Journal of Mobile Multimedia, Vol.4, No.1 (2008) 019-041

A RULE-BASED INTELLIGENT MULTIMEDIA STREAMING SERVER SYSTEM

ZHOU XIAOFEI ONG KENNETH

National University of Singapore
{Xiaofei, eleongk }@nus.edu.sg

Received July 1, 2007
Revised October 15, 2007

In this paper, a novel multimedia transmission server is designed using rule based expert system technology. It is superior on current media transmission servers because it is more powerful on streaming, flexible on control, and reliable on maintenance. In the proposed expert server, working parameters and management methods are separated from the decision-making procedure and stored in an xml database called knowledge base. The server fires different transmission strategies inside the knowledge base by runtime inference. Thus it can easily adjust transmission parameters under various environments and adopt new developed methods without significantly changing the main body of server codes. In this paper, we not only analyzed the time complexity of inference procedure and the real time characteristics of the server, but also tested the server performance on local area networks. Results showed that the expert system can deliver smooth streams with around 50% deduced throughput oscillations when compared with a single rate control method. The saved 50% bandwidth could be used for supporting more users. When congestion happened, the expert system reacts intelligently and conducts cooperative steps to relocate its resources. At the same time, the congestion related information is recorded and referred for future congestion avoidance. Attractively, these enhanced performances are achieved by taking less than 10% of the CPU time for the execution of the expert server control program.

***Key words*:** Expert System, Multimedia, Streaming

1 Introduction

Expert system [1] [2] [3], also called knowledge based system, has developed rapidly within the family of Artificial Intelligence (AI). It simulates the procedure for a human expert to solve a problem based on on-hand data, past experience, and appropriate reasoning. Unlike the conventional computer systems that usually do rigidly defined routine work, expert system is mostly suitable to perform high level decision-making work, which is difficult to be completed by a simple algorithm. The solution it gives depend largely on the correctness of knowledge base, the precise of working parameters, the efficiency of inference procedure, and also on the time limitation for a real time system. Rather than given out a strictly optimal solution, expert system gives out a sub-optimal solution first and take it into consideration for further reasoning, getting closer and closer to the final decision. Even the final solution is not necessarily optimal, but it must be feasible and correct in a large probability. Thus expert systems are mostly used in fields like medical diagnose, mechanical and chemical process control, and financial service.

As early as 1988, AI researchers have put their attention on using the expert system on network control [4]. Later, the expert systems were used in network capacity planning [5].Nowadays, researchers turned their attentions to more specific areas like traffic prediction, task distribution [6], and active queue management [7]. However, all works focused on a broad domain and did not specify a target application. Thus the heuristic rules were hardly proved to be effective considering the variety of applications. As a result, the outcomes of early attempts to control the network applications or task

scheduling are lack of domain related significance and seem ambiguous on the problems they suit to solve. Due to this drawback, the inference procedures were not convincible since the rules in those systems were designed for multiple applications that may have different or even contradict requirements.

To analyze the suitability of applying the expert system into media streaming, we first consider the streaming servers nowadays. They need to handle two types of work for a successful transmission. One type is the routine work, like receiving and analyzing client requests, sending media data based on various protocols, and maintaining session states. The second type is control work like session parameters adjusting, schedule strategy selection, congestion response, etc. These control work can be solidly coded similar to the routine work, alternatively they can be extracted from the major part of server program to form a separate supporting database. Server selects the proper method to use according to characteristics of problems and runtime parameters. Hence expert system is suitable to video server system as a global control mechanism, especially for distributed server clusters. Regrettably, still no research has been done in this area until now. Therefore, we provide our creative work of applying the rule based expert system for the video transmission server in this paper to make the server smart on decision making and self-learning.

The rest of the paper is organized as follows. Section 2 gives the network topology and general structure for the expert system. Section 3 presents the detailed design method. Theoretical analysis of the expert system is provided in section 4. In section 5, we gives out a case study and the corresponding results are shown in section 6. Section 7 summarizes the whole paper.

