our experiments, the requests are issued from S1 to S4 sequentially. If a request needs to be transferred to S2, S1 forwards the request to S2 and wait for ACK or Reject message. If S2 cannot handle the request, S1 has to ask S3 or S4 as a second choice. In these cases more time is taken by such kind of re-forward processes. Reflected from results, the startup time may rise to several different levels.

Before we explain the curve of each picture, we should notice that the average delay between S1 S2, S2 S3, S3 S4 is 100ms, 20ms, 50ms respectively (Table 5-6). In scenario 1, the S1 accepted 15 requests. From request 16, S2 is found to be a better choice to serve the session, so it forwarded the following requests to S2 until S2 saturated at 35 sessions on it. Then the S1 forward the request to the farer S3 for another 13 requests and the most far away S4 17 sessions. In the second scenario, S1 accepted 15 sessions, and then forwarded the rest 25 requests to S2. S2 accepted these requests. After this, S2 has 40 new requests coming but it can only support 10 more on itself. Therefore it forwarded 13 requests to S3 and 17 requests to S4. Because the delay between S2 S3 and S2 S4 is much lower than those for S1, the startup delays for these forwards took less than half the time as previous forwards from S1 to S3 and S4. Similarly in scenario 3, S1 forwarded from request 16 to 32 to S2; S2 accepted the following 16 requests issued to it; S3 accepted 13 requests on its own and forwarded 2 requests to S2 and 17 requests to S4. In scenario 4, S1 forwarded 5 requests to S2; S2 accepted these forwarded requests and all 20 requests issued to it. S3 accepted 13 requests originally issued to it and forwarded 7 to S2. Similarly S4 accepted 17 requests and forwarded 3 to S2. Since each expert server knows the capacity of others and it runs the same rule base as others, they should make the same decision for traffic distribution and load balance. The maximum delay brought by expert control in the experiments is 582ms, which

is acceptable as the tradeoff for the performance enhancement.

5.3.2 Playback Scheduling

Most commercial VoD systems need to insert advertisements to make money. These Ads could be inserted randomly in between the movie. However, some advertisements are only suitable to a specific group of people, and they may need to compete with other content providers for limited time slots. In this case, the expert control is a good solution to automatically solve the schedule problem without interference of maintenance engineers, especially for close system VoD. For example, JCDecaux company did a market search ([70]), finding that advertisement should be presented according to the clients' nationalities, genders and ages. We use their research results and add the race and occupation as optional input parameters to decide the advertisement selection, sequence and frequency. We did not provide the streaming results for advertisement schedule because it is difficult to present the results in a static way. Here are the example profiles provided to the expert server and an example schedule finally generated.

Available Information	Representation
Client Profile	Client Demographics File
Content Profile	Content Description Files
Advertiser Profile	Ad Description Files

Table 5-7 Profiles for playback schedule

Name	Gender	Age	Nationality	Race	Occupancy
HIEW KAR HON	Male	45	Singapore	Chinese	Manager
YEO LEE HUA	Female	28	Singapore	Chinese	Accountant
RUSMA IBRAHIM	Male	32	India	India	Engineer
…	…	…	…	…	…

Table 5-8 Client profile example

The expert server will analyze the client profile and generate a summary file like:

Gender Count File

Male	780	46%
Female	700	42%
Children	200	12%

Nationality Count File

Singaporean	1080	64%
Malaysian	428	26%
Indonesian	172	10%

Age Count File

Age 0~12	80	5%
Age 12 – 18	120	7%
Age 18 – 26	320	19%
Age 26 – 45	960	57%
Age 45 – 60	150	9%
Age above 60	50	3%

Occupation Count File

Finance	280	17%
Engineer	668	40%
Academy	140	8%
Hospitality	320	19%
Tourism	70	4%
Management	202	12%

Table 5-9 Sample statistic file of subscribers

Movie/ Ad.	Language	Length	Preferred Ages	Gender	Occupation
M_Spiderman	English	90 mins	12 ~ 45	All	All
M_Enchanted	English	75 mins	12 ~ 45	Female	All
M_TheMyth	Chinese	100 mins	18 ~ 60	All	All
A_ToughBook	English	30 sec	Above 18	All	Management
A_MacDonald	English/Chinese	10 sec	All	All	All
…	…		…	…	…

Table 5-10 Example of a content profile

	Reach	**Frequency**
Adults	19.9	64%
Age 15-24	20.9	63%
Age 25-34	19.9	64%
Age 35-44	22.4	62%
Age 45-54	26.2	63%
Age 55+	10.9	66%
Men	20.3	66%
Women	19.3	61%

Table 5-11 Example of an advertiser profile

The advertiser profile is provided by advertisement companies, specifying the percentage of age group they want to reach and the frequency these Ads are expected to show. Our rules are designed to set the weight for each advertisement according to above profiles. Finally the server will calculate the sum weight. The top ones will be selected and inserted into the

demanded movie in sequence. Here is a sample output playback schedule table and several snapshots of different schedules.

Playback Schedule (Sample)		
1	Welcome Video	30 secs
2	ToughBook-2008-June	30 secs
3	McDonald-2008-June	10 secs
4	Spiderman3 Part 1/5	20 mins
5	Ad12-2008-June	10 secs
6	Ad14-2008-June	30 secs
7	Spiderman3 Part 2/5	20 mins
8	Cosmetic Ads.	20 secs
9	MariFrance Bodyline Ads.	40 secs
10	Spiderman3 Part 3/5	20 mins
…	……	…

Table 5-12 Sample output playback schedule

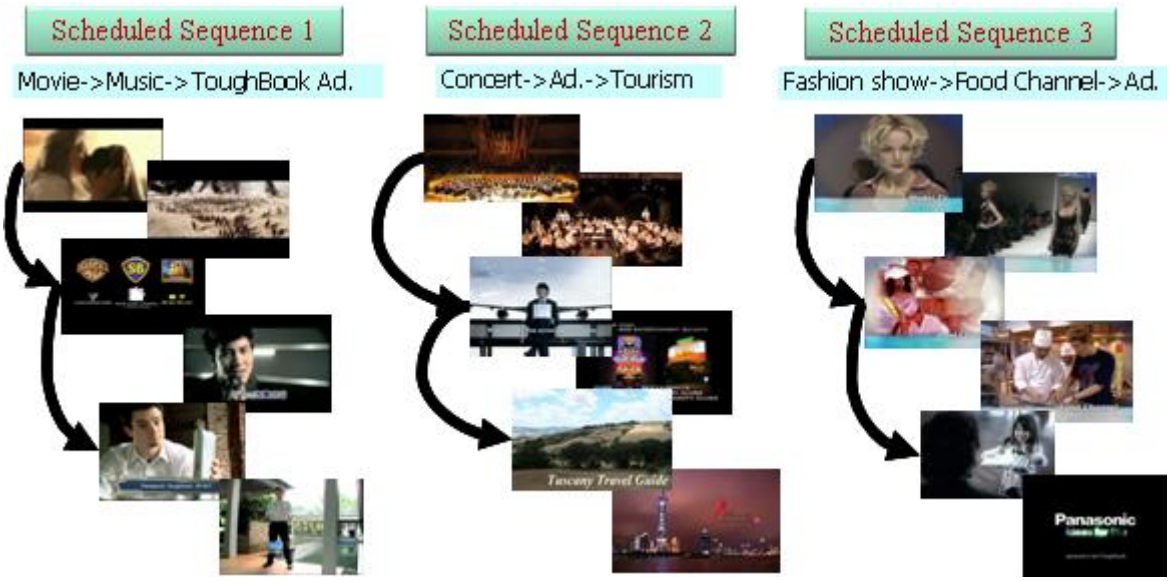


Figure 5-8 Playback schedule examples

The playback schedule could also be applied in another way that scheduled based on parameters like device screen size, the signal strength, the device battery life, etc. This method has been used in some commercial streaming servers. They ask users to select the

link speed, whether they are using ADSL, modem or LAN. Then the server chooses the size of movie file to be transmitted. The proposed expert server extended the scope of such scheduling to broader areas, and realized the schedule in a completely automatic way.

5.3.3 HD Streaming Rate Control

These HD streams, required high sending rate, will greatly impact the network traffic load, so it is important to perform effective rate control for these HD streaming. In basic QTSS, they have a rate control mechanism called reliable-UDP. It starts sending from a very low bit rate; then adjusts the subsequent rates based on a service-feedback byte from RTCP receiver report. It is sufficient for low bit-rate streams like stream-1(movie 320) and stream-2 (movie 480p), but simple for HD streams like stream-3 (movie 720p), which has high throughput fluctuations. Thus, we added the algorithms in Table 5-5 to avoid congestion for these streams. The major congestion avoidance rules are written like this. If the router could provide feedback regarding the previous loss, EMKC is applied to share and maximize bandwidth utilization among all streams based on the subscription fee paid by the client. For client who can send its information to the server, TFRC is used to compete with other TCP traffic fairly but friendly on routers along the route. DLQ is also turned on to cooperate with TFRC for smooth throughput and to guarantee to overflow or underflow of client buffer. If unstable wireless network is the physical layer, the AIMD is used, and only I frames will be scheduled for retransmission if loss happened. Here, DLQ is turned on only when client buffer occupancy could be provided. It is also used as a reference to avoid client buffer overflow and underflow. Figure 5-9 compares the throughput with and without expert rate control for stream-3 (movie 720p).

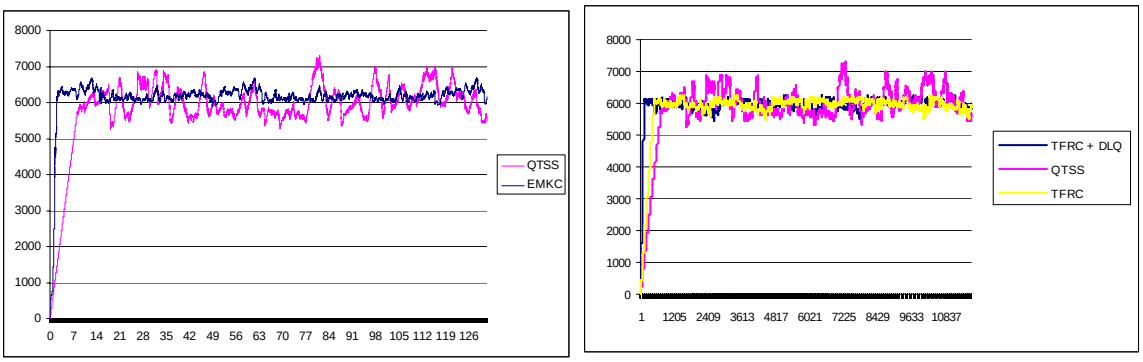


Figure 5-9 Client side throughputs with and without expert control

The left picture of Figure 5-9 shows the throughput under basic QTSS control (purple) or with EMKC control. The throughput with basic QTSS control fluctuates within a range of 2Mbps (5.2Mbps to 7.2Mbps) while the throughput with EMKC changes within 0.7Mbps (6Mbps to 6.7Mbps), which is much smaller. The time consumed before stabilization with EMKC control (around 2s) is only a quarter as that with QTSS control (around 8s). The right picture in this figure compares the throughput under basic QTSS or pure or TFRC and DLQ control. It illustrates that the smoothness of stream increases from QTSS control, TFRC control, and combined control of TFRC and DLQ, and the raising time before stabilization increases reversely. The phenomenon happened because the basic QTSS and our expert control have different abilities to adapt their sending rate under a fluctuating environment. Basic QTSS calculates sending rate only according to satisfactory byte in RTCP receiver report, while the expert system takes into consideration the traffic demand, the available network bandwidth, the previous sending rate, and the client situation. When loss rate or RTT changes and influent the QuickTime playback quality, QTSS hurriedly changes its sending rate to compensate the changes. The expert system, on the other hand, updates the new p value and the demand data size of client and calculates whether the current sending rate can still meet the requirement of client under new environment settings. If the requirement can be

met, the current sending rate is carried on. If necessary, the expert system will switch on the QoS mechanism to change the TOS simultaneously, in stead of changing the sending rate, in response to the changes of loss rate. In this way, the client can receive a smoother media flow with less buffer overflow or underflow problems. Since the client side player is not open source software, we use Wireshark to capture the incoming data size and estimate the buffer occupancy by

$$S(t) = \begin{cases} S_d - S_h - \int_{t=0}^{T} r_{pb}\,dt \\ 0 \end{cases}$$

$S(t)$ is the client buffer occupancy at a certain time t

S_d is the total size of data received until time T

S_h is the total size of packet header received until time T

r_{pb} is the playback rate at time t

Using Stream-1 (movie 320), the network delay is set to 20ms with 5ms variation, and the loss rate is 0.1%. The total client side playback-buffer size is set to 30MB. Using DLQ method, the client buffer utilization with and without expert control is shown in Figure 5-10.

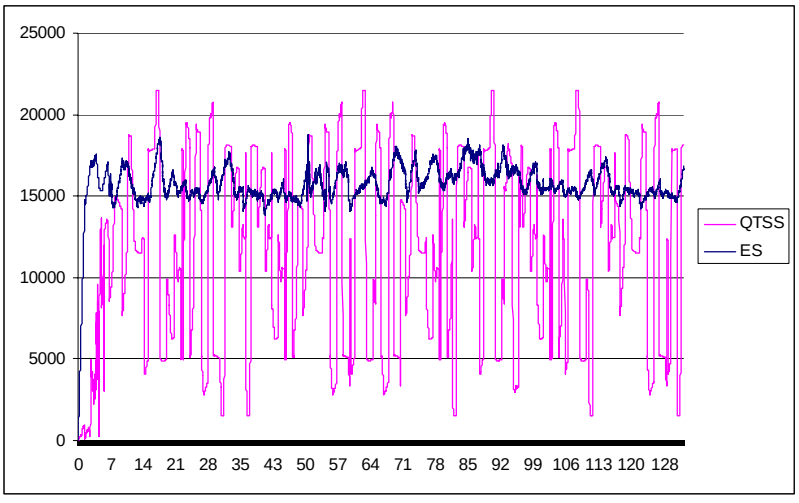


Figure 5-10 Client buffer utilization of QTSS with/without expert control

To prevent overflow, the expert server used only around 60% of the maximum client buffer allocated for media receiving. During the transmission, it traced the usage of client buffer from client feedback packets. It is obvious that the client buffer occupancy fluctuated randomly with the playback rate under basic QTSS control, but it became quite smooth when expert control was added. This is because the basic QTSS does not consider the client buffer status when deciding the sending rate. It only refers to one RTCP byte about client satisfaction. As long as the playback is continuous, it will not adjust its sending rate. On the other hand, expert system can detect client buffer overflow or underflow problems from low level client feedback. At the same time, it checks the movie characteristics and predicts the data consumption at client side. Within the bandwidth limitation that the network could provide, an optimal sending rate is decided to guarantee the stability of client buffer occupancy without overflow. In this test, the expert control used 60% allocated buffer to calculate its optimal sending rate. Actually, if underflow happens frequently, the expert system may increase the client buffer size parameter to 70% or 80% of total allocated size. If overflow happens frequently, the client buffer parameter is decrease to a percentage less than 60%. Such a kind of dynamic adjustment performed the same way as an expert controlling the server with its empirical knowledge. It guarantees the server's efficiency given any client device. The small ripple of client buffer may seem to be not a crucial performance enhancement, but it will provide enough space for sudden bandwidth changes, especially during streaming handover that will be shown in sub-section 5.3.4.

5.3.4 Streaming Handover

In literature, researchers are investigating seamless handover when the client switches among different networks during transmission (e.g. between WLAN and Cellular network [71]). In

our expert server, we will consider a more challenge scenario that the terminal devices are changed during playback and the expert control could perform online streaming handover. Consider the cases that the client device switches from the slower hand phone streaming to a faster network with plasma TV display, or vice versa. These kinds of handover require the server support for signaling and movie transfer. In our test, we use the IBM laptop as the original terminal user to simulate the hand phone streaming. When a handover signaling is issued to the server, the playing movie will be automatically transferred to another terminal with a large plasma TV. An appropriate resolution of movie will be selected for the new terminal. The test bed is set as in figure 5-11. We use a shuttle XPC as the switch. The delay and loss rate are on-the-spot tested value.

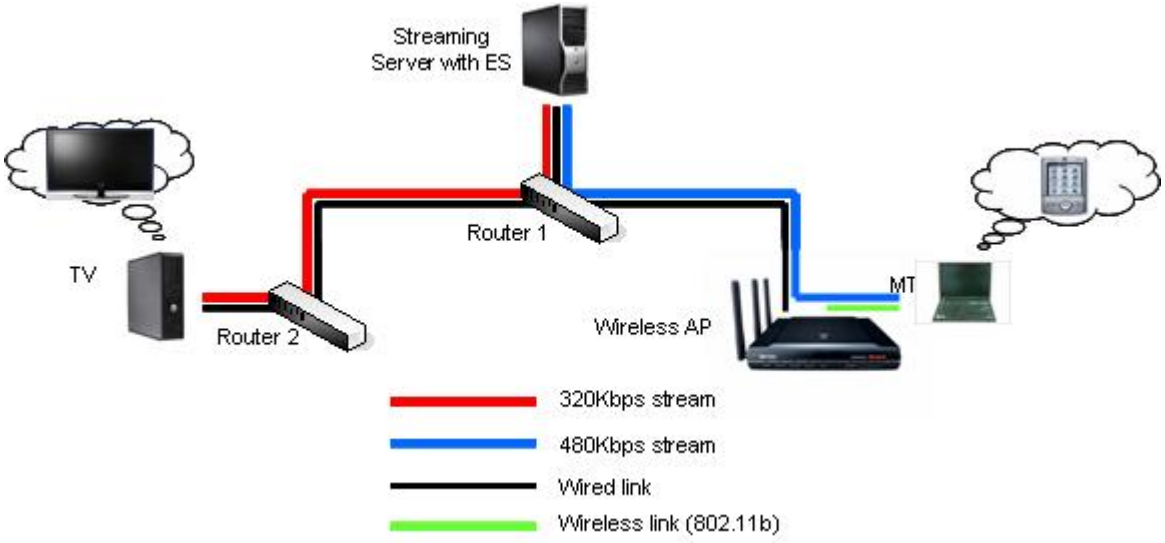


Figure 5-11 Test bed configuration of streaming handover experiments

Because the handover from hand phone to plasma TV or handover from plasma TV to hand phone are invertible process, we only illustrate one case and provide the results for both cases. The results of experiments are shown from Figure 5-12 to Figure 5-14.

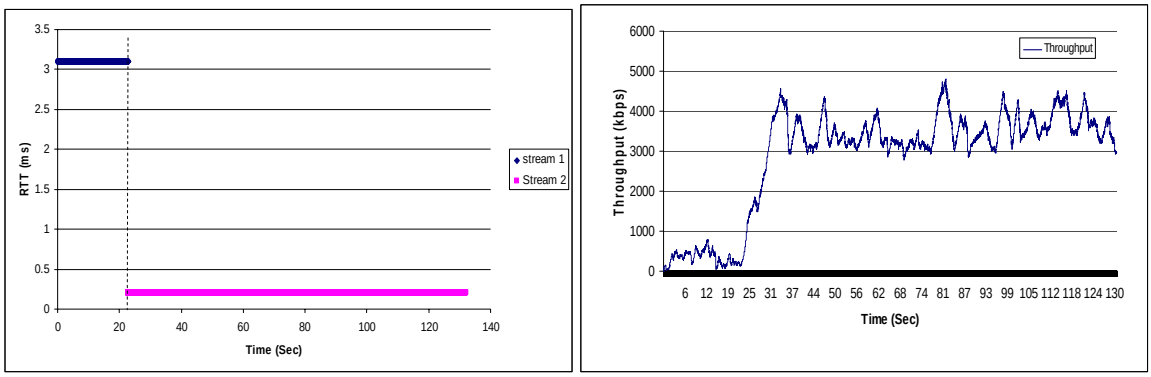


Figure 5-12 RTT and Throughput during streaming handover (wireless → wire)

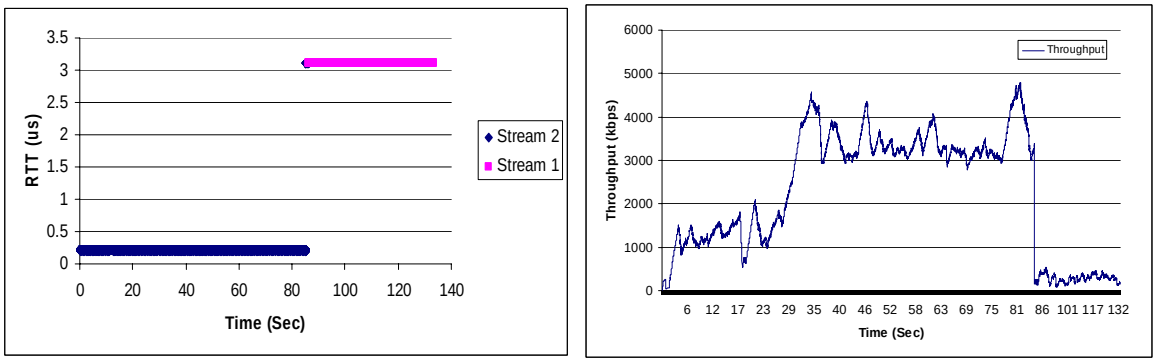


Figure 5-13 RTT and Throughput during streaming handover (wire → wireless)

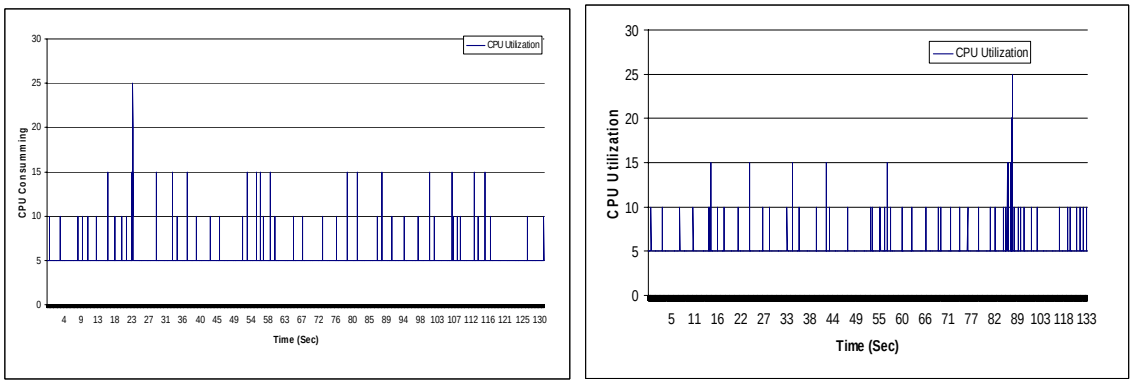


Figure 5-14 CPU-utilization during streaming handover

(Left: wireless → wire; Right: wire→wireless)

In Figure 5-12 left picture, the server received an RTSP-like handover request at 21s indicating the target device IP, frame number, and desired resolution for handover. Then the server started handshaking with the new device and test its RTT value. The overall handover took about 5 seconds to finish, including the new startup buffer time. The sequence of three

party handshaking is shown below.

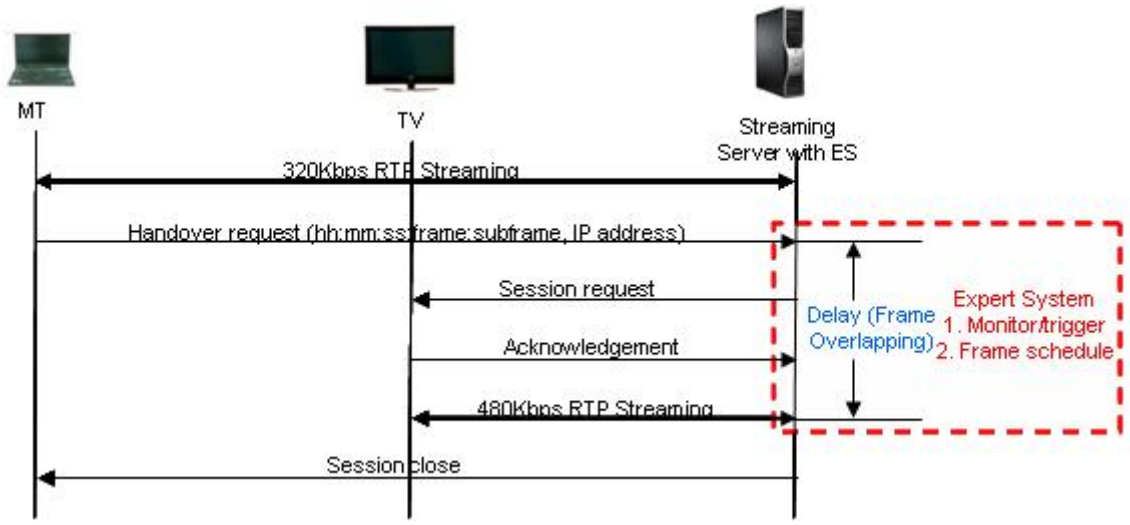


Figure 5-15 Handover signaling from wireless to wire devices

From Figure 5-12 right picture, when the handover happened at 21s, the server continued the original streaming to the wireless device; meanwhile it started to handshake and to buffer the required data at plasma TV. When the buffer procedure completed at 26s, the original session is shut down and the new high-resolution session began to play. The throughput is changed from stream-1 (movie 320) to stream-2 (movie 480p). During the handover, the server CPU utilization increased twice as usual. The overhead is temporarily increased to 25%.

There are other ways to accomplish the handover, for example trans-coding introduced in [71]. From this paper, the handover handled by trans-coding will take 50 seconds to finish. It is too long to wait, so we did not deploy it in our implementation. Our approach is to store several files with different resolution. If handover is needed, a new resolution file is selected and the file pointer is relocated at the same frame as in the original file.

5.3.5 Congestion Control

The congestion control in expert server could be divided into several steps: congestion avoidance, congestion mitigation, congestion response, and session re-distribution.

Congestion avoidance couples admission control and RTCP rate control mechanism to cautiously avoid congestion. However, if the streaming do becomes congested, the expert server congestion control needs to start prioritize-RED to alleviate it. If the severe situation continues, the expert control will react according to the congested location. That is, if it is client side or intermediate network congested, servers could only dramatically reduce the traffic demand to cooperate on mitigating the problem, but it cannot solve the congestion problem in itself. On the contrary, if congestion happens at server side, the server could take full responsibility and get itself out from it by re-distribute the session. Thus in this section, we are going to conduct our experiments for these congestion control steps.

Step 1: Congestion Avoidance

The first step is to use congestion control algorithms to avoid transmission jam under normal situation. We have introduced the reliable-UDP in basic QTSS to control the sending rate according to RTCP feedback. This mechanism is not needed when network load is light, so QTSS provides the choice to start this function or not with a byte specified in RTSP requests. This simple congestion control is good enough for small movie streams, for example stream-1 (movie320) or stream-2 (movie480), but as we introduced in sub-section 5.3.3, it has problems like long start up time and large throughput fluctuations for high definition movies. To make the control procedure smarter and more flexible, our congestion avoidance rules are implemented like this. When the loss rate is less than 1%, reliable-UDP is disabled. The rules will enable it only when the loss rate increases above 1%. For HD content, reliable-UDP is always disabled, regardless the loss rate. Instead, the HD streaming rate control mechanism introduced in sub-section 5.3.3 is applied.

Step 2: Congestion Mitigation

When transmission encounters a loss rate that is larger than 5% and the mean RTT is 1.5 times as before, it is very likely that the congestion is forming. To mitigate the network load, the expert server starts prioritized-RED to drop some unimportant frame packets. Meanwhile, the packet for I frame is marked to be the highest-priority TOS as QoS reference for intermediate network. Figure 5-16 illustrated the real network test for this case. This test is done between Singapore and Chicago. The client at Chicago sent requests to the server at Singapore. The intermediate network is public Internet. All tests were done with real network, where un-congested test was done at morning and congested test done at 8pm in Singapore. In the morning, the public Internet at server side is not congested while at 8 pm, the Internet is terribly congested. We provide this test to verify the real congestion mitigation ability of the expert server.

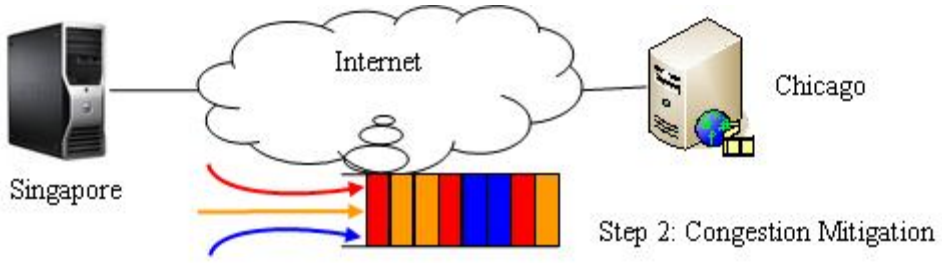


Figure 5-16 Test configurations for congestion mitigation

We chose the movie of Apple CEO Steve Jobs' keynote (Stream-4) for testing. This keynote was addressed for iPhone in Macworld Conference & Expo 2007. The traffic under different network conditions was analyzed using the WireShark Packet Analyzer. Figure 5-17 shows the video effects and throughputs under light-load network. In the figure, the left side picture is the snapshot of the movie with basic QTSS control and the right side picture with the expert system control. The two pictures have nearly identical resolution and playback speed. Both of them are clear and smooth.

Figure 5-17 Basic QTSS streaming video effects without/with Expert System

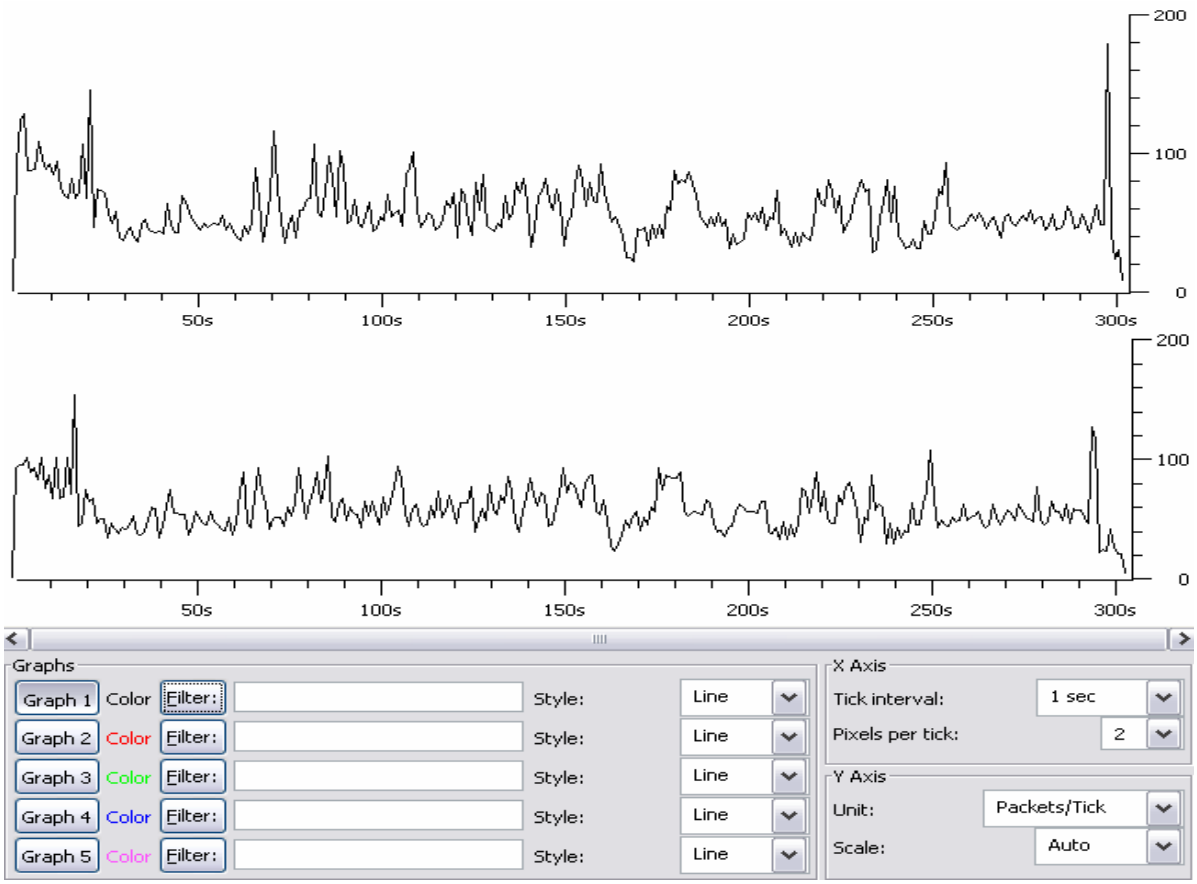


Figure 5-18 Basic QTSS throughputs without/with Expert System

If we examine their throughputs, as shown in figure 5-18, we will see that their traffic is also similar. The upper picture in figure 5-18 is the throughput using basic QTSS server while the lower picture is the one with expert system control. Both of them have an average

throughput of around 70 packets per second, which is approximately 0.85Mbps.

We now change to a slower network and add heavy background traffic to compete for the bandwidth. This traffic was other randomly generated streaming applications. The video effects and the throughputs of the two systems are shown in figure 5-19 and figure 5-20 respectively. Using basic QTSS, the streamed video under insufficient bandwidth suffered seriously from jitters. The movie is discontinuous with occasionally corrupted pictures, like the left side picture in figure 5-19. It is impossible to recognize what is really on the screen when Mr. Jobs introducing the iPhone. However, the video effect with expert control is more acceptable. It lowered the transmission speed and discarded some unimportant frames to save bandwidth. Although the playback is also slower than normal situation, it indeed provided a continuous stream with recognizable pictures.

Figure 5-19 QTSS streaming effects during congestion without/with Expert System

Throughput analysis is shown in figure 5-20. The upper picture is the throughput of QTSS with expert control and the lower picture is the one under basic QTSS. As introduced at the beginning of this section, the QTSS uses reliable-UDP as its congestion control mechanism. The reliable-UDP imitates the behavior of TCP streams and relies on feedback information to adjust the sending rate. Therefore the basic QTSS behaved like TCP flows when network

bandwidth is not sufficient. In figure 5-20 (lower picture), the reliable-UDP control always tries to raise its sending rate until receiving bad-performance feedback from the client. Then it suddenly resets its delivery speed to alleviate the congestion. Such kind of rate fluctuations happened along the transmission, which made the playback process discontinuous.