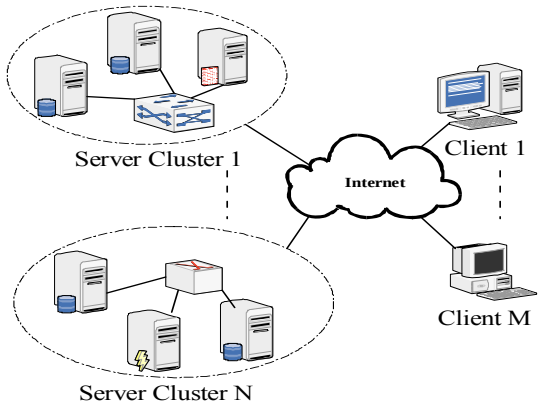


Figure 1 Expert Server System Network Topology

2 Expert system structure

To deliver multimedia streams smoothly, servers should be arranged in a suitable network topology and communicate with each other efficiently. In this section, we introduce the distributed expert servers system and the general structure diagram of an expert server.

2.1 Network Topology

Servers locate at a place is grouped into a server cluster (figure 1). Each server has its own decision making mechanism. That is, each server is a small expert system. Working parameters are periodically

broadcasted among servers in the same cluster and also among clusters. 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.

2.2 System Structure

As a rule based expert system, the server is mainly divided into four parts: knowledge base, working memory, and inference engine. The structure is shown in Figure 2. Rule base and method base are stored before hand in the server as reference database. Methods are the algorithms or functions to perform actions like distributed server allocation, buffer management, scheduling, and congestion response. Rules are IF-THEN clauses used to monitor the whole system, adjust parameter values, and also decide which method to choose under different situation. When new requests or tasks come, inference engine searches the rule base. If the condition of a rule is satisfied, it will modify the parameter in working memory or select a method to execute when necessary, or sometimes it will direct the inference engine to check other related rules. We will introduce each part in detail in the following paragraphs.

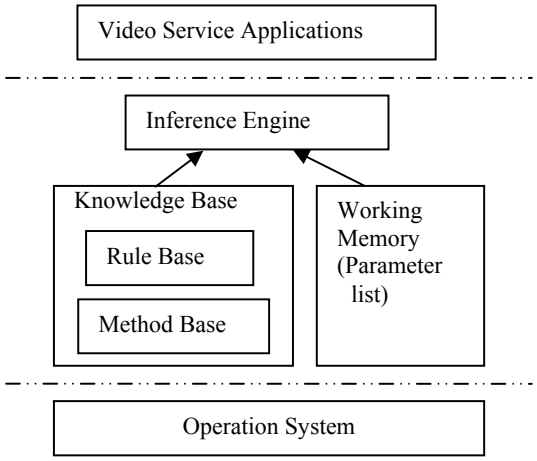


Figure 2 Rule base system structure

- Knowledge Base

An individual rule is divided into two parts: conditions (premises) and actions (decision part of rules). Action part is divided into two types: rule call (call other rules) and function call (execute a function to modify parameters or handle the streams). That is:

IF *Conditions* THEN *Actions (Rule call/ Function call)*

The functions are stored in method base, which is also a part of the knowledge base. Rules are written in xml language and stored in a separate file for the flexibility of extension. When the system starts execution, xml rule parser first reads rules from the rule-base file and translates them into codes that recognizable by other parts of the program. Inference engine does not refer to method base directly; instead, it calls methods through firing the rules. Since the construction of knowledge base is crucial to the search efficiency and system performance, we will introduce it in detail in Section 3.

- Inference Engine

Inference engine fetches rules and functions from knowledge base, use them together with the runtime parameters stored in working memory to deduce an either optimal or at least feasible solution for a request or a task. It starts the search from meta-rules. Meta rules direct inference engine to the group of rules that match the problem. If the conditions satisfied for a rule, its action part will be fired. If any rule is fired in a round of search, inference engine needs to search the same group again for fear that modified parameters may make previously unsatisfied conditions become true. To preventing loop, those rules that already fired will not be taking into consideration again. In conventional expert system, there are forward chaining and backward chaining for inference. Forward chaining method starts inference from a set of selected facts and eventually makes decisions for the problem. It is suitable to derive solutions when data are available. Backward chaining is used for diagnostic system where hypothesis (goal) is available and to be tested by the inference engine. Video transmission server is a plan system that parameters can be tested or fetched through network, not a diagnostic system. Therefore we only adopt the forward chaining for our inference engine.