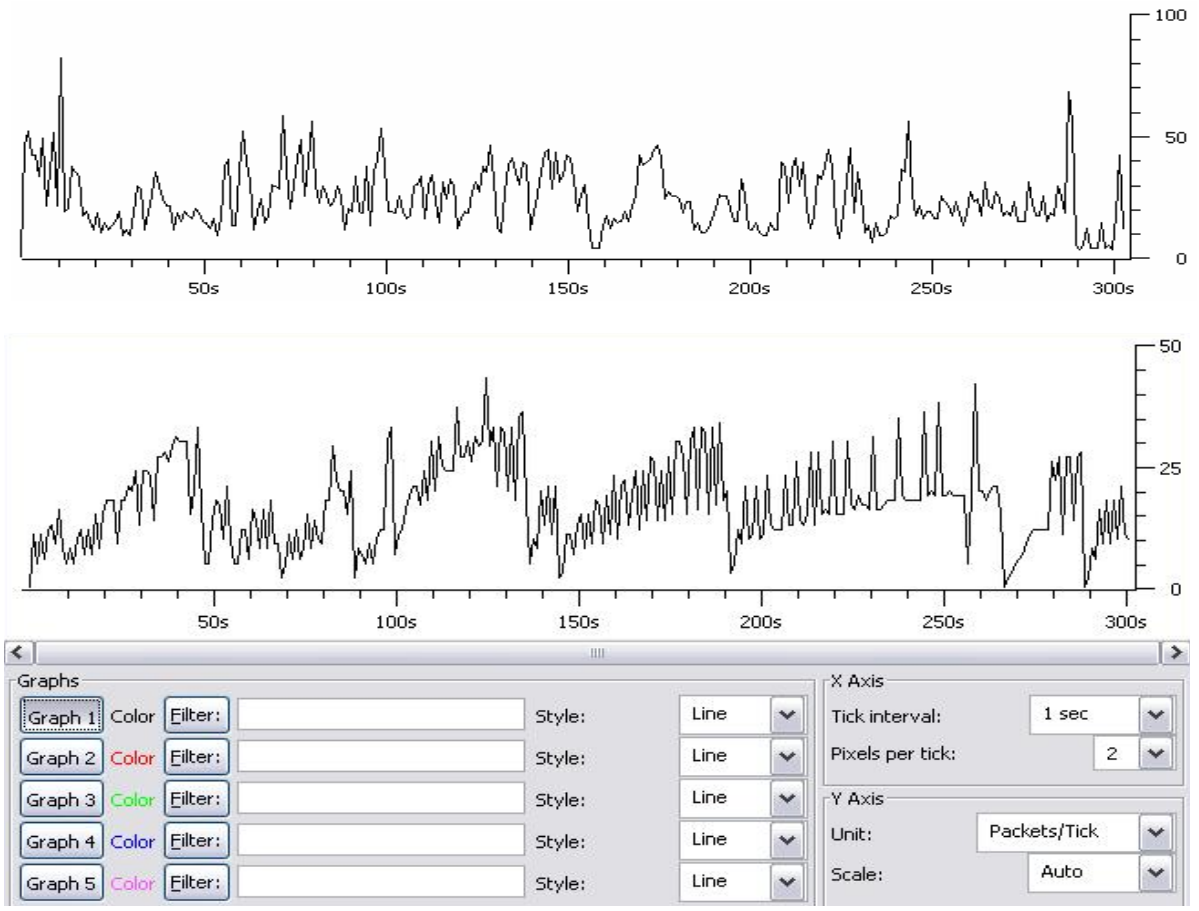


Figure 5-20 QTSS throughput under congestion (with/ without Expert Control)

The expert control detected the insufficiency of available bandwidth and lowered its sending rate at the beginning of transmission. It switched on movie quality filters for outgoing media packets, sending only the base layer. This will cause the reconstructed picture blurred (right picture in figure 5-19), yet it is more acceptable compared to have periodically corrupted pictures during the playback. The drop threshold and drop percentage were decided by some heuristic rules, which shall be written and adjusted by experienced media expert.

Step 3: Congestion Response

If congestion happened, and the expert server load is moderate, it means the congestion happened at intermediate network or client side. In this case, only drop frames are not enough to help. The expert server would intensively decrease its traffic demand by delivering a lower resolution file of the same movie. The client side congestion control test could be illustrated with Figure 5-21.

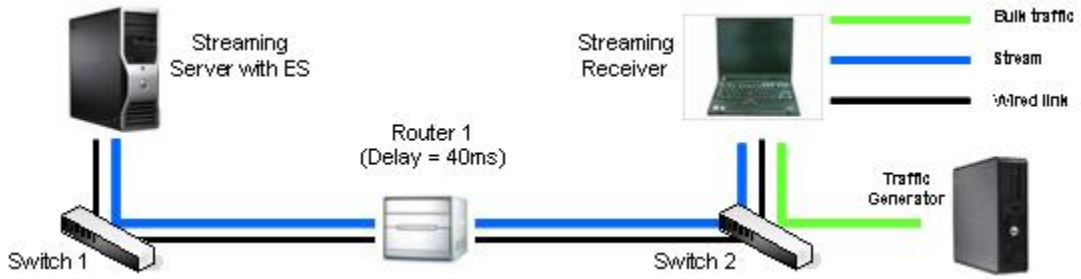


Figure 5-21 Congestion response experimental configurations

Figure 5-21 shows a case when congestion happened when a large amount of FTP traffic started through the same router. In this case, the streaming session may encounter lager jitter and losses although streaming server does not have a high load level. If such kind of congestion is detected, the expert control will handle the QTSS to perform a resolution adjustment. Initially stream-2 (movie 480p) is used. The related results are shown in Figure 5-22 and Figure 5-23.

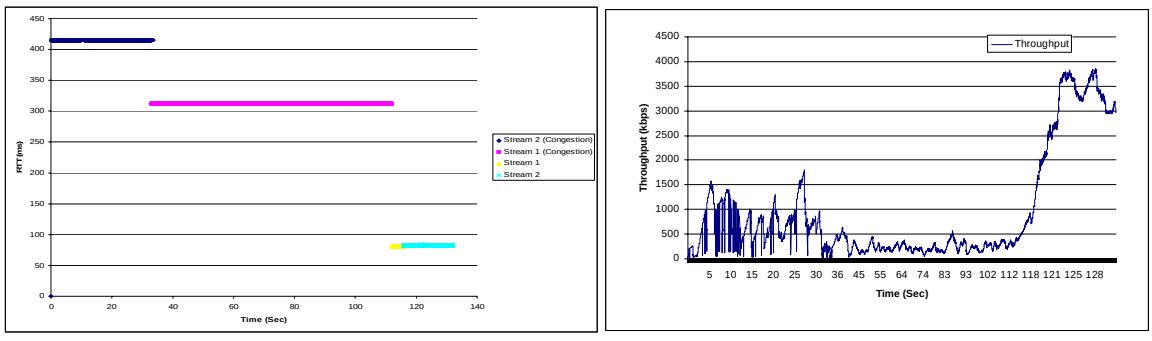


Figure 5-22 RTT and throughput during client/network congestion

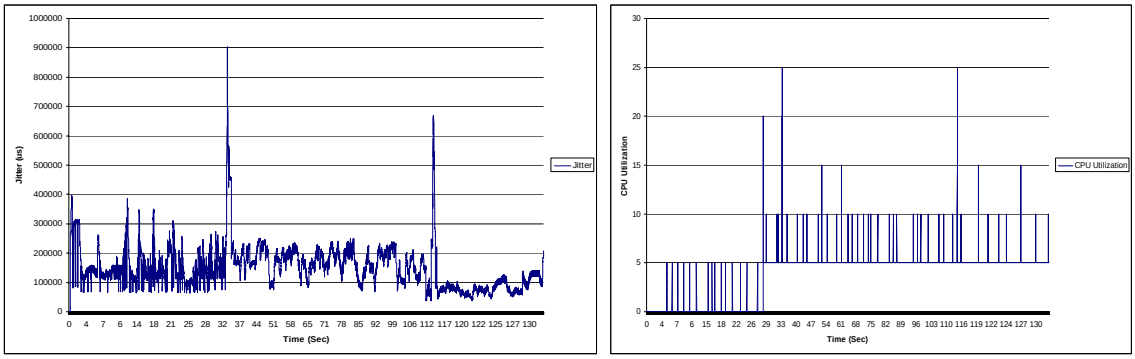


Figure 5-23 Jitter and CPU-utilization during client/network congestion

We started the expert control at around 28s. Before the expert control is switched on, the system has been congested for a while. In Figure 5-22 right picture, the throughput before 32s did not reach the required level for stream-2, and the loss rate at that time period is up to 50%. After the expert control was turned on, it detected that the congestion was already severe and was not due to the server side jam. It rescheduled a lower resolution movie (stream-1) to help alleviating the network or client side congestion. The throughput and jitter from 32s to 112s clearly shown such a movie change. When congestion past at 105s, represented by sharply decreased RTT and loss rate, the expert control schedule a recover procedure to change the resolution of stream back to the original one (stream-2). The CPU utilization during movie switches shot up to 25% temporarily for less than 100ms.

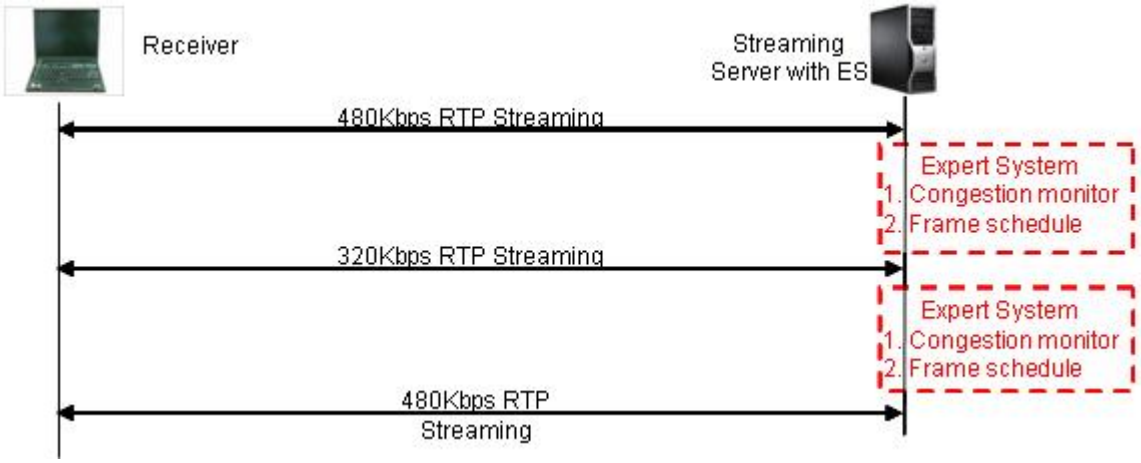


Figure 5-24 Signaling during handover from high to low resolution movies

Step 4: Session Re-distribution

If congestion at the server side local network is detected, the expert server could transfer the session to another server. In Figure 5-25, red dotted line is congested due to a large amount of background traffic from server 1 out. The expert control selects and contacts another server 2 to take over the current session(s). Movie-320 is used in this experiment. The background traffic and the server outgoing parameters are listed in Table 5-13.

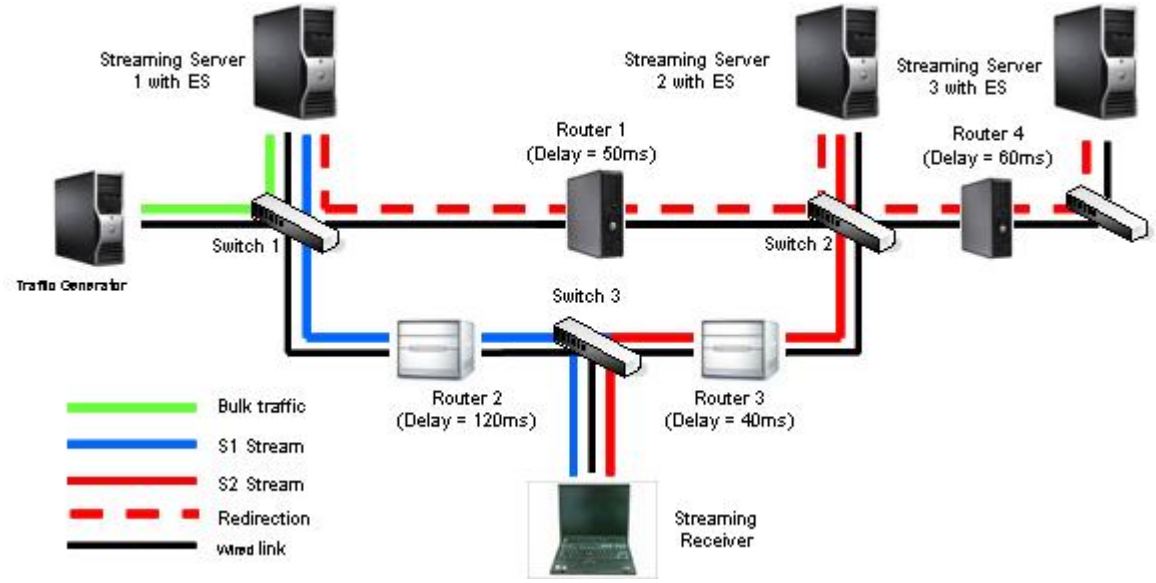


Figure 5-25 Test bed configuration for session re-distribution

Parameter \ Server	**S1**	**S2**	**S3**
Available Bandwidth	10M	100M	50M
Delay	120ms	40ms	100ms
Background traffic	8Mbps FTP	2Mbps FTP	4Mbps FTP

Table 5-13 Setup parameters of session re-distribution test

The results are obtained at client side and shown in Figure 5-26 and Figure 5-27. Throughput and jitter are obtained at client side. CPU utilization is captured from two servers. The data before session transfer (around 22s) describes the CPU utilization on S1 and the rest bars represents the CPU usage on S3. All background traffic start together at the beginning of streaming.

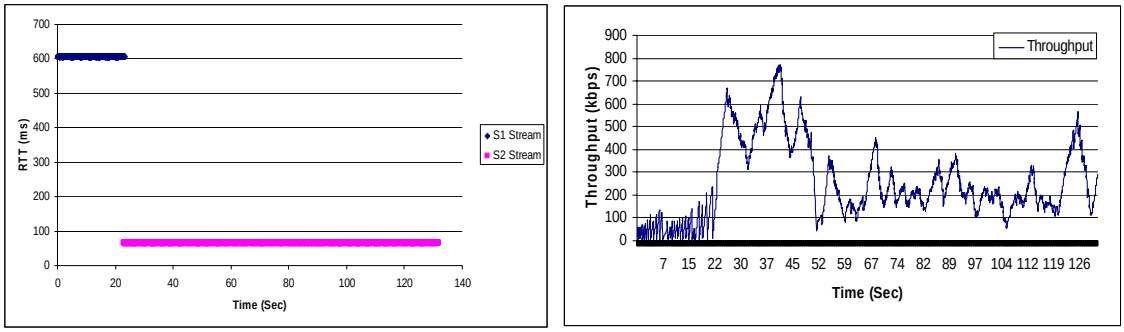


Figure 5-26 RTT and throughput during server side congestion

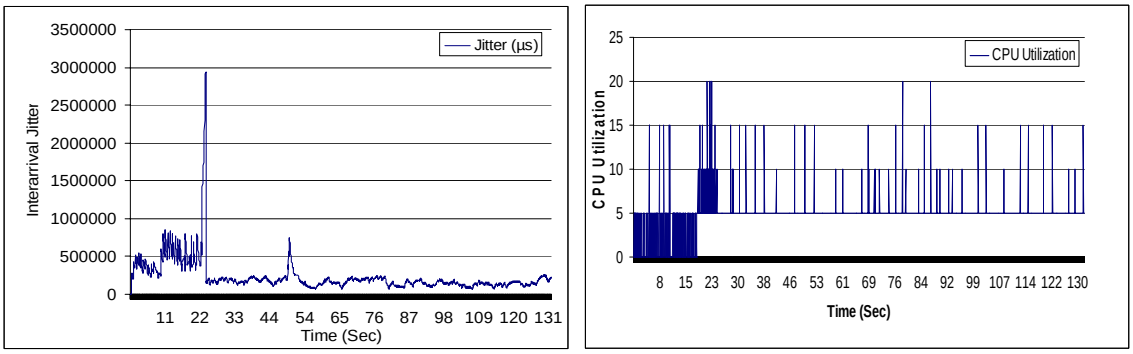


Figure 5-27 Jitter and CPU-utilization during server side congestion

We turned on the expert control at around 18s. After 5 seconds detection, the expert control confirmed the congestion happened at server side, so it selects the most uncongested server for help. After negotiation, the current session is taken over by S2, which has largest capacity and is least congested among the three servers. Then the session is totally transferred to S2 at around 24s, and the throughput at client side suddenly increased. During this transfer, the client side jitter became four times as before and the streaming is not continuous for 2 seconds. Here we just use the plain client player. If a modified client device could feedback its buffer and playback information, our rate control methods could provide a much higher and stable client buffer occupancy, and the buffered data would support the playback during session transfer. The handshake signaling for such session re-distribution is shown in Figure 5-28.

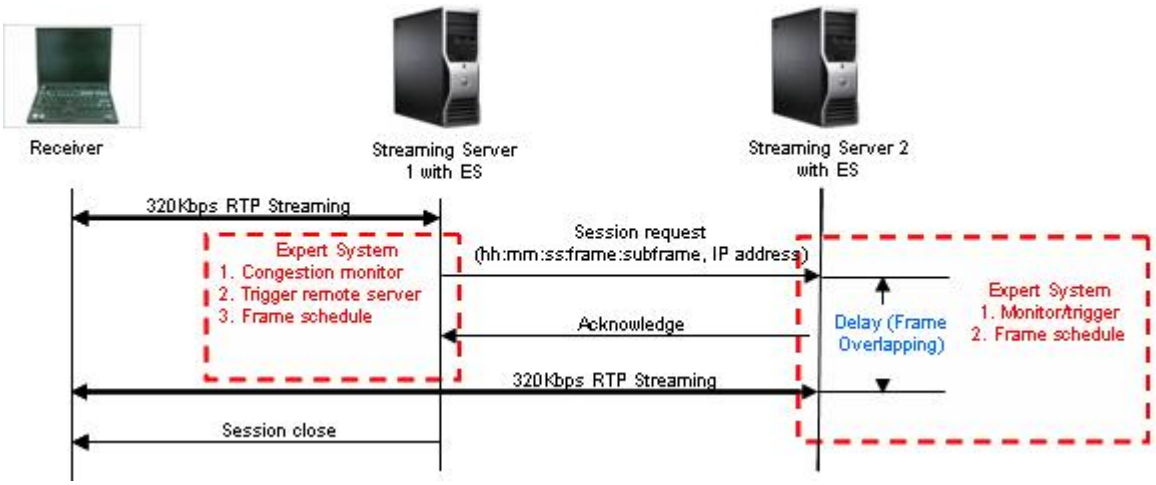


Figure 5-28 Signaling during session re-distribution

Now we only realized the transfer of sessions during congestion. Ideally, the other server should be able to provide P2P support to the congested sessions. That is, servers would drop the movie file into pieces and each one delivers a part of pieces separately to the client. This implementation needs the support of client software because client must know how to concatenate these pieces from different sources together. Since our design targets at server, we skipped such kind of implementation.