- Working Memory

Working memory is used to record on-the-fly parameters like actual CPU and BW usage, measured QoS, network environment, and the client situation. Each parameter is given a unique number and listed in a lookup table. It provides supports for inference engine on decision making. It can only be modified by real time monitor module or rules.

3 Server design

In this section, the detailed server structure is provided and explained. After that, we will introduce the XML implementation of knowledge base and the corresponding inference procedures. The communication mechanism among modules of expert servers is shown with a diagram in the last sub-section.

3.1 Framework of Expert Server

Based on the introduction about expert system in section II, we design the video streaming server system module structure as shown in figure 3. In the left top corner, 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. Furthermore, monitor records the current resource situation and calculate some statistical parameters to manage sessions.

Master control is an independent request classification routine. It receives requests from the monitor, 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 frequently, the QoS management module will be called to handle the problem.

The session establishment module, one level down from the master control, is called by the master control procedure. It is used to perform admission control and establish a new session. If the movie does not locate on the current station or the load of the current station is too high to serve more clients, the session establishment 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.

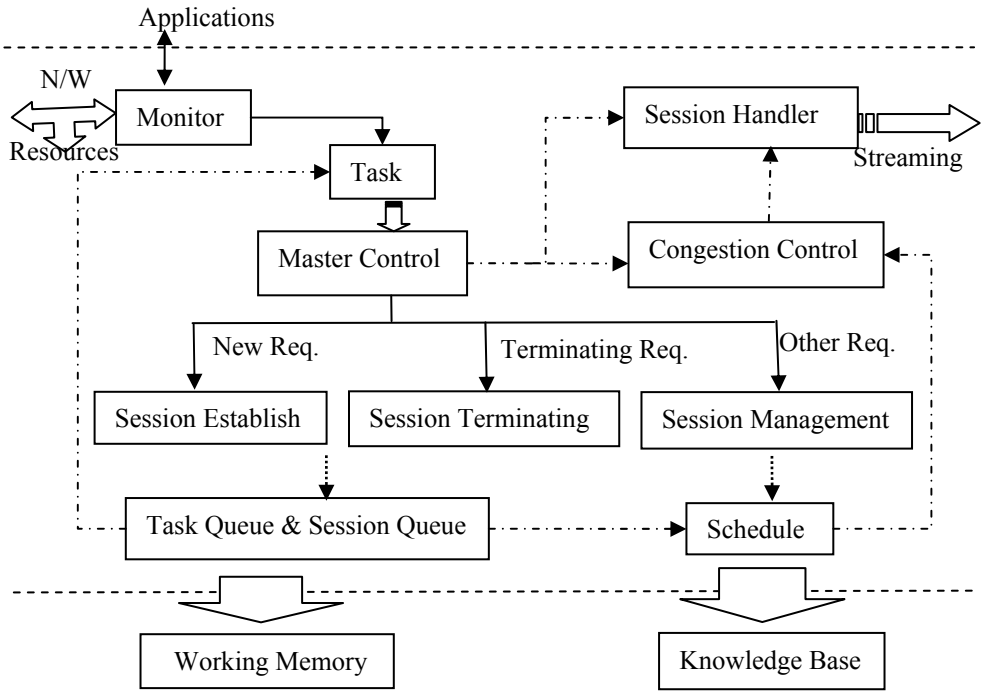


Figure 3 Expert system modules structure

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 resource shortage motivated termination, 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.

a) ***Meta rules.*** These are special rules used to make upper level decision or to decide which group of rules is the starting point of searching. It also contains rules using to decide the search method (e.g. Y in the following example) based on time constraints.

   Example: IF *New Subscription Request* THEN *Search Session establish/termination rules.*

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

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

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 a Suitable Congestion Control Scheme*

With the above group segmentation, the inference engine does not need to search all rules every time. It starts from the meta-rules and searches only the request-related rule group according to the decision of meta-rules.

3.2 Rule base implementation

Our rule base is realized using XML files. A typical XML document contains markup and character data. The markup contains the meaning, such as “movie name”, and is held in tags and other XML elements. The character data is the content. The rule structure, as introduced in sub-section 2.2, consists of condition part and decision part. Two types of decisions could be made by a rule. That is, function call or rule call. For consistency, we use functions to modify or check the real-time value of server parameters, although these parameters are accessible directly by rules. Each rule has a unique rule number and a type indicating which group the rule belongs to. The parameters referenced by rules or functions called by rules also have their unique identity number. 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. In our implementation, we allow the function call in passed parameters. Here is an example rule.

```
<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 the 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. At the beginning of this rule, the rule number 78 and the rule type 'Monitor' is given. When server loads the rule base, this rule will be automatically put into Monitor-rules group. At the first line of condition part, a function named GetSePara is called to check a session's status value. This is the only parameter that going to be passed to GetSePara, so the 'para' link has only one element. This parameter is given by a string of its name 'Status'. If condition satisfied, the function TaskGenerator is called to generate a task to end the unresponsive session.

3.3 Decision making procedure

Decisions are made by either following a routine procedure or searching the rule base to form an optimal searching tree. Now we use a potion of memory allocation rules to illustrate the basic decision tree (Figure 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 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 decided to be moderate, then 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 session handler and working memory. After that, search process goes on to perform other resource allocations by repeating the same search algorithm.

The tree is set up automatically during the inference process. For each rectangle, the search of its child nodes terminates whenever a solution (circular rectangle) is reached. We call it First-Match decision making. After all parameters in session handler 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. 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.

Due to unique characteristics of media streaming transmission, the knowledge base and the inference procedure of such an expert server are 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 abundant 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 previous buffer allocation example, the buffer allocated depends largely on the disk speed 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 comparing to conventional expert system applications.

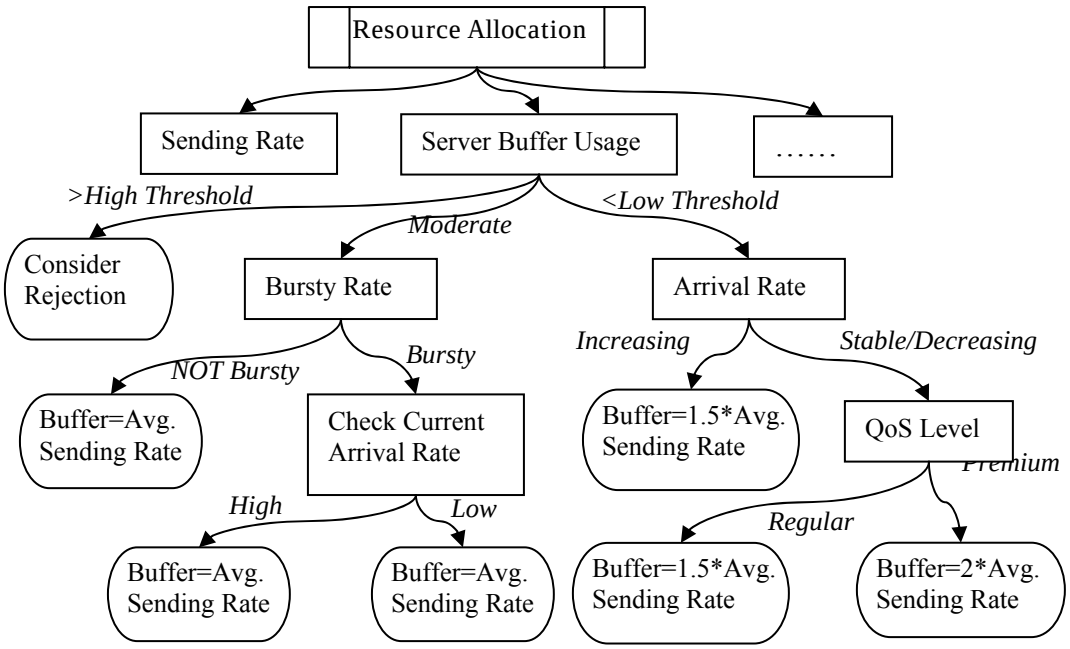


Figure 5 Decision tree for buffer allocation

It may be argue that since the types of resources in media expert server are clearly defined and the links among parameters are close, it may be better to pre-coding all IF-THEN clauses into server programs. This distrust could be eliminated from three aspects.

Firstly, some heuristics are difficult to be mathematically modeled and sequentially coded, although they are practically helpful under the heterogeneous network environment. Additionally, the set of heuristics needed for a decision is not always the same. The intelligent inference performed by the expert control can not be completed by merely going through a list of possible solutions in sequence. For these reasons, more effective way would be coding heuristics as rules separated from the main program.

Secondly, the organization of the expert system enables great flexibility on knowledge base updating and system maintenance. Compared with traditional applications where problem-solving algorithms are encoded in programs, the expert system encodes all problem related expertise in data structures only; none is in programs.