5.4 Summary of Experimental Results

In this chapter, we presented the detail implementation of the rule-based expert system in streaming media servers. At the beginning, we introduced the test bed configurations, the movie characteristics, the parameters intent to measure, and the QTSS platform. Then we gave the DTD of our rule base, followed by the explanations of parsing and linking process. We choose cooperative mechanisms of buffer management, packet scheduling, and congestion control to form the method base in the expert server. A complete rule base was built. In the results and discussion part, we provided the detail configuration for each experiment. The measured parameters are listed and evaluated. Then we presented five

carefully designed experiments using QTSS server, comparing the results with and without our expert control. The experiments investigate the ability of expert control on balancing the servers load, scheduling the playback sequence, realizing smoother rate control for high definition movies and meanwhile maintaining high stable client buffer occupancy. It could also perform streaming handover between different devices and accomplish effective congestion control in four consecutive steps.

From the results, the requests reached QTSS server are processed locally and served in a best effort manner. The excessive requests cause the previous sessions discontinuous. When we distribute the requests to four QTSS servers with different percentage, the CSR is 43.75%, 87.5%, 80%, and 90% respectively, depending on the capacity of servers and the number of requests initiated to each. On the contrary, no matter which server the requests were initially sent, QTSS with expert control will judge the capacity of server cluster and perform effective admission control, then the requests are reasonably distribute among servers according to the server and the intermediate link parameters. Results have shown that QTSS with expert control could achieve approximately identical load distribution for despite the request distribution strategy, and the CSR is 100%. To achieve load balance ability, the expert server lengthened the startup delay to maximum 4 times as basic QTSS (582ms/145ms). Since the lengthened startup delay is only half a second, it is still acceptable as the tradeoff for the performance enhancement.

In sub-section 5.3.2, our expert server extended the current link speed scheduling to a broader area, which automatically schedule the playback sequence of mixed contents according to client profile, content profile, and advertiser profile. This function is most suitable to closed network ISPs and subscribers.

The rate control for high definition movie is effective in two ways. It decreased the fluctuation of throughput and speed up the stabilization time. Furthermore, the client buffer occupancy with expert control is significantly better than basic QTSS control. The QTSS method does not consider any client side parameter. Although the overflow at client buffer causes higher loss rate, which will consequently inform the QTSS for rate control, the influence is comparatively trivial. The expert system always considers client side parameters for its decision. Therefore it maintained the client buffer usage at a very stable level with no overflow and underflow. Similarly, the expert system controlled the server buffer efficiently. Due to the high and stable of buffer usage, the client could achieve seamless continuous stream during congestion response with movie resolution changes or session transferring to another server. Because the control parameters are calculated with a dedicated function, the CPU consumed on rate control could be neglected. In summary, for the purpose of maintaining smoother stream and efficient client buffer usage for high definition movies, the expert system has demonstrated itself as a good choice.

In section 5.4.4, we tested the smartness of our system on changing terminal devices during playback. Basic QTSS has no such function, and such kind of system is rare in current server market. Consider the cases that the client device switches from slower hand phone streaming to a faster network with plasma TV display, or vice versa, the expert server could embed SIP-like protocol as signaling to support the movie transfer.

When congestion happened, the expert reacted smartly in steps. Instead of fluctuating with changes of network environment aimlessly, it verified the severe of congestion by checking related parameters like server load level, network RTT and loss, and client side loss and buffer occupancy. It takes cooperative regulations for the whole transmission to cope with the

congestion, rather than only decreasing the sending rate blindly. The reactions in sequence could be summarized as starting congestion rate control for HD content, turning on prioritized-RED, changing to lower resolution movie file, and finally transferring the session to an un-congested server. If a lower resolution movie is chosen in response to the congestion, the expert system recorded it in its historical statistics and automatically recovered to the high resolution movie after the congestion.

Attractively, the enhanced performances summarized until now can be achieved by spending only a small portion of CPU time for the expert control. In all experiments, the expert control took only 10% to 15% the CPU time periodically. Even when intensive reaction like handover is required to be taken, the CPU utilization increased to maximum 25% for a short time as 100ms before return to the normal level. However, we should note that CPU results may not exactly describe the time consumed by the expert control because the inference process embedded in QTSS is not protected by semaphores and therefore may be preempted by transmission tasks. Nevertheless, the overhead brought by the expert control has been tested to be acceptable. Thus the expert system is feasible for practical use.

The performance obtained in the experiments depends largely on the effective of rules that written by us. Hopefully, if these rules could be modified by a group of experts on media streaming techniques and be adjusted in commercial environment, the expert server system could perform much better than in our experiments.

Chapter 6 A Novel Rate Scheduler in Expert Server Knowledge-Base: DLQ Rate Control

The knowledge base is a major component that gives intelligence to the expert system. We hope it covers major types of rate control methods. Unfortunately, there were a lot of server-oriented or network-oriented schemes but no suitable client-oriented method in literature. Therefore we have to design a new client-oriented rate control method for the completeness of the knowledge base. The designed method, named DLQ (Discrete Linear Quadratic) rate control, was used for congestion avoidance.

In this chapter, we propose our design of this client oriented rate scheduling method. Our study shows that DLQ is superior to conventional rate control schemes especially on client buffer utilization. In the following paragraphs, it will be also called *DLQ schedule system* because it is a system that schedules sending speed for packets.

6.1 Introduction

Linear Quadratic (LQ) control is a well developed theory in automatic control area but its implementation in media packet scheduling started only from this century. In this section, we are going to provide the background of optimal control and introduce the DLQ we used for media transmission, and then the reason of choosing DLQ. Finally we will restate the relation between DLQ and the rule-based expert server system design.

6.1.1 Optimal Control

Optimal control ([73], [74]) is a mathematical field that is concerned with control policies that can be deduced using optimization algorithms. It deals with a close loop system and tries to find a control law for it such that a certain optimality criterion is achieved. Usually, the optimality criterion is either a measure of performance to be maximized or a cost function to be minimized. Given a dynamic system with input *u(t)*, output *y(t)* and state *x(t)*, the control law can be derived by solving the Hamilton-Jacobi-Bellman equation. The Jacobi function usually takes the form of an integral over time of a certain function, plus a final cost that depends on the state in which the system ends up:

$$J = \int_0^T l(x, u, t)\,\mathrm{d}t + m(x_T).$$

In the formula, *l* is the system description function. Parameters *x*, *u*, *t*, *T* are the state variable, the control variable, the time variable, and the terminal time respectively. And x_T is the final state. The control law *u=f(x)* would be derived by minimize or maximize the left hand side *J*. For a dynamic system, the control scheme should not only control the system, it also needs to estimate the system states in order to provide the best feedback information for a better performance of the control scheme. This work is often done by the filters, which are designed to extract useful information from the background noise or predict possible changes under various environments.

6.1.2 Linear Quadratic Control

Linear quadratic control is a typical problem in the optimal control area. It is also called mean-square control in its early stage. The term ‘linear’ means the systems considered were assumed linear. And the term ‘quadratic’ comes from the evaluation function that contains the

square of an error signal. In general, the problem considers minimizing a quadratic performance measure:

$$J = \int_0^\infty (x^T(t)Qx(t) + u^T(t)Ru(t))\,\mathrm{d}t$$

Subject to the linear dynamical constraint:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

where the matrices Q and R are positive semi-definite and positive definite, respectively. The optimal control problem defined with the previous functional is usually called the state regulator problem and its solution is a feedback matrix gain of the form:

$$u(t) = -K(t) \cdot x(t)$$

where K is a solution of the continuous time dynamic Riccati equation. This problem was elegantly solved by Rudolf Kalman (1960). The DLQ we used in our design is the discrete form of linear quadratic regulator problem. It is to find a state-feedback control law of the form $u_i = -k_i x_i$ that minimizes a quadratic performance measurement function for a linear system. The aim is to maximize the client buffer usage with minimum control efforts. If there were no disturbances, the system could stabilize with a minimum index function value. With disturbance, the solution formula includes an additional factor to trace the disturbance as shown in next section. Thus we considered the scheduling problem as a DLQR problem under disturbance in our design. The disturbance here comes from the decoding procedure. Media data in the client buffer is fetched by the decoder for playback. We refer to this fetch rate as the client buffer vacancy rate, which is a random variable.

6.1.3 Reason of selecting DLQ

Recall the categories classified in Chapter 2, researchers either use priority schemes to isolate timing sensitive flows from bursty ones or enable reservations to guarantee QoS. Packet scheduling schemes like separate priorities for different frames ([75]), multi-channel data scheduling ([76]), multi-thread distributed delivery ([77]), and real media-rate control ([78]) are widely used. All these mentioned schemes are network oriented open loop control. Here we consider the DLQ as an end-to-end close-loop control for its following merits.

- It is cost-effective. If the received data were not consumed by the decoder, DLQ could result in strictly optimal speeds to gradually fill the buffer.
- It is fairly accurate. The precise mathematical calculations supporting the method allow it to trace and control the system accurately and effectively.
- It is easy to implement. DLQ is an end-to-end control method that neglects the complexity of intermediate networks.

Previous works [79] and [80] used the similar method but they are incomplete. First, all previous approaches use continuous system model, which is not proper for network transmission system. Although multimedia can be delivered in a streamed way, it is handled discretely on the rate control layer. Second, they did not consider network delay and noise. Last, no detail mathematical deductions are provided. Therefore, we try to mend these gaps and gives out a systematic design of DLQ rate scheduler.

6.1.4 Relation with Expert Server System

The DLQ, together with other rate control and congestion avoidance methods, were implemented into the knowledge base of the rule-based expert server. It will be selected by the inference module in competition with other methods according to the runtime parameters.

6.2 Mathematical Design of DLQ

In this section, we are going to introduce the detail design of DLQ scheduler. The network model is set up first and the control procedure is introduced. Following them, we provide the mathematic model and solution for DLQ schedule problem. The Kalman filter is introduced to minimize the influence of noise. For the consistency of this section, we put the design of DLQ Kalman filter into appendix C.

6.2.1 Network Model

The network model is described in the Figure 6-1. For clarity, we do not draw the other supporting mechanisms but only the DLQ-scheduler-related parts. On the server side, the scheduler calculates the transmission parameters according to the client buffer setup and the video information; then saves them into the control parameter table in server memory. During the transmission, the scheduler looks up the table for parameters of this video. The optimal sending rate is calculated during runtime using these parameters and feedback information from the client.

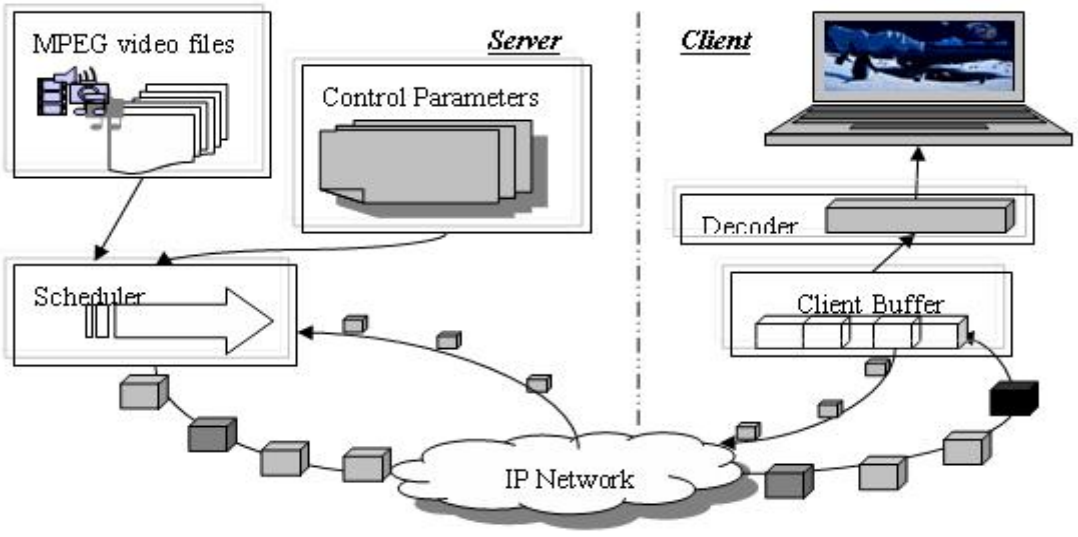


Figure 6-1 Network model for media stream transmission

On the client's side, media streams are received into the client buffer. The buffer monitor records the current buffer occupancy and the buffer vacancy rate. Then it sends this information back to the server periodically at a certain sampling rate.

6.2.2 Mathematical Solutions

The mathematic model is shown in Figure 6-2. The client gives the feedback of buffer vacancy x_i and buffer vacancy rate v_i. Buffer vacancy rate is the rate on which media data in client buffer are fetched for decoding. It is a Gaussian variable randomly distributed between the maximum and minimum playback rate of the transmitted movie. Q_r is the allocated client buffer size. Q_i is the instant client buffer usage. The difference between Q_r and Q_i gives the buffer vacancy x_i and it is feedback to server. The server calculates the optimal transmission rate u_{i_pre} according to the feedback x_i and control gain k_i, then considers the decoding rate v_i with feedback gain k_{i_fb}. Finally the optimal sending rate u_i is decided.

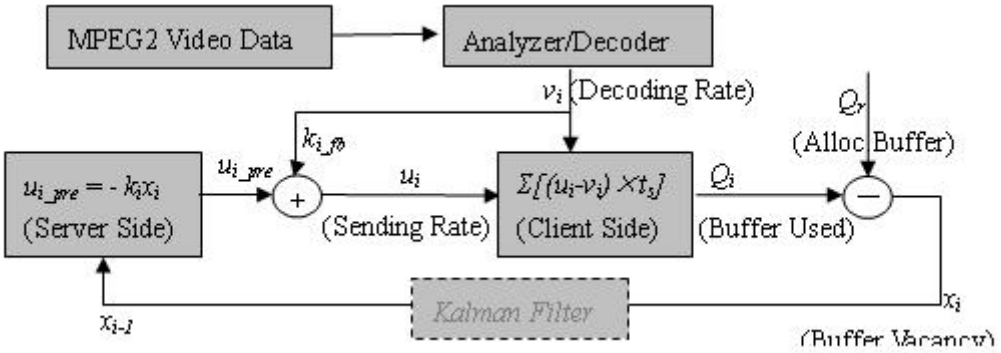


Figure 6-2 DLQ scheduler mathematical model

A Kalman filter is added to handle the noise along the transmission path. In figure 6-2, we use dash-line rectangle for Kalman Filter module because of two reasons. Firstly, this filter is not a must for all situations. In wired network where noise is negligible, this filter could be omitted for the simplicity of system. While in unstable wireless networks, it is highly

recommended to get a more accurate state variable estimation. Secondly, the design of this module is comparatively independent. It does not influence the DLQ system design. We omit it in the following mathematical deduction and provide its design in the appendix C. Here we did not consider the transmission delay between the client and server. The delay problem will be discussed in a separate section.

With reference of the mathematical system model (Figure 6-2), the state function and the index function are:

$$x(i+1) = x(i) + t_s u(i) - t_s v(i) \qquad i \geq 0 \qquad \text{---(6.1)}$$

$$J = \sum_{i=0}^{N} [x^2(i) + u^2(i)] \qquad \text{---(6.2)}$$

The sum of quadratic term of buffer vacancy *x(i)* and sending rate *u(i)* is the measurement criteria. The following deduction focuses on finding a $u^* = -kx$ that minimize the *J*. Definitions of parameters are listed in table 6-1. Here we use (-client buffer vacancy) as the state variable to make its coefficient and the coefficients of control variable (*u*) to be positive.

Variables	Signification	Explanation
x	-Client Buffer vacancy	State Variable (Bytes)
u	Optimal Sending Rate	Control Effort (Bytes/s)
v	Client Buffer Vacancy Rate	Disturbance (Bytes/s)
t_s	Sample Interval	Sample Rate: min. twice/frame

Table 6-1 Definition of variables in scalar DLQ formula

Let $J^*(x_i, i)$ denote the minimal value of performance measure starting at time $t=i \times t_s$ and state $x(i \times t_s)=x_i$. Then the optimality principle states that any input that is optimal over the interval (*i*, *N*) must necessarily be optimal over the interval (*i+1*, *N*). So that the following recursive relation must hold true:

$$J^*(x_i, i) = \min_{u_i} \{x_i^2 + u_i^2 + J^*(x_{i+1}, i+1)\}$$

Because J* has the quadratic form, Let $J^*(x_i,i) = p_i x_i^2 + 2b_i x_i + c_i$, so that:

$$J^*(x_i,i) = \min_{ui}\{x_i^2 + u_i^2 + p_{i+1}x_{i+1}^2 + 2b_{i+1}x_{i+1} + c_{i+1}\} \quad \text{---(6.3)}$$

To find the u_i that minimize J^*, we let:

$$\frac{\partial J^*(x_i,i)}{\partial u_i} = 0$$

That is: $2u_i^* + 2t_s p_{i+1}(x_i + t_s u_i^* - t_s v_i) + 2t_s b_{i+1} = 0$

$$u_i^* = \frac{-t_s p_{i+1}x_i + t_s^2 p_{i+1}v_i - t_s b_{i+1}}{1 + t_s^2 p_{i+1}} \quad \text{---(6.4)}$$

Rewrite (6.3) as:

$$p_i x_i^2 + 2b_i x_i + c_i = x_i^2 + u_i^{*2} + p_{i+1}x_{i+1}^2 + 2b_{i+1}x_{i+1} + c_{i+1}$$

Substitute (6.4) into this function, we have the following three functions (See Appendix B).

Quadratic terms in x:

$$p_i = 1 + \frac{p_{i+1}}{1 + t_s^2 p_{i+1}} \quad \text{---(6.5)}$$

Linear terms in x:

$$2b_i = \frac{2b_{i+1} - t_s p_{i+1}v_i}{1 + t_s^2 p_{i+1}} \quad \text{---(6.6)}$$

Terms independent of x:

$$c_i = t_s^2 p_{i+1}v_i^2 - \frac{t_s b_{i+1}v_i(2 + t_s^2 p_{i+1}) + t_s^2 b_{i+1}^2}{1 + t_s^2 p_{i+1}} + c_{i+1} \quad \text{---(6.7)}$$

Substitute (6.6) back into (6.4), we get the final formula of u^*:

$$u_i^* = -\frac{t_s p_{i+1}}{1 + t_s^2 p_{i+1}}x_{i-1} + \frac{t_s^2 p_{i+1}}{2(1 + t_s^2 p_{i+1})}v_{i-1} - t_s b_i \quad \text{---(6.8)}$$

Given the terminal values of p_n and b_n, (6.5) and (6.6) will decide all p and b values

recursively. In equation (6.8), the coefficient before x_{i-1}, that is $t_s p_{i+1}/(1+t_s^2 p_{i+1})$, is the k_i in figure 6-2 and the coefficient before v_{i-1}, that is $t_s^2 p_{i+1}/2(1+t_s^2 p_{i+1})$, is the k_{i_fb}. The aim of this DLQ tracker rate control scheduler is to minimize the client buffer vacancy (represented by x) while at the same time saving as much network bandwidth (represented by u) as possible.