Lastly, expert server is feasible considering the performance improvement and the overhead. Compared to pre-coded search, the only searching overhead brought from using knowledge base comes from the translations of parameters from a parameter number into its actual value in working memory. Since the knowledge base written in XML is parsed and linked as a database with the main

server program beforehand, searching it would not require much more time than search IF-THEN clauses pre-coded in the server program.

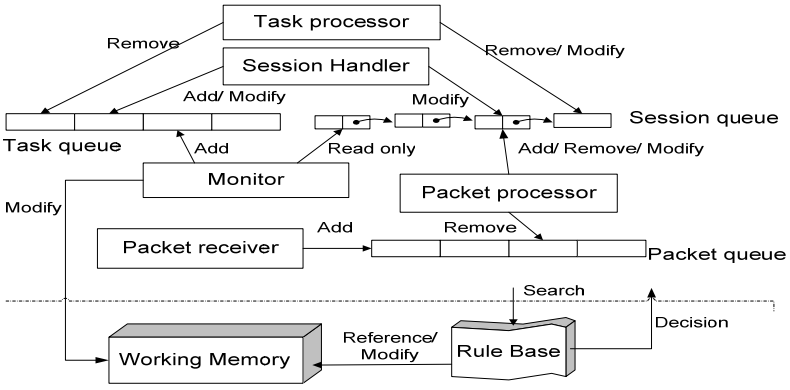


Figure 6 Communication Diagram of an Expert Server

3.4 Communications of Modules

Figure 6 is the relations among processes in the server program. We set up a packet queue for receiving requests from network and clients; a task queue for information from current server station; a session link to manage active sessions on current station. Five processes, task processor, packet processor, session handler, monitor, packet receiver, will work on these three queues as demonstrated in the figure; obviously semaphores are needed on each queue for mutual exclusion. All global runtime parameters and resource tables are stores in working memory. Rule base is off-line edited. The line connect from rule base to the working memory means that rules will execute the decisions it made without inform the upper level processor that access the rule base.

4 Analysis of Performance

In this chapter, we will analyze the computational complexity and real time characteristics of an individual expert server, followed by an estimation of the server overall capacity.

4.1 Complexity and Computation Time

To balance the computational overhead and system accuracy, the master control program of expert system is scheduled to executing every 100ms. The complexity of server program can be divided into two parts. One is decision-making process used to response to requests and tasks; the other is rule base searching process. In execution, rule-base searching process is embedded into the decision-making process. That is, decisions are made by searching the rule base. If taking the searching process as a single statement, the decision making process is run in a sequential manner. The complexity is O(n), where n is the session number. As a result, the main execution complexity for expert system program comes from searching the rule base. Now we focus on calculating the complexity of rule-base searching process.

The searching process is complex given that rules are organized as multi-branch tree. Additionally, the fire of one rule may cause the condition of another rule is satisfied. Then another rule will be fired in chain. Under this case, several rounds of searches are needed for an inference on the rule base. Here we give out the worst case analysis for two typical conditions. 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 an inference. The fired rule, 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 thus cause the r3 is satisfied and ready to fire. As a result of firing r3, the r2 is satisfied, and similarly r1 is ready to fire after firing r2. The final situation is all rules are fired after **n** iterations of searching. Of course C2 is nearly impossible to happen during real execution because many rules contradict to each other. Those rules cannot be fired together for the same problem. Here we use C2 as the worst case bound. In practice, the searching time should be close to the computation time of C1.

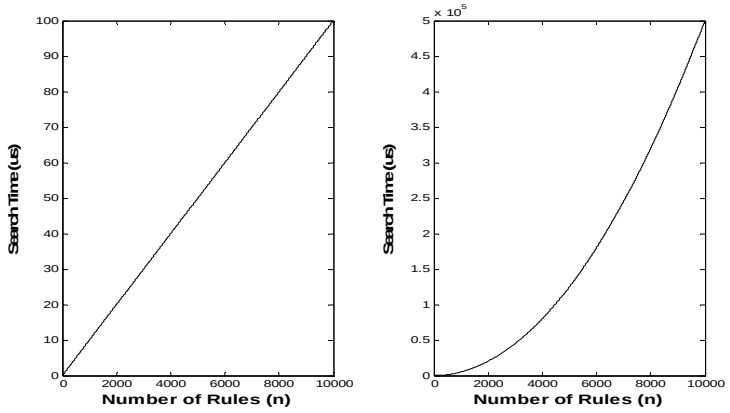


Figure 7 Worst case searching time for C1 and C2

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 7, 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 curse of C1 if rule base is properly organized. We will discuss the influence of complexity to system performance in next subsection, considering the real time characteristics.

4.2 Real time characteristics

In media server, decisions are made within a specific deadline. For example, a new request should be served within one minute; otherwise the client will complain or cancel the request. A feedback packet should be finished processing within feedback intervals. Those are the real time characteristics of the expert system. But for a media server, most CPU time should be used for transmission.

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 50 decisions for each round of control, and every decision must be made by inference on the rule base. The overall time taken is 5010us. Considering the monitor interval of 100ms, the time portion for making decisions is 5010/100,000 = 5.01%. This is the best case mean service time value and the calculation does not consider the 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 the following formula.

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

Another important real time characteristic is that the server system can not break down at any time. Here we are not considering the physical breakdown since it is very unlikely for a distributed system encountering physical breakdown of all stations simultaneously. However, if rules are not designed properly and organized carefully, the system might breakdown due to rules collisions. In our system, if collision happens and no meta-rules aim at solving it, the expert server will gives out error message and take the default value been set offline beforehand. If the collision is critical, the server will terminate itself after transferring current under service sessions to other stations.

4.3 Capacity of a Single Server

Based on the analysis of computational complexity and real time characteristics of expert system, capacity of a single server can be derived now. Suppose monitor interval is set properly as illustrated in section 4.2, the capacity bottleneck of the server would be the outgoing bandwidth. We have the following formula for calculating approximate capacity.

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

According to the above formula, the server capacity depends largely 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.

5 Case Study

The performance of expert server system depends largely on the efficiency of rules and methods. Therefore it is difficult to give uniform experimental results without a standard knowledge base. Here we provide a case study to demonstrate the performance of expert server system.

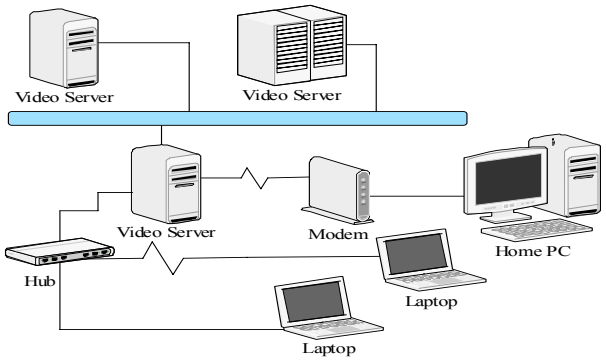


Figure 8 Experimental Topology of Expert Server System

5.1 Experiment setup

We set up the experimental server network as shown in figure 8. Servers communicate through the bus. Laptops and PC act as clients. More than one session is set up between the server and a client. Statistic data is gathered at the video server connects to modem and hub in the figure.

5.2 Buffer management methods

We choose prioritized-RED and layered drop as the buffer management method. RED [8] 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.

5.3 Scheduling methods

Three scheduling methods are implemented in expert server. They are round robin (RR), weighted round robin (WRR) [9], and priority queuing. In RR, all sessions are treated equally and are allocated in average the same portion of bandwidth. It is suitable to best effort transmissions. WRR gives each session a unique weight number according the client parameters and demanded QoS level. The higher the weight, the more bandwidth is allocated to the session. In expert system, the weight is set during session setup by the *session establish* rules and maintained during execution by the *QoS management* rules and *congestion control* rules. Priority queuing method divides sessions into premier group and normal group. Certain amount of bandwidth is reserved for sessions in premier group to guarantee their QoS. On the contrary, the normal sessions receive only best effort service.

It should be noted that the expert system decide more than merely selecting a scheduling algorithm for sessions. It creates smart combinations of those algorithms to make the transmission efficiently. For example we can serve the premier sessions in a RR way for fairness while serve the normal sessions 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.