6.2.3 Compatibility of DLQ in Expert Server

DLQ is merely a rate control method designed for taking care of client buffer usage and avoid congestion. During transmission, it may be selected with other QoS management methods. Thus it is necessary to discuss here the compatibility of DLQ with other mechanisms the server may use.

The main supporting mechanisms that a server adopts to enhance its performance and capacity are the video caches and frame filters. Caches are used to store frequently referenced contents for quick browsing and frame filters are used to decide which contents to send according to network capacity. DLQ is compatible with these mechanisms because it does not differentiate types of frames to decide the sending rate. Therefore, the server can drop one or two layer(s) according to the calculated transmission rate.

In fact, the whole system performs better if we combine DLQ with buffer management mechanisms together. Large u^* (optimal sending rate) means the client needs more data. Or in other words, data is consumed more quickly on the client side. Take t_s equals to one second as an example, more GOPs will be transmitted if only the basic layer is sent out within this second. More GOPs take longer time to playback. Thus it reduces the buffer vacancy rate. But if u^* is not too high, we can send all levels. The combination of DLQ with other support mechanisms is realized in expert system.

6.3 Results and Discussion

The assumptions for implementations are listed as follows. In fact, if outgoing bandwidth is enough for transmission, these assumptions will not influence the solutions we get.

1) No transmission delay between client and server is considered.

2) No other services (http, ftp, etc.) except media streaming are provided by the server.

3) Only one media stream is established between the server and a client.

There is another problem for implementation. As we mentioned, all parameters should be calculated and stored in a parameter table to minimize the runtime overhead. But from (6.6), v_i (online value) is also necessary to get b_i (offline value). There are two ways to obtain v_i without a real transmission. First, a long-dependent MPEG video model ([81]) can be used. Second, analyze the video in each frame and give out the estimated value. We choose the second method based on two considerations. One is that the stored movie is available to analyze for accurate v_i. The other is that the MPEG2 long dependent model is complex for implementation and not so accurate for various types of videos. Thus, we use MPEG-analysis software to analyze the frames for the size and the playback time of each frame. The actual buffer vacancy rate in experiment is a little bit different from the v_i we used to calculate b_i.

The simulations were conducted using an (18,3) m2v video clip. The notation (18,3) is a MPEG encoding format that has 18 frames on a Group of Picture (GOP) with two B frames between a pair of I or P frames. The parameters of the video are listed in the table below.

Total Frames	8760	Video Length (Seconds)	292
Minimum Frame Size (Bytes)	2324	Playback time for each frame (s)	0.033
Maximum Frame Size (Bytes)	81340	Allocated Buffer Size (MB)	1
Playback Rate (Frames/s)	30	Sample Rate (Times/s)	62.5

Table 6-2 Key parameters in simulation

The client buffer we allocate (1MB) can accommodate nearly 5 GOPs. Sample interval is 0.016s (Nyquist theory: minimum twice per frame). Then we attained the performance of client buffer occupancy in Figures 6-3.

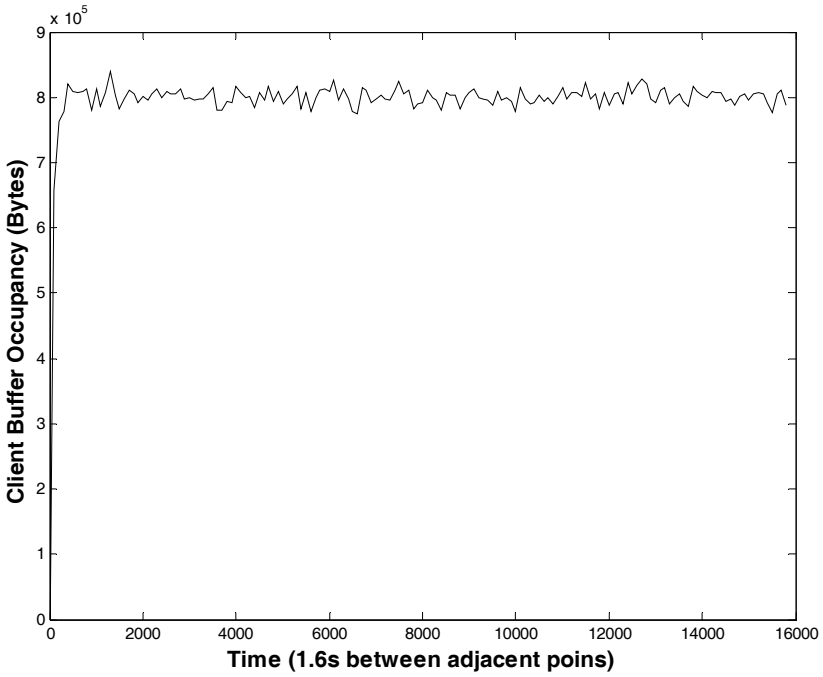


Figure 6-3 Client buffer occupancy using DLQ

The statistics show that the client buffer is, on average, around 79.86% full. At the beginning of transmission, the server sends data more than the client can consume. As a result, the client buffer occupancy becomes higher and higher until stabilization is reached. In our simulation, the maximum sending rate at very early beginning is 1.2MB/s. After the system reaches its steady state, the buffer occupancy stays between 0.7 MB and 0.86 MB with small fluctuations. Such high client buffer occupancy with small fluctuation enables the client having much less jitters because there is always enough data to be decoded.

If a maximum BW is set, for example 0.6 MB/s, the curve will climb up slower and take a longer time to reach the steady point. As shown in Figure 6-4, the system without maximum BW limitation rises faster than the system with maximum BW limitation at the beginning of

transmission. After all, these two lines emerge eventually after stabilization and give out the same buffer occupancy performance. This means the limitation of maximum sending rate has no influence on the steady state performance but only slow down the stabilization time.

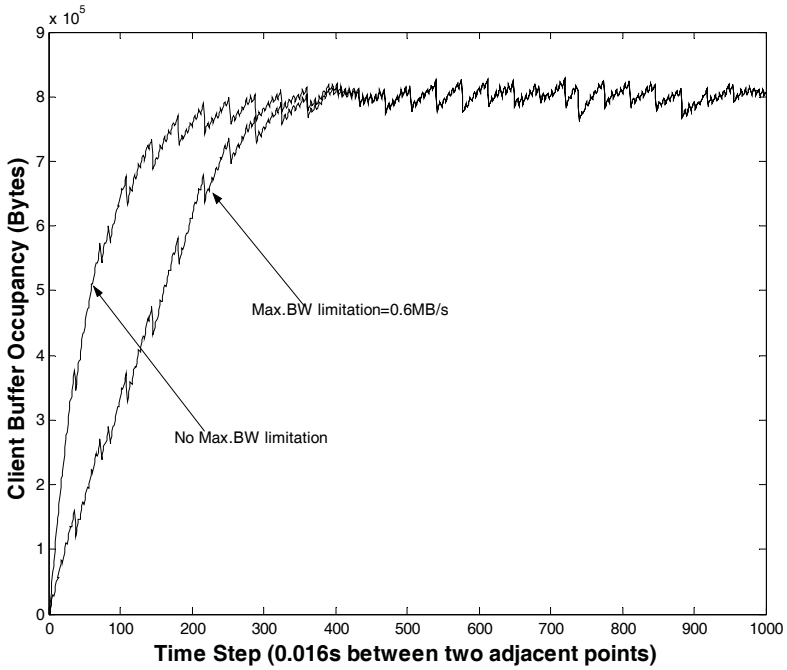


Figure 6-4 Rise time with/without BW limitation

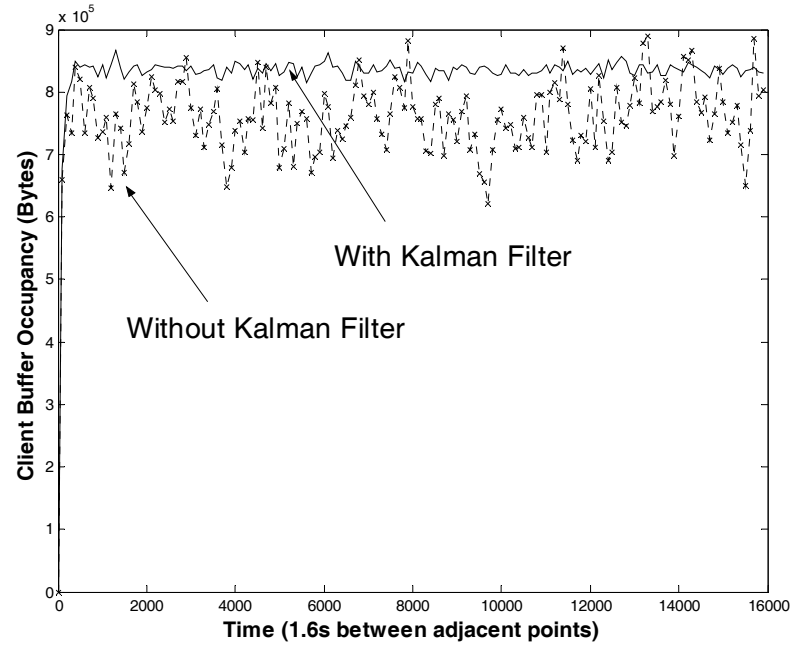


Figure 6-5 Buffer occupancy under noise with/without KF

If network noise is considered, the optimal rate with and without Kalman filter will be quite different (figure 6-5). In figure 6-5, the average deviation of state variable x due to network noise is set to 1.0×10^5 Bytes. Such noise degrades the performance of transmission by pulling down the client buffer occupancy and enlarging its fluctuation. Kalman filter lightens the problem and offers a stable delivery. To make the Kalman filter work more efficiently, it is important that the covariance of noise is properly probed. Techniques for online noise measurement could be used here together with the filter.

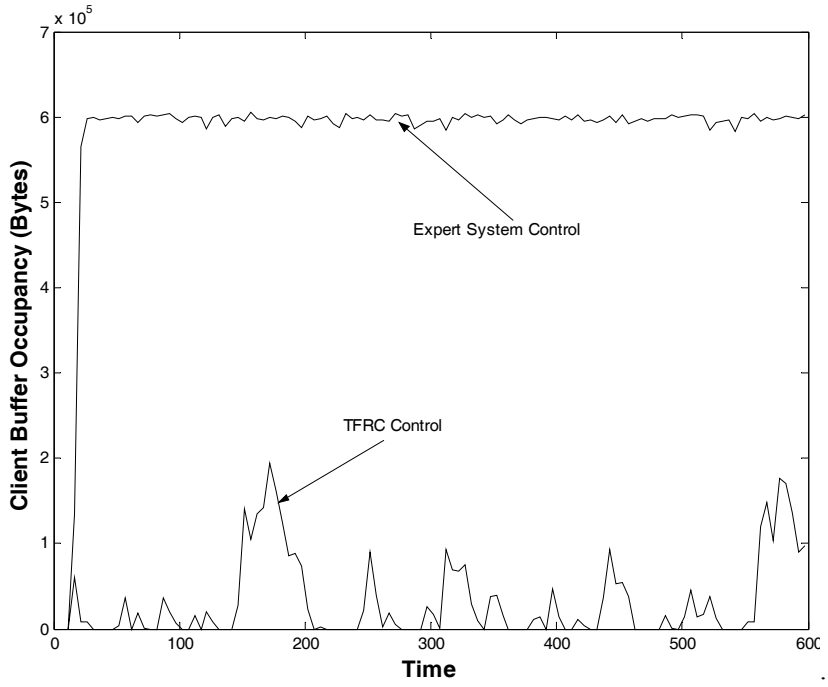


Figure 6-6 Client buffer occupancy under TFRC and DLQ control

Figure 6-6 compares the buffer occupancy under commonly used rate control method TFRC (TCP Friendly Rate Control) with DLQ rate control. It is obvious that the client buffer occupancy is quite smooth with DLQ control while the client buffer frequently encountered underflow under TFRC control. This is because the TFRC method targets at the intermediate network environment. It takes loss rate and delay as the input parameters to predict the network situation. The original sending rate is strictly held as long as the intermediate

network does not change. On the other hand, DLQ can detect client buffer overflow or underflow problems, collect the real-time playback rate, associate this information with the movie characteristics, determining the most suitable transmission bit-rate.

6.4 Impact of delay on DLQ rate control

Delay is an important factor for the streaming performance. The delay we discussed here is produced primarily by queueing delays in intermediate networks. We did not model the delay into function (6.1) and (6.2) because that will violate the linear property of the system. Since the DLQ performs control at discrete time points, the delayed feedback has to wait until the next control point to be considered. With this manner, the current *x(i)* is decided by *x(i-n)*, where n depends on the instant value of the delay. As a result, the state function is no longer a first order difference equation and the whole system is no longer a linear system that can be optimized by the DLQ method. Thus in this section, we investigate the delay impacts through testing and provide a practical one-step amendment strategy.

6.4.1 Assumptions

Assume the delay follow a normal distribution with mean value μ and variance σ^2. The following rules are used to simulate the behavior of basic DLQ under delayed feedbacks.

1) Round trip and process delays are represented by feedback delay only.
2) Feedback packets between two adjacent sampling points will be retained for a decision later on.
3) If no new feedback comes within a sample interval, the scheduler maintains the previous sending rate.

4) If several feedback packets come together during an interval, the scheduler takes the latest one for calculation and discards all the others.

6.4.2 Results of Delay Impact

We first fix the σ^2 to 144ms and investigate situations with mean values (μ) of delay to be 0ms, 32ms, and 128ms ([82]) respectively, which are 0, 2, and 8 times the sampling interval (16ms). Choosing basic DLQ for simulation, the client buffer occupancy with and without delay is shown below.

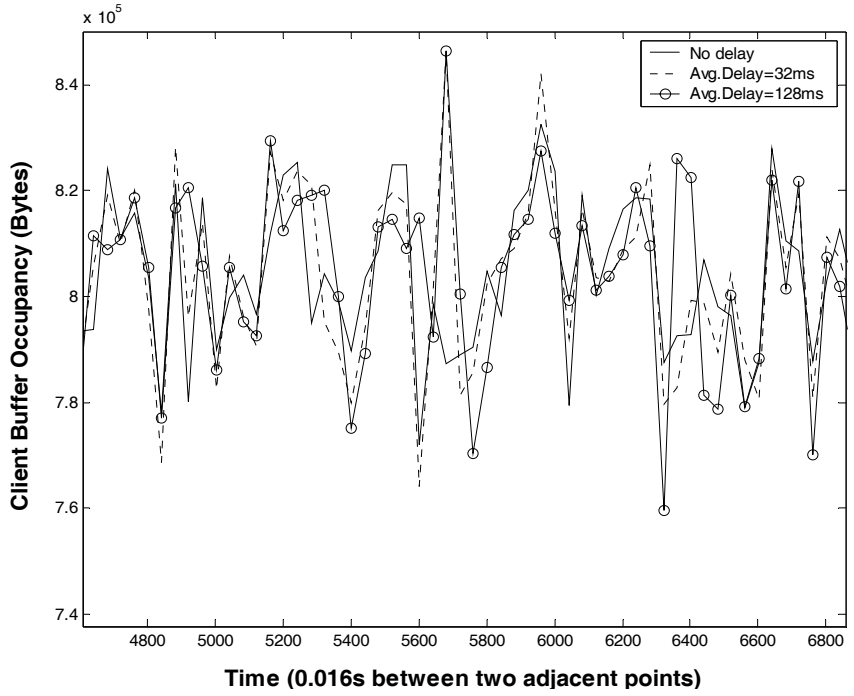


Figure 6-7 Buffer occupancy with mean delay of 0ms, 32ms, 128ms

Figure 6-7 is an enlarged picture to show the differences more clearly. From the figure, delay brings larger fluctuations to the buffer occupancy and the changes are not significant. For example, the buffer usage with no delay rises after 5600 points from 0.77 MB to 0.8 MB and then decreases (solid line), but for the delayed situation, the curve rises from 0.765 MB to 0.85 MB before decreasing (dotted and solid with circle line). Thus the delayed situation is

more likely to encounter overflow. Moreover, the buffer occupancy with larger delay (solid with circle line) swings away from no-delay situation (solid line) more seriously than the one with smaller delay (dotted line).

Now we fixed μ to 96 ms and investigated the variance (σ) of 144 ms and 1024 ms. Comparing it with no delay situation, we get the statistics in the following table.

Delay\Client buffer occupancy	Max (MB)	Mean(MB)	Median(MB)	Std(KB)
No delay	0.8395	0.7949	0.8008	64.69
σ^2=144ms	0.8370	0.7943	0.8013	64.69
σ^2=1024ms	0.8443	0.7948	0.8019	64.71

Table 6-3 Buffer occupancies with different σ of delay

From the table, delay variance also has small influence on buffer occupancy. Far from what is expected, delay does not bring hazardous disruption to the DLQ system. Reasons lie on the characteristics of MPEG2 video and the DLQ system itself. Delay affects two feedback variables: the current client buffer occupancy and the current buffer vacancy rate. However, MPEG2 video contains IBBP frames in repetition, and the sizes of the same type of frames are close. If the feedback packet for a frame is delayed to the sampling point for a following frame of the same type, the information of buffer vacancy rate it contains is near to the real value. On the other hand, since the DLQ adjustments are at a fine-grain level, the high sampling rate ensures the outdated feedback will not mislead the schedule decision to a long time. In other words, DLQ system reacts fast enough to correct its deflection.

6.4.3 Compensation for Delay Impact

The trouble in sub-section 6.4.2 was caused by two types of problems brought by the delay.