5.4 Congestion control methods

We select four congestion control methods for our case study. They are DLQ [10], TFRC [11], HSTCP [12, 13], EMKC [14,15,16,17,18], basic AIMD [19]. The brief comparison of these methods is given in table 1. The variable p in the table means loss rate.

TFRC, HSTCP and AIMD are window based congestion control while DLQ and EMKC are rate based. Notice that most provided congestion control methods in table 1 take loss rate p as the input, especially for EMKC, who take p as the only parameter of the environment. Usually, the loss rate carried by feedback packets can not be precisely calculated due to measurement noise and are often delayed by intermediate network. Therefore we add the Kalman Filter [20] in expert system to predict the p value. Limited by the length of this paper, we move the design of Kalman Filter into appendix A and only give out the simulation result for EMKC method in figure 9 since the main purpose of this paper is not design or improve the congestion control algorithms. The EMKC with kalman filter is called EMKC_KF.

	Response Function	Advantages	Disadvantages
TFRC	$\frac{s}{RTT\sqrt{2p/3}+3RTO\sqrt{3p/8}\,p(1+32p^2)}$	Satisfactory performance for most conditions	Worse than other improved methods on high speed N/W
HSTCP	$\frac{S*C}{RTT*p^d}$	Suitable to High Speed N/W.	Performs worse than TFRC with high p.
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 implement	Not considering the intermediate N/W 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.

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

Table 1 Comparison of Congestion Control Methods used in Expert Server System

From the figure 9, EMKC_KF is more stable than EMKC under violated noise and delay. In the implementation, we use EMKC_KF instead of EMKC.

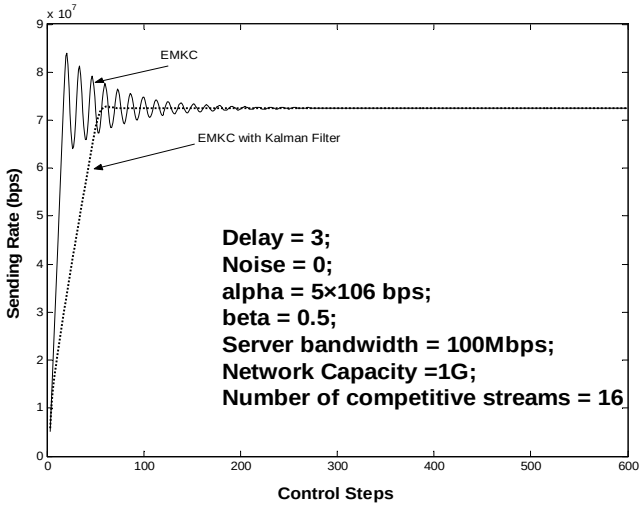


Figure 9 Performance of EMKC_KF

To take the advantages of each method and avoid their shortcomings, the expert system selects different algorithm according to different network condition. If high speed network is available with low loss rate p, HSTCP could be selected. With the increase of p, HSTCP performs worse than TFRC. Thus congestion control task is switched to TFRC. 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 take over the control. In case of wireless application where sudden change of environment happens occasionally, AIMD is a suitable way to guarantee the quality of transmission. All these methods do not consider the receiver's capability. DLQ is responsive to make up the gap. The expert system keeps on calculating the optimal sending rate according to client buffer occupancy using DLQ method and comparing it with the sending rate decided by other method. If possible overflow detected, the sending rate is capped to the value get from DLQ. Similarly, if potential underflow problem exits, sending rate is raised to the value calculated by DLQ and simultaneously the prioritized-RED is turned on to discard some less important packets.

At the beginning of the transmission, the expert system uses only 60% of the maximum client buffer allocated for media receiving. During the transmission, it traces the usage of client buffer from client feedback packets. If underflow happens frequently, the expert system may increase the used client buffer 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%.

6 Results and Discussions

Considering throughput and low-jitter are the main criteria for media server, we compare the results of expert system with TFRC, which is a most widely used conventional method on rate control.

6.1 Throughput variations for a single flow

Client side throughput under TFRC or expert system control is shown in figure 11. The results were gotten under the settings that network loss rate ranges from 0 to 40% with average value 0.5% (figure 10, left), and the RTT randomly distributed between 4ms to 20ms (figure 10, right). Figure 11 showed the throughput under TFRC control (dotted line with circle marker) fluctuates larger than that under expert system control (line with plus marker