1) Time reverse problem

If several feedbacks came within a sample period in sequence, DLQ takes the one that arrived latest as the reference for decision. A feedback sent at time slot 10 may reach the server earlier than the feedback sent at slot 9. Suppose these two feedbacks come within the same sample interval, DLQ will discard the former one (sent at slot 10) but take the later one (sent at slot 9). This problem can be solved using a time stamp mechanism introduced later.

2) Outdated information problem

Cases where no feedback came within a sample period or feedback coming late are considered as the outdated problem. This problem can be solved by enhancing the network transmission speed which is not the scope of our design.

Within the two parameters influenced by the delay, buffer occupancy and buffer vacancy rate, making prediction on buffer vacancy rate during run-time will increase the system's complexity greatly without significant performance improvement. So the simple one-step prediction mechanism proposed here makes prediction only on actual buffer occupancy. We propose to give each feedback packet a time stamp when sent out. Receiving a new feedback, the DLQ scheduler compares it with the current time and predicts the current buffer occupancy using:

Buffer occupancy = Buffer occupancy in current feedback packet + Sent data during ([(Time stamp-Current time)/Ts]+1) steps – Playback data during this period.

Here, [*(Time stamp-Current time)/Ts*] means selecting the integer part of *(Time stamp – Current time)/Ts* and *Ts* is the sample interval. The playback data is calculated using the buffer vacancy rate in the current feedback packet. Of course if the system receives several feedbacks in a sample period, it compares their time stamps and trusts the latest one.

Adopting μ =128ms and σ^2 =1024, we redo the previous delay-influence simulation and add the curve with prediction mechanism. From figure 6-8, the simple prediction mechanism

(dotted line with cross marker) helps the buffer occupancy perform nearly as good as the no delay situation (solid line), much better than the no prediction situation (solid line with diamond marker). The mean square error between delayed and no-delay situations is 8.94×10^7, but only 5.98×10^6 between the delayed with prediction and no-delay situations.

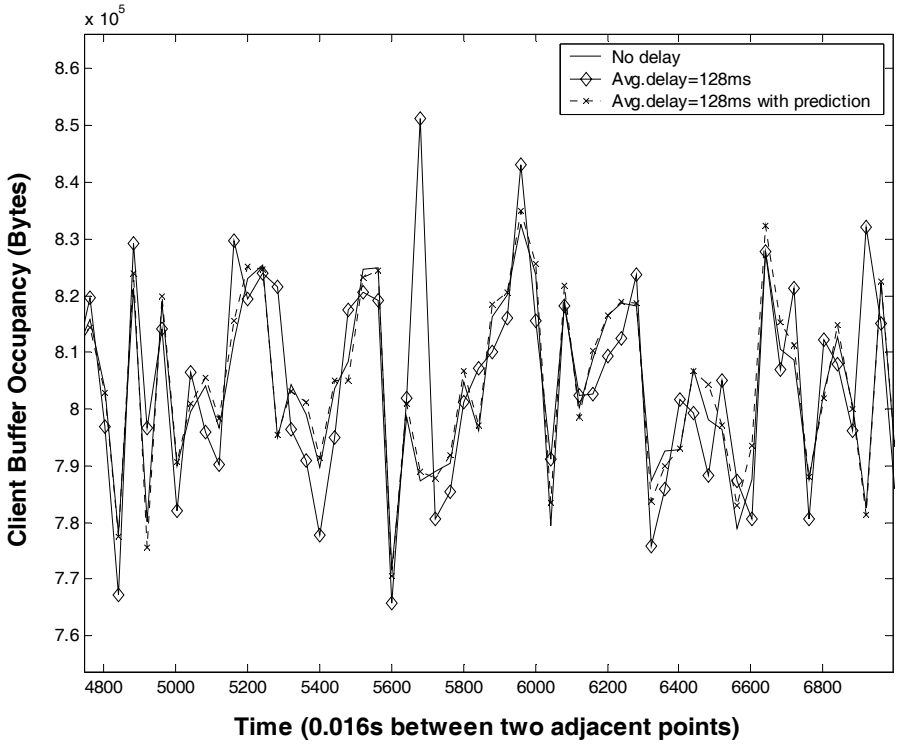


Figure 6-8 Buffer occupancy with delay prediction

6.5 Overhead of DLQ Rate Scheduler

The overhead brought by DLQ control are estimated here. From our test, the 3.0G CPU of SUN Fire 880 takes approximately 0.54 μs to depacketize the data, 0.04 μs to fetch the p_i and b_i from table, and 2.16 μs to do the calculation using formula (6.6). Switches between users consume around 2 μs. Then the total time is depacketize time + fetching parameters + calculation for optimal transmission rate + switch time between users consumes 4.74 microseconds. For the feedback interval of 16 ms, such a CPU can serve $16/(4.74\times 10^{-3})$ =

3375 users. Suppose the transmission is conducted on a 1G network bandwidth with 70% of its full capacity used for streaming, DLQ can support 70% * 1G / 0.4M = 1750 streams in the normal situation. The 0.4M in denominator is the steady state transmission rate obtained from the simulation. This result is much more than the number of client supportable by the expert server derived in Chapter 4. Thus DLQ scheduler will not decline the capacity of the expert server.

Chapter 7 Conclusion

At the end of the journey, we will take a global review on what has been presented in this thesis and conclude its importance, achievement, limitation and future trends. We hope that this chapter would further clarify the purpose of the research and our contributions. Most limitations mentioned in this chapter can be solved or at least improved by future research. The conclusion suggests a promising future of the expert server system for streaming control. The following paragraphs will answer the questions through three aspects: the strength of the expert system on streaming control; our achievements and limitations; and last, the possible research directions that may lead the expert server to be a practical and successful technique.

7.1 The Strength of the Expert Server

The research presented is inspired by the intention of making current streaming servers more powerful on streaming, flexible on control, and reliable on maintenance. Analyzing the current streaming servers, their performances have room for improvement given certain hardware and network configurations. The idea of using an interdisciplinary work by applying the expert system into conventional server design came up with the discovery that most expected improvements can be accomplished by the expert control. Now we first list the possible improvements from conventional servers and the solutions provided by the expert server system.

- The transmission quality would be better if the server could see a global picture of the transmission, and not only using some unrelated parameters of the stream. In the expert server, operation parameters are monitored and their relations are handled by heuristic

rules written by human experts. This makes the expert server perform like a real human expert who has insight into the problems and react intelligently to different cases, even when these cases have similar phenomena.

- The server may provide a customized service instead of a uniform delivery. This is the way most commercial servers are used. The expert system analyzes the client parameter, allocates proper resources and performs per flow rate control along the transmission. This client aware control could save resources and provide the same video quality.
- Instead of comparing which technique is better nowadays, the server could adopt all of them and take good use of them. The expert system refers to the rule base to select the most suitable lower level management strategies depending on the criteria of performance optimization. Even the search schemes and the selection criteria themselves can be coded as rules and adjusted online.
- The server should be easier to extend with the development of future techniques. Compared with conventional servers that fix their service strategy to unchangeable code, the expert system can be modified by simply adding or deleting rules and functions in the knowledge base. The maintenance is done without the influence of the main server program.

It can be concluded that the expert server system is an upgraded server with extended capacity and flexibility on control and maintenance. The expert server is a good combination of the integrated service and the differentiated service. It offers a differentiated service to each stream and integrates optimal delivery methods along the transmission steps. In the world without a one-for-all solution, the design of such an expert server system is an enlightening endeavor to find a powerful way for the problems under heterogeneous

environments.

7.2 The Achievements and Limitations of the Expert Server

Our research is a pioneering work for that it is the first time a multimedia streaming expert server system is designed, evaluated, and tested. The rule-based expert system design shown in Chapter 3 follows the conventional steps of designing an intelligence system. It considered the special characteristics of media streaming and applied this domain-related knowledge for the transmission as an expert. The position of such a rule-based expert system would be at the middleware level. It does not run all the time, but only performs periodic global control. Thus the overhead brought by it is controllable. The demands for better control and less overhead are balanced by selecting a proper monitor interval. The rule-based expert server system was theoretically evaluated in Chapter 4. Although it is difficult to analyze the transmitted video quality offline, we approximately analyzed the system from request response time varying with different rule base size. Using traditional forward chaining inference method, the computational time was bounded within a linear line and an exponential curve. The computation time gave the average service time and further decided the response time given certain client density. Due to the real-time characteristic of streaming requests, the maximum number of clients supportable by a single server was limited by the average requests response time. From theoretical analysis, the computational overhead (< 10%) brought by the expert control did not influence the server capacity, which has been proven to be settled by the network bandwidth. However the smoother streaming achieved by the expert control could greatly alleviate the network burden and consequently extend the system capability.

Practically, the expert server designed in this thesis realizes the following key features:

A. Perform effective admission control and traffic distribution. These two features are used

to limit the incoming requests. Accepted traffic would be distributed as illustrated in section 3.1.4, where the server will make a global decision based on latest working parameters and forward the request to the most suitable station for processing. Results have proved this function in sub-section 5.3.1

B. Adjust parameters dynamically and maintain QoS during runtime. It provides primary support for any intelligence pursuing by the expert server. The accuracy of on-the-fly parameters in working memory directly influences the correctness of the decision and consequent transmission quality. For the distributed server system, these working parameters are periodically broadcasted to update other servers.

C. Provide playback scheduling and streaming handover. Sub-section 5.3.2 and 5.3.3 described these two novel functions of the expert server. It gathered and classified the profiles from movie and advertisement provider, scheduling the playback sequence based on statistics of subscribers to maximize the entertainment effect. When the client device changes, it provides SIP-like signaling procedure to realize movie handover within different devices. This is a very useful function in modern families with multi-terminal receivers. Both the ISP and the users would be benefit from these two functions.

D. Carry out smooth rate control and buffer management for high definition movies. High definition movie requires high bandwidth. Even with fast development of physical network media material, it is still a killer application. To make the transmission of it smoother and more stable so that other traffic on Internet is not severely impacted, the expert control carried out efficient rate control. Test results in sub-section 5.3.4 shown the apparent effects on small throughput fluctuation and shorter stabilization time.

E. Perform knowledge based congestion control under diverse environments. We carefully

selected several typical congestion control methods and implemented them into our rule base, as shown in sub-section 5.2.4. The following results in section 5.3.5 demonstrated that the expert system could response to different level of congestions step by step intelligently. When it predicts a possible congestion, only rate control algorithms are start up. When light congestion happened, it locates and differentiates the jam. For server side jam, admission control becomes stricter. If the jam is at the network or at the client, prioritized RED is switched on to reduce the traffic of the stream and QoS management increases the packet importance. The experiment from Singapore to Chicago (sub-section 5.4.5) proved that expert control handles congestions more reasonably than a traditional server.

F. Implement failure detection and recovery mechanisms. The server system should not break down at any time. According to our design, if rule collision happens and no meta-rules aim at solving it, the expert server will output an error message and take the default value that has been set offline. If the collision is critical, the server will terminate itself after transferring current serving sessions to other stations.

G. The computational overhead is managed within an acceptable scope. The control overhead is a significant issue in the expert system design. In this thesis, we put much effort on handling this problem: the rule base is classified into functional groups; the parsing and linking process is done offline; the translated rules have concise structures; the binary rules are stored in fixed locations in the rule table; the monitor interval is set to occupy less than 10% CPU time; the inference is directed by effective heuristics. All these efforts performed effectively to reduce the overhead of the expert control. The analytical results (sub-section 4.2.1) and experimental results with each experiment

showed that the computational time was bounded within an acceptable range.

H. For the completeness of our rate control method base, we also designed a discrete DLQ rate control scheme with the Kalman filter. The DLQ, together with its modifications on delay impact, has been demonstrated to be effective on detecting the client buffer occupancy and dynamically adjust consequent sending rate. It is particularly suitable for the transmission with stable network conditions. Although the Kalman filter can deal with Gaussian noise on the transmission line, the DLQ will lost its strength if network bandwidth or loss rate changes greatly, for that it does not take the network parameter into calculation. To compensate this weakness, we use it together with other network sensitive rate control schemes. As the results shown in Chapter 5, the DLQ takes care of the client requirements while the other rate-control methods are aware of the network conditions. They make their own decisions and compromised to a more reasonable solution.

Besides all achievements listed above, the expert server system also has its limitations. We classify them into two groups and explain respectively in subsequent paragraphs.

❖ Inherent limitations

The expert server is inherited from expert systems family. Therefore it encounters the problems that appear in most expert systems.

1) When the rule size increases, the corresponding searching time may go up exponentially. This is a fundamental question for expert system design. The situation that is suitable for using the expert system often has its limitations on using conventional computer control algorithms. That is, the problem can not be solved easily by going through hundreds of lines of code. Problems with this attribute usually needs abundant of heuristic knowledge

to progressively approach the optimal solution. Thus it may be concluded that problems fit for using expert systems are usually attached with a huge rule base. Unfortunately, those problems commonly have a deadline for the solution. So how to balance the rule size so that decisions could be given within the deadline is perplexing for the designers.

Unlike them, media streaming problem has special characteristics. It is not a must to use expert system for multimedia transmission. Obviously the streaming can be performed successfully without the expert control, as realized by those commercial streaming servers. Our purpose is to enhance the server's performance by adding the global expert control to enhance its intelligence on reacting to various situations. Hence the expert server is different with typical expert systems on the target problems. As a result, the expert server does not necessary need a super huge rule base. In fact, the rule base size could be restricted according to the decision deadlines since the basic function of the server has been completed by a fixed algorithm.

2) Control vibration

Recall that the function of an expert system is to perform global control. It decides the appropriate combination of delivery strategies based on working parameters. This technique makes the control more flexible and effective, yet it also brings problems if the strategies change too frequently. Consider a situation when network or server parameters vibrate, the expert system may switch on and off some strategies to trace the trend of changes. As we know, most rate control or congestion control strategies need some time to reach their stable points. As a result, switching it on and off without waiting for its stabilization is not good except to cause the system to fluctuate and become unstable.

To solve this problem, we created many statistical global variables to describe the long-

term behavior tendency. For example, the *Historical Request Arrival Rate* and the *Variety of Arrival Rate* appeared in section 3.3 (Figure 3-4) are two parameters used to record the statistical characteristics of the arrival rate. With them, heuristic rules can be written like an expert is monitoring the whole system and making judgments based on his experiences. Decisions on changing transmission strategies are not so simple as merely check the current working parameters. It also relies on the historical trends of the changes and the stabilization time of a strategy.

This solution brings along another question that whether the hesitation of changing strategies prevents the expert server from quick responsive to the changes of environment. The answer to this question is: it depends on the rules. The rules, or the intelligence from the experts, are responsible to adjust the balance point between the problem of control fluctuation and the server responsiveness.

❖ Design limitations

In addition to the inherent limitations that come along with most expert systems, the expert server described in this thesis has its design limitations due to some uncontrollable reasons. We point them out here to make the whole picture of the expert server design more comprehensible.

1) The provided expert server system does not carry out OS level scheduling.

   In sub-section 4.2.2, the streaming server is categorized to be a real time system that supports real time streaming applications. With real time requirements, the system needs the support of operation system for timely interrupt response and preemptive scheduling. For example, the packet receiver is an independent event driven thread; the monitor and the session handler are periodic task whose timely execution rely entirely on the

scheduling of OS. At the beginning of this research, we planned to tackle the OS level scheduling. However, this kind of work was not completed due to time limitation. Nonetheless, the server would definitely perform better if supported by a real time OS, from which the modules designed for the expert control could get guaranteed service.

2) The performances tested are not standardized due to lack of rule base benchmark.

   From the laborious description of expert server in this thesis, we have come to an understanding that the transmission quality depends largely on the effective of rules. On the other hand, there is no standard rule base as a benchmark for us to test the server performance. The case study given in this thesis only tests the congestion avoidance performance of the expert control. The rule base we built is not completed and has much more space for extension. Thus the results presented in the thesis are far less than the highest performance achievable by the expert server system.

3) Failure detection inference procedure is not implemented.

   Due to time limitation, the only failure detection mechanism implemented in our work is output an error message when collision happening and take the default value for safety. Actually, backward chaining techniques could be used to detect potential errors or diagnose the failure like the example given in sub-section 1.1.3, the SOAR system to diagnose the failure in circuit simulations. Consequent recover methods could be carried based on the detection results. This will enhance the expert server's reliability through preventing the server to accumulate errors that finally cause undesirable break down.

7.3 Future Development

There are many potential improvements that could be developed in the future. We suggest some directions that may be inspiring.

- Adopt advanced network topologies

  When client-server topology is used, the maximum overall system capacity is fixed. No matter how well the scheduling scheme is, the utilization factor can not exceed 1. This bound limits the growth of the population of clients. Since the expert system provides high level control and not relies on the hardware and network infrastructures, it could be implemented into modern networks, for example peer-to-peer (P2P) network.

  In the year 2001, P2P networks appeared and grew up very quickly and now it has been a practical technology for broadcasting or VOD streaming applications. The P2P network is a fully distributed system that each node in the network is both a server and a client. When a client device requires services from the network, its bandwidth and processing capacity is also added into the network. Thus the P2P network capacity grows together with the increase of clients. Such a characteristic enables P2P network providing unlimited file sharing services among clients. Using it, the service capacity of the whole system will not be bounded by the number of servers like in client-server architecture.

  The main problem for a P2P network is how to find the optimal group of peers for data exchange, regarding the reliability and distances. For such kind of selection, the human performs much better than computer if they are provided a local area traffic distribution graph and the character of each peer. It is hopeful that an expert system could bring such information-handle ability into P2P network. The marriage of these two technologies would greatly enhance the performance of multimedia streaming services.

- Self-training of rules

  Until now, the rule base is set up offline by experts who write heuristic rules according to their experience and knowledge of the streaming delivery. The rules may be outdated and

need frequent maintenance under different hardware configurations. Therefore we suggest creating some rules to record and evaluate the reliability of decisions and the performance of selected strategies. These rules are in charge of adjusting parameters of other rules during run time and enhancing the probability of making correct decisions in the future.

- Rule base auto-evolution

  If previous self-training ability could be obtained, a more intelligent feature would be make the execution of the server a parallel procedure with the rule base evolutionary process. That is, the rule base will update itself concurrently when server providing services. It may modify existing rules; discard outdated rules; or even creating new rules to fit for the changing circumstance. Considering the fast development of generic algorithms, we believe it is possible to realize this feature in the future.

Bibliography

[1] Tanimoto, S.L. "The Elements of Artificial Intelligence", New York, NY: Computer Science Press, 1987.

[2] Buchanan, B. and E. Feigenbaum, "DENDRAL and Meta-DENDRAL: Their Applications Dimension", Artificial Intelligence, vol. 11, 1978.

[3] Avelino J. Gonzalez, Douglas D. Dankel, "The Engineering of Knowledge-based Systems: Theory and practice", NJ: Prentice Hall, 1993, ISBN: 0132769409

[4] Durkin John, "Expert systems: design and development", New York: Maxwell Macmillan International, 1994. ISBN: 0023309709

[5]Suzanne Smith, Abraham Kandel, "Verification and validation of rule-based expert systems", CRC Press, c1993, ISBN: 084938902X

[6] S.J.Jang et al., "Automated Individual Prescription of Exercise with an XML-based Expert System", 27th Annual Conference on Engineering in Medicine and Biology, 2005

[7] Thomas Lumpp, Juergen Schneider, Wolfgang Kuechlin, Carsten Sinz "Loop detection in rule-based expert systems", Patent No.: 6952690

[8] Christopher W.Lehman in IBM Colorado Springs, Mary Jane Willshire in Dept. of Computer Science of Colorado Technical University, "A Rule-Based Expert System for the Diagnosis of Convergence Problems in Circuit Simulation", 2006

[9]Zakrzewski, E.J.; Quillin, R., "Applications of expert systems to network control", IEEE International Conference on Communications, 1988. ICC 88. Digital Technology - Spanning the Universe. Conference Record. 12-15 June 1988 Page(s):1734 - 1739 vol.3

[10] Erfani, S.; Malek, M.; Sacher, H.;"An expert system-based approach to capacity allocation in a multiservice application environment", IEEE Network, Volume 5, Issue 3, May 1991 Page(s):7 – 12

[11] Rao, M.S.S.; Soman, S.A.; Menezes, B.L.; Pradeep Chawande; Dipti, P.; Ghanshyam, T.; "An expert system approach to short-term load forecasting for Reliance Energy Limited, Mumbai", IEEE Power India Conference, 2006, 10-12 April 2006 Page(s):6 pp.

[12] Calleja, J.A.B.; Troost, J.; "Dealing with high workload in future naval command and control systems", IEEE International Conference on Systems, Man and Cybernetics, 2005, Volume 1, 10-12 Oct. 2005 Page(s):733 - 739 Vol. 1

[13] Jin Wu; Djemame, K.; "An expert-system-based structure for active queue management", International Conference on Machine Learning and Cybernetics, Volume 2, 2-5 Nov. 2003 Page(s):824 - 829 Vol.2

[14]Cai, L.; Shen, X.S.; Mark, J.W.; Pan, J., "QoS support in Wireless/Wired networks using the TCP-Friendly AIMD protocol", IEEE Transactions on Wireless Communications, Volume 5, Issue 2, Feb. 2006 Page(s):469 – 480

[15] Layaida, O.; Atallah, S.B.; Hagimont, D., "Reconfiguration-based QoS management in multimedia streaming applications", 30th Euromicro Conference, 2004, Page(s):248 – 255

[16] Xiaohui Gu; Nahrstedt, K., "Distributed multimedia service composition with statistical QoS assurances", IEEE Transactions on Multimedia, Volume 8, Issue 1, Feb. 2006 Page(s):141 – 151

[17] Peng Zhu; Wenjun Zeng; Chunwen Li, "Joint Design of Source Rate Control and QoS-Aware Congestion Control for Video Streaming Over the Internet", IEEE Transactions on Multimedia, Volume 9, Issue 2, Feb. 2007 Page(s):366 – 376

[18] Fitzek, F.H.P.; Reisslein, M., "A prefetching protocol for continuous media streaming in wireless environments", IEEE Journal on Selected Areas in Communications, Volume 19, Issue 10, Oct. 2001, Page(s):2015 – 2028

[19] Argyriou, A.; Madisetti, V., "A media streaming protocol for heterogeneous wireless networks", IEEE 18th Annual Workshop on Computer Communications, 20-21 Oct. 2003 Page(s):30 – 33

[20] Yasukawa, K.; Baba, K.-I.; Yamaoka, K., "Classification of nonstream flows to reduce negative interactions between stream and nonstream flows", IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 2003, Volume 2, 28-30 Aug. 2003 Page(s):772 - 775 vol.2

[21] Lombaedo, A.; Schembra, G.; Morabito, G., "Traffic specifications for the transmission of stored MPEG video on the Internet", IEEE Transactions on Multimedia, Volume 3, Issue 1, March 2001, Page(s):5 – 17

[22] Liu Yin; Liu Wenyin; Wei-Jia Jia; Jiang Changjun, "Link bandwidth detection for multimedia streaming in a distributed server environment", Joint Conference of the 4th International Conference on Information, Communications and Signal Processing, and the 4th Pacific Rim Conference on Multimedia. 2003, Volume 1, 15-18 Dec. 2003 Page(s):438 - 442 Vol.1

[23] Lombardo, A.; Morabito, G.; Palazzo, S.; Schembra, G., "A Markov-based model of MPEG-2 audio/video traffic", GLOBECOM '99, Volume 2, Page(s):1189 - 1193 vol.2

[24] Chandra, K.; Reibman, A.R., "Modeling one- and two-layer variable bit rate video", IEEE/ACM Transactions on Networking, Volume 7, Issue 3, June 1999 Page(s):398 – 413

[25] Nelson Tang, Sonia Tsui, and Lan Wang, "A Survey of Admission Control Algorithms," Computer Science Department, UCLA, December 16, 1998

[26] In-Hwan Kim; Jeong-Won Kim; Seung-Won Lee; Ki-Dong Chung, "Measurement-based adaptive statistical admission control scheme for video-on-demand servers", 15th International Conference on Information Networking, 31 Jan.-2 Feb. 2001 Page(s):471 – 478

[27] Vieron, J.; Guillemot, C., "Real-time constrained TCP-compatible rate control for video over the Internet," IEEE Transactions on Multimedia, Volume 6, Issue 4, Aug. 2004 Page(s):634 – 646

[28] Jiang, M.; Ling, N., "Low-Delay Rate Control for Real-time H.264/AVC Video Coding," IEEE Transactions on Multimedia, Volume 8, Issue 3, June 2006 Page(s):467 – 477

[29] Yuko Onoe et al, "Network information based Rate Controls on Multimedia Streaming Servers," 23rd International Conference on Distributed Computing Systems, 2003, pp.543~548

[30] Peng Zhu; Wenjun Zeng; Chunwen Li, "Joint Design of Source Rate Control and QoS-Aware Congestion Control for Video Streaming Over the Internet", IEEE Transactions on Multimedia, Volume 9, Issue 2, Feb. 2007 Page(s):366 - 376

[31] Floyd, S.; Jacobson, V., "Random early detection gateways for congestion avoidance", IEEE/ACM Transactions on Networking, Volume 1, Issue 4, Aug. 1993 Page(s):397 – 413

[32] Dong Lin and Robert Morris, "Dynamics of Random Early Detection", SIGCOMM'97

[33] Hutschenreuther, T.; Schill, A., "Content based discarding in IP-routers", 9th International Conf. on Computer Communications and Networks, 16-18 Oct. 2000 Page(s):122 – 126

[34] Bajic, I.V.; Tickoo, O.; Balan, A.; Kalyanaraman, S.; Woods, J.W., "Integrated end-to-end buffer management and congestion control for scalable video communications",

International Conference on Image Processing, Volume 3, 14-17 Sept. 2003 Page(s):III - 257-60 vol.2

[35] Bai, Y.; Ito, M.R., "User-oriented fair buffer management for MPEG video streams", 17th International Conference on Advanced Information Networking and Applications, 27-29 March 2003 Page(s):241 – 246

[36] Awad, A.; Sivakumar, R.; McKinnon, M.W., "MPFD: a look-ahead based buffer management scheme for MPEG-2 video traffic", 8th IEEE International Symposium on Computers and Communication, 2003 Page(s):893 - 898 vol.2

[37] Jane W.S. Liu, "Real-Time systems", NJ : Prentice Hall, 2000, ISBN: 0130996513

[38] Bennett, J.C.R.; Hui Zhang, "WF2Q: worst-case fair weighted fair queueing", 15th Annual Joint Conference of the IEEE Computer Societies, INFOCOM '96, Volume 1, 24-28 March 1996 Page(s):120-128 vol.1, Digital Object Identifier 10.1109/ INFCOM. 1996.497885

[39] Cheung, S.Y.; Pencea, C.S., "BSFQ: bin sort fair queueing", the 21th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2002, Volume 3, 23-27 June 2002 Page(s):1640 - 1649 vol.3

[40] IETF DiffServ Working Group page: http://www.ietf.org/html.charters/OLD/diffserv-charter.html

[41] West, R.; Zhang, Y.; Schwan, K.; Poellabauer, C., "Dynamic window-constrained scheduling of real-time streams in media servers", IEEE Transactions on Computers, Volume 53, Issue 6, June 2004 Page(s):744 – 759

[42] Xiaofei Liao; Hai Jin; Hao Chen, "Double-layer stream scheduling scheme with QoS control for cluster video servers", International Conference on Computer Networks and Mobile Computing, 2003, 20-23 Oct. 2003 Page(s):86 - 91

[43] Nguyen, T.; Mehra, P.; Zakhor, A., "Path diversity and bandwidth allocation for multimedia streaming", ICME '03, Volume 1, 6-9 July 2003 Page(s): I - 1-4 vol.1

[44] Prabhakaran, B., "Scheduling multimedia information delivery over unicast wireless channels", 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, March 2000, Page(s):33 – 37

[45] Jinsung Cho; Heonshik Shin, "Heuristic scheduling for multimedia streams with firm deadlines", IEEE 4th International Workshop on Real-Time Computing Systems and Applications, Page(s):67-72, 27-29 Oct. 1997

[46] Mehra, P.; De Vleeschouwer, C.; Zakhor, A., "Receiver-driven bandwidth sharing for TCP and its application to video streaming", IEEE Transactions on Multimedia, Volume 7, Issue 4, Aug. 2005 Page(s):740 – 752

[47] Christian Poellabauer, Karsten Schwan, Richard West, "Coordinated CPU and Event Scheduling for Distributed Multimedia Applications", ACM Multimedia, Ottawa, Ontario, Canada, October 2001

[48] http://www.microsoft.com/windows/windowsmedia/forpros/server/server.aspx

[49] http://service.real.com/help/library/guides/g270/realsrvr.htm

[50] Charles F. Goldfarb, Paul Prescod, "Charles F. Goldfarb's : XML handbook", N.J. : Prentice Hall PTR, c2002, ISBN: 0130651982

[51]Presern, S.; Brajak, P.; Vogel, L.; Zeleznikar, A.P., "An adaptable parallel search of knowledge bases with beam search", Proceedings of the 22th Annual Hawaii International

Conference on System Sciences, 1989. Vol.III: Decision Support and Knowledge Based Systems Track, Volume 3, 3-6 Jan. 1989 Page(s):262 - 270 vol.3

[52] S. Ortmanns, A. Eiden, H. Ney, N. Coenen, "Look-Ahead Techniques for Fast Beam Search," *icassp*, p. 1783, 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97) - Volume 3, 1997

[53] Rong Zhou and Eric A. Hansen, "Breadth-First Heuristic Search", 14th International Conference on Automated Planning and Scheduling (ICAPS-04), Whistler, British Columbia, Canada, June 3 - 7, 2004

[54]Thomas H. Cormen et al, "Introduction to algorithms", Cambridge, Mass. : MIT Press, c2001, 2nd edition, ISBN: 0262032937

[55] Thomas G. Robertazzi, "Computer networks and systems: queueing theory and performance evaluation", 3rd edition, New York : Springer-Verlag, 2000, ISBN: 0387950370

[56]Christian Maihofer, "A Bandwidth Analysis of Reliable Multicast Transport Protocols", University of Stuttgart, Institute of Parallel and Distributed High Performance, Systems (IPVR), Breitwiesenstr. 2022, D70565, Stuttgart, Germany

[57]Jain, Raj. "The art of computer systems performance analysis : techniques for experimental design, measurement, simulation, and modeling", New York : Wiley , c1991, ISBN: 0471503363

[58] http://developer.apple.com/documentation/QuickTime/QTSS/QTSS.pdf

[59] Liang Ji; Arvanitis, T.N.; Woolley, S.I., "Fair weighted round robin scheduling scheme for DiffServ networks", Electronics Letters, Volume 39, Issue 3, Page(s):333 – 335, 6 Feb. 2003.

[60] X.Zhou and K.Ong, “Discrete LQ rate control schedule system for multimedia transmission,” International Conference on Advanced Information Networking and Applications, Vol. 1, Page(s):6,18-20 April 2006.

[61] Jinyao Yan; Katrinis, K., May, M.; Plattner, B., “Media- and TCP-friendly congestion control for scalable video streams”, IEEE Transactions on Multimedia, Volume 8, Issue 2, April 2006.

[62] Lisong Xu, “Extending Equation-Based Congestion Control to High-Speed Long-Distance Networks: Smoothness Analysis”, IEEE Globecom, 2005.

[63]F.P.Kelly, A.Maulloo, and D.Tan, “Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability”, Journal of the Operational Research Society, 49, 1998.

[64] M. Dai and D. Loguinov, “Analysis of Rate-Distortion Functions and Congestion Control in Scalable Internet Video Streaming”, ACM NOSSDAV, June 2003.

[65] Y.P.Zhang, S.R.Kang, and D.Loguinov, “Delayed Stability and Performance of Distributed Congestion Control”, ACM SIGCOMM, August, 2004.

[66] S.R.Kang, Y.P.Zhang, M.Dai, and D.Loguinov, “Multi-layer Active Queue Management and Congestion Control for Scalable Video Streaming”, IEEE Distributed Computing Systems, 2004.

[67] Cai, L.; Xuemin Shen; Jianping Pan; Mark, J.W., “Performance analysis of TCP-friendly AIMD algorithms for multimedia applications”, IEEE Transactions on Multimedia, Volume7, Issue 2, April 2005.

[68] Y.P.Zhang; D.Loguinov, “Oscillations and Buffer Overflows in Video Streaming under Non-Negligible Queuing Delay”, ACM NOSSDAV, June 2004.

[69] M. S. Grewal and A. P. Andrews, "Kalman filtering: Theory and practice using MATLAB, 2nd Edition", John Wiley & Sons, 2001.

[70] JCDecaux Market Reach, http://www.jcdecaux.com.sg/content/research/reach.htm

[71] Massimo Bernaschi et al, "Adaptive Streaming on Heterogeneous Networks", WMuNeP'05, October 13, 2005, Montreal, Quebec, Canada.

[72] Zhou, X.F; Ong, K., "A Rule-Based Expert Server System for Multimedia Transmission", IEEE 21st International Conference on Advanced Networking and Applications, 2007. AINA '07, 21-23 May 2007 Page(s):305 - 310

[73] PeterDorato, Chaouki Abdallah, and Vito Cerone, "Linear-Quadratic Control, An Introduction," Prentice Hall, 0-02-329962-2

[74] Desineni Subbaram Naidu, "Optimal control systems," Boca Raton, FL: CRC Press, c2003.

[75] Markus Fidler, "Real-Time Multimedia Streams in a Differentiated Services Network," Dept. of Computer Science, Informatik IV, Aachen Univ. of Tech., Proc. 10th international conference on computer and communications and Networks, 2001, pp.308~385.

[76] Randy Chow at al, "Traffic Dispersion Strategies for Multimedia Streaming," Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'01), 2001, pp.18~24.

[77] Gwang S.Jung,Kyung W.Kang, and Qutaibah Malluhi, "Multithreaded Distributed MPEG1-Video Delivery in the Internet Environment,"

[78] Kang Li at al, "A Rate-Matching Packet Scheduler for Real-Rate Applications," Dept. of Computer Science and Engineering, Oregon Graduate Institute

[79] Mei-Ling Shyu1 et al, “Optimal Resource Utilization in Multimedia Transmission,” IEEE international conference on Multimedia and Expo, 2001. pp.673~676.

[80] N.Uchida, K.Takahata and Y.Shibata, “Optimal video stream transmission control over wireless network,” IEEE international Conference on Multimedia and Expo, Vol.3, pp.1791-1794, 27-30 June 2004

[81] O.Rose, “Simple and Efficient Models for Variable Bit Rate MPEG Video Traffic,” Performance Evaluation, vol.30, pp.69~85, 1997.

[82] http://ipnetwork.bgtmo.ip.att.net/pws/network_delay.html

[83]http://www.cisco.com/en/US/netsol/ns669/networking_solutions_solution_segment_home.html

Appendices

Appendix A: Extended Kalman Filter (EKF) for Kelly's Rate Control

This appendix gives the Kalman filter diagram, the loss rate state function, and the implementation algorithm for Kelly's rate control.

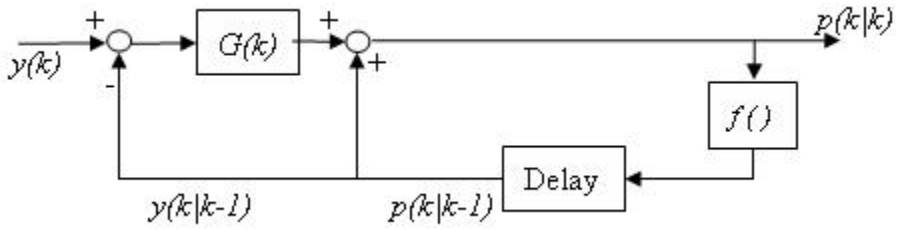


Figure A-1 Kalman filter block diagram for Kelly's rate control

The aim of Kalman Filter (with reference to figure A-1) in Kelly's rate control is to get the best estimation of loss rate *p(k)* given previous predicted value of *p(k-1|k-1) p(k-2|k-2)* … *x(0|0)* and the observed current state variable value *y(k)*. It tries to get a filter gain *G(k)* so that the prediction $p(k \mid k) = p(k \mid k-1) + \mathrm{G}(k)[\mathrm{y}(k) - p(k \mid k-1)]$ minimizes the predict error covariance $\Phi(k \mid k) = E[(p(k) - p(k \mid k))(p(k)p(k \mid k))^T]$. Compared with conventional Kalman filter applications, the problem of Kelly's control system is its non-linear character, which is going to be shown in following paragraphs.

In [63], the discrete Kelly's rate control function is:

$$r(k+1) = r(k) + \alpha - \beta r(k)p(k) \qquad (\mathrm{k} = 0, 1, 2, \ldots) \qquad \text{---(A.1)}$$

In (A.1), *r(k)* is the sending rate for a media stream. α and β are constant coefficients, and *p(k)* is the loss rate at time step *k*. If the following assumptions hold true:

- Fairness among traffic is achieved in intermediate network routers.

- Loss rate is calculated by the amount of traffic exceed router's capacity / total traffic.

For any time instance k, we have:

$$p(k) = \frac{Nr(k) - C}{Nr(k)} \qquad \text{---(A.2)}$$

$$p(k+1) = \frac{Nr(k+1) - C}{Nr(k+1)} \qquad \text{---(A.3)}$$

N is the number of client. *C* is the system capacity.

Combining (A.2) and (A.3), canceling the *C* and *N*, we get the relation of *p(k+1)* and *p(k)* is:

$$p(k+1) = 1 - \frac{r(k)(1 - p(k))}{r(k+1)}$$

Replace *r(k+1)* in the above function using (A.1):

$$p(k+1) = 1 - \frac{r(k)(1 - p(k))}{r(k) + \alpha - \beta r(k) p(k)}$$

So the state and measurement functions for Kalman filter are:

$$p(k+1) = 1 - \frac{r(k)(1 - p(k))}{r(k) + \alpha - \beta r(k) p(k)} \qquad \text{---(A.4)}$$

$$y(k) = p(k) + n(k) \qquad \text{---(A.5)}$$

From (A.4), state function for loss rate is non-linear. It can be solved by the extended Kalman filter or unscented Kalman filter (UKF). Since (A.4) is not too complex to be linearized, we choose the simpler EKF for the solution. The main drawback of stability problem of EKF is omitted here because the rate control performs at small enough time interval, usually once every packet or at most once several packets. We use f to represent the function (A.4). The *n(k)* is the Gaussian white network noise with zero mean and covariance *R(k)*.

The (A.4) is linearized as:

$$p(k+1) = \nabla f \mid_{p(k|k)} \; p(k) + f(p(k \mid k)) - \nabla f \mid_{p(k|k)} \; p(k \mid k)$$

Where $\nabla f\big|_{p(k|k)} = \left.\frac{\partial p(k+1)}{\partial p(k)}\right|_{p(k|k)} = \frac{(1-\beta)r^2(k)+\alpha r(k)}{(r(k)+\alpha-\beta r(k)p(k))^2}$

Now the problem is changed to design a Kalman filter for a linear system. The detailed deduction of Kalman gain is shown in Appendix C and therefore omitted here. The implementation algorithm is given below.

Appendix B: Deduction of DLQ Control Formula (6.5), (6.6), (6.7)

Function (6.3) is rewritten as:

$$p_i x_i^2 + 2b_i x_i + c_i = x_i^2 + u_i^{*2} + p_{i+1} x_{i+1}^2 + 2b_{i+1} x_{i+1} + c_{i+1}$$

Substitute state function $x(i+1) = x(i) + t_s u(i) - t_s v(i)$ into it:

$$p_i x_i^2 + 2b_i x_i + c_i = x_i^2 + u_i^{*2} + p_{i+1}(x_i + t_s u_i^* - t_s v_i)^2 + 2b_{i+1}(x_i + t_s u_i^* - t_s v_i) + c_{i+1}$$

After manipulation:

$$\begin{aligned} p_i x_i^2 + 2b_i x_i + c_i = {} & (1 + p_{i+1}) x_i^2 + 2t_s p_{i+1} x_i u_i^* + (1 + t_s^2 p_{i+1}) u_i^{*2} \\ & + (2b_{i+1} - t_s p_{i+1} v_i) x_i + (2t_s b_{i+1} - t_s^2 p_{i+1} v_i) u_i^* \\ & + t_s^2 p_{i+1} v_i^2 - 2t_s b_{i+1} v_i + c_{i+1} \qquad \text{---(B*)} \end{aligned}$$

Now we need to replace the u_i^* with:

$$u_i^* = \frac{-t_s p_{i+1} x_i + t_s^2 p_{i+1} v_i - t_s b_{i+1}}{1 + t_s^2 p_{i+1}}$$

To make things clearer, we first neglect the terms without u_i^* in right hand side of (B*) and substitute u_i^* into $2t_s p_{i+1} x_i u_i^*$, $(1 + t_s^2 p_{i+1}) u_i^{*2}$, and $(2t_s b_{i+1} - t_s^2 p_{i+1} v_i) u_i^*$ respectively.

$$2t_s p_{i+1} x_i u_i^*$$

$$= 2t_s p_{i+1} x_i \frac{-t_s p_{i+1} x_i + t_s^2 p_{i+1} v_i - t_s b_{i+1}}{1 + t_s^2 p_{i+1}}$$

$$= -\frac{(2t_s^2 p_{i+1}^2)}{1 + t_s^2 p_{i+1}} x_i^2 + \frac{2t_s^3 p_{i+1}^2 v_i - 2t_s^2 p_{i+1} b_{i+1}}{1 + t_s^2 p_{i+1}} x_i \qquad \text{---(B.1)}$$

$$(1 + t_s^2 p_{i+1}) u_i^{*2}$$

$$= (1 + t_s^2 p_{i+1}) \frac{(-t_s p_{i+1} x_i + t_s^2 p_{i+1} v_i - t_s b_{i+1})^2}{(1 + t_s^2 p_{i+1})^2}$$

$$= \frac{t_s^2 p_{i+1}^2}{1+t_s^2 p_{i+1}} x_i^2 + \frac{2t_s^2 p_{i+1}(b_{i+1} - t_s p_{i+1} v_i)}{1+t_s^2 p_{i+1}} x_i + \frac{t_s^4 p_{i+1}^2 v_i^2 + t_s^2 b_{i+1}^2 - 2t_s^3 p_{i+1} b_{i+1} v_i}{1+t_s^2 p_{i+1}} \quad \text{---(B.2)}$$

$$(2t_s b_{i+1} - t_s^2 p_{i+1} v_i) u_i^*$$

$$= (2t_s b_{i+1} - t_s^2 p_{i+1} v_i) \frac{-t_s p_{i+1} x_i + t_s^2 p_{i+1} v_i - t_s b_{i+1}}{1+t_s^2 p_{i+1}}$$

$$= \frac{-(2t_s b_{i+1} - t_s^2 p_{i+1} v_i) t_s p_{i+1}}{1+t_s^2 p_{i+1}} x_i + \frac{(2t_s b_{i+1} - t_s^2 p_{i+1} v_i)(t_s^2 p_{i+1} v_i - t_s b_{i+1})}{1+t_s^2 p_{i+1}} \quad \text{---(B.3)}$$

Substitute (B.1), (B.2), (B.3) back into function (B*) and equalize the coefficients of x^2, x, and constant on both side of the equation (B*):

Quadratic terms in x:

$$p_i = 1 + p_{i+1} - \frac{t_s^2 p_{i+1}^2}{1+t_s^2 p_{i+1}} \Rightarrow p_i = 1 + \frac{p_{i+1}}{1+t_s^2 p_{i+1}} \quad \text{---(6.5)}$$

Linear terms in x:

$$2b_i = 2b_{i+1} - t_s p_{i+1} v_i + \frac{2t_s^3 p_{i+1}^2 v_i - 2t_s^2 p_{i+1} b_{i+1}}{1+t_s^2 p_{i+1}}$$

$$+ \frac{2t_s^2 p_{i+1}(b_{i+1} - t_s p_{i+1} v_i)}{1+t_s^2 p_{i+1}} - \frac{(2t_s b_{i+1} - t_s^2 p_{i+1} v_i) t_s p_{i+1}}{1+t_s^2 p_{i+1}}$$

$$\Rightarrow 2b_i = \frac{2b_{i+1} - t_s p_{i+1} v_i}{1+t_s^2 p_{i+1}} \quad \text{---(6.6)}$$

Terms independent of x:

$$c_i = \frac{t_s^4 p_{i+1}^2 v_i^2 + t_s^2 b_{i+1}^2 - 2t_s^3 p_{i+1} b_{i+1} v_i}{1+t_s^2 p_{i+1}} + \frac{(2t_s b_{i+1} - t_s^2 p_{i+1} v_i)(t_s^2 p_{i+1} v_i - t_s b_{i+1})}{1+t_s^2 p_{i+1}}$$
$$+ t_s^2 p_{i+1} v_i^2 - 2t_s b_{i+1} v_i + c_{i+1}$$

$$\Rightarrow c_i = t_s^2 p_{i+1} v_i^2 - \frac{t_s b_{i+1} v_i (2 + t_s^2 p_{i+1}) + t_s^2 b_{i+1}^2}{1+t_s^2 p_{i+1}} + c_{i+1} \quad \text{---(6.7)}$$

Appendix C: Kalman Filter for DLQ Scheduler

In the design of Kalman filter, we analyze the frame size distribution of a typical 766s MPEG2 video (shown in the figure C-1) and model the playback rate as a Gaussian distribution variable according to the result. In literature, MPEG2 video is commonly modeled using finite Markov chain for the short-term correlation and long-term dependency among frames. Considering the simplicity and efficiency, we approximately model the playback rate by normal distribution.

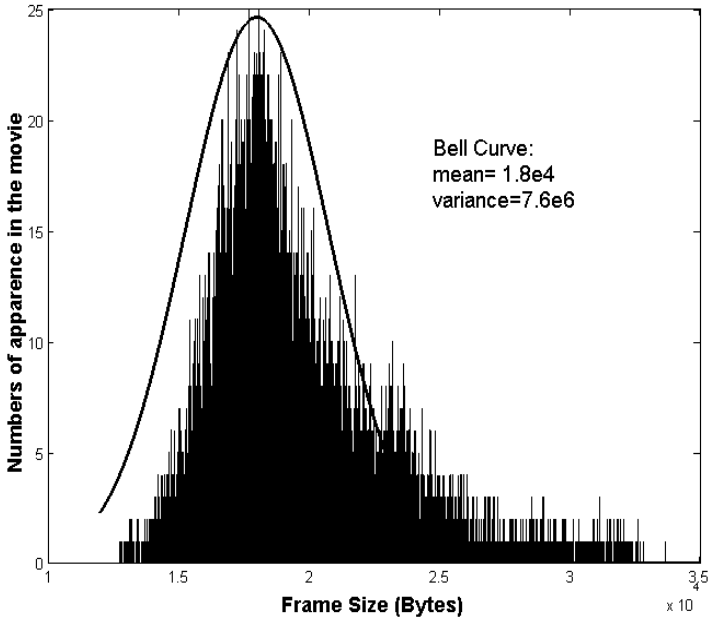


Figure C-1 Histogram of MPEG2 video frame size

A. Filter Problem Definition

$$x(i+1) = x(i) + t_s u(i) + m + w(i) \qquad i \geq 0 \qquad \text{---(C.1)}$$

$$y(i) = x(i) + n(i) \qquad \text{---(C.2)}$$

In state function (C.1), we divide the playback disturbance $t_s v(i)$ into average playback rate m plus a zero mean random variable $w(i)$. The accumulate effect of $m+w(i)$ performs the same as $t_s v(i)$ in initial state function (1). In observation function (C.2), $y(i)$ is the observed state

variable under network noise *n(i)*. Where {*n(i)*}, similar to{*w(i)*}, is a sequences of white Gaussian noise with zero mean comes from network. Their joint covariance matrix is:

$$E\left[\begin{pmatrix} w_i \\ n_i \end{pmatrix}\begin{pmatrix} w_i^T & n_i^T \end{pmatrix}\right] = \begin{bmatrix} Q_i & 0 \\ 0 & R_i \end{bmatrix} \qquad \text{---(C.3)}$$

Like EKF in Appendix A, the aim of Kalman Filter (figure C-2) in DLQ rate control system is to get the best estimation of state variable *x(i)* given previous predicted value of *x(i-1|i-1) x(i-2|i-2)…x(0|0)* and the observed current state variable value *y(i)*. The difference is that the Kalman Filter in DLQ system uses $x(i) + t_s u(i) + m$ as the a priori state estimate. Parameters for filter design and their meaning are listed in table C-1.

Variable Name	Meaning
i	Time step
y(i)	Measured value of state variable *x* at time step *i*
x(i)	Accurate value of state variable *x* at time step *i*
x(i|i-1)	A priori prediction of *x* before giving *y(i)*
x(i|i)	A posterior estimation of *x* given *y(i)*
G(i)	Filter gain at time step *i*
Φ(i|i)	Covariance of a priori prediction error
Φ(i|i-1)	Covariance of a posterior estimation error

Table C-1 Definition of variables in DLQ Kalman filter design

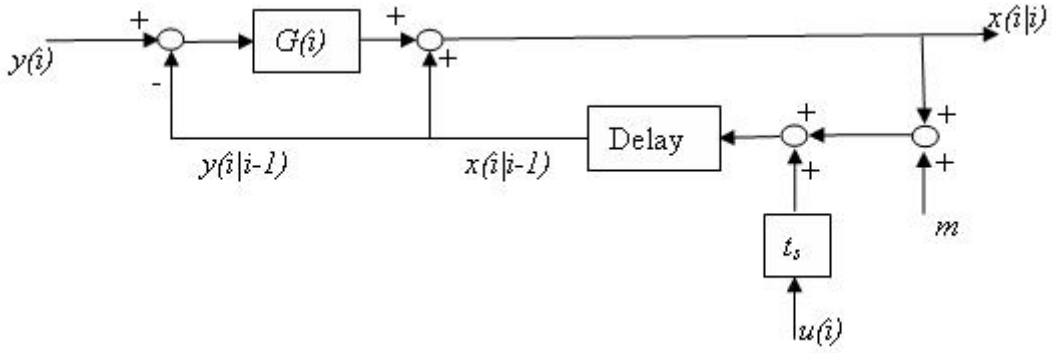


Figure C-2 DLQ Kalman filter block diagram

(2) The deduction of filter gain G

According to the filter problem definition, we have:

$$x(i|i) = x(i|i-1) + G(i)[y(i) - x(i|i-1)] \qquad \text{---(C.4)}$$

$$\Phi(i|i) = E[(x(i) - x(i|i))(x(i) - x(i|i))^T] \qquad \text{---(C.5)}$$

$$\Phi(i|i-1) = E[(x(i) - x(i|i-1))(x(i) - x(i|i-1))^T] \qquad \text{---(C.6)}$$

Now we are going to find $G(i)$ that minimize $\Phi(i|i)$.

Substitute (C.4) into (C.5)

$$\Phi(i|i) = E[(x(i) - x(i|i-1) - G(i)(y(i) - x(i|i-1)))^2]$$

$$= E[(x(i) - x(i|i-1))^2 + G^2(i)(y(i) - x(i|i-1))^2 - 2G(i)x^2(i|i-1)$$

$$+ 2G(i)y(i)(x(i|i-1) - x(i)) + 2G(i)x(i)x(i|i-1)] \qquad \text{---(C.7)}$$

Substitute (C.2) into (C.7) and let the derivative of $\Phi(i|i)$ with respect to $G(i)$ equals to 0:

$$\frac{\partial \Phi(i|i)}{\partial G(i)} = E[-2(x(i) - x(i|i-1))^2 + 2n(i)(x(i|i-1) - x(i))$$

$$+ 2G(i)(x(i) + n(i) - x(i|i-1))^2] = 0 \qquad \text{---(C.8)}$$

Because $\Phi(i|i-1) = E[(x(i) - x(i|i-1))(x(i) - x(i|i-1))^T] = E[(x(i) - x(i|i-1))^2]$

Substitute it into (C.8):

$$\frac{\partial \Phi(i|i)}{\partial G(i)} = E[-2\Phi(i|i-1) + 2n(i)(x(i|i-1) - x(i)) + 2G(i)(x(i) + n(i) - x(i|i-1))^2] = 0$$

$$\Rightarrow G(i) = E[\frac{\Phi(i|i-1) + n(i)(x(i) - x(i|i-1))}{\Phi(i|i-1) + n^2(i) + 2n(i)(x(i) - x(i|i-1))}]$$

$\because E[n(i)n^T(i)] = R(i)$ and the noise $n(i)$ is independent with prediction error $(x(i)-x(i|i-1))$, that is: $E[n(i)(x(i) - x(i|i-1))] = 0$

$$\therefore G(i) = E[\frac{\Phi(i|i-1)}{\Phi(i|i-1) + n^2(i)}] = \frac{\Phi(i|i-1)}{\Phi(i|i-1) + R(i)} \qquad \text{---(C.9)}$$

From the solution (C.9), *G(i)* only depends on the covariance of prediction error and the network noise, not the state function. Thus *G(i)* is ubiquitous suitable for linear system Kalman filters, together with those linearized system like the case in appendix A. Thus the following implementation algorithm is also similar to what has been given in appendix A.

Publications

Xiaofei.Zhou and Kenneth.Ong, "A Rule-based Intelligent Multimedia Streaming Server System," Journal of Mobile Multimedia (JMM) Special Issue on Multimedia Modeling and Applications, Vol.4 No.1 pp 019-041

Xiaofei.Zhou and Kenneth.Ong, **"**Discrete LQ Rate Control Scheduler for MPEG2 Video Transmission System," Journal of Multimedia, issue 2, 2008.

Xiaofei.Zhou and Kenneth.Ong, "A Rule-Based Expert Server System for Multimedia Transmission," International Conf. on Advanced Information Networking and Applications, May'07.

Xiaofei.Zhou and Kenneth.Ong, "Discrete LQ rate control schedule system for multimedia transmission," International Conf. on Advanced Information Networking and Applications, Vol. 1, Page(s):6,18-20 April 2006.

*Xiaofei.Zhou and Kenneth.Ong, "*Problems on Implementation of LQ Rate Control Schedule System for Multimedia Transmission*,"* Fourth International Conference on Intelligent Multimedia Computing and Networking, Salt Lake City, Utah, USA, July 21-26, 2005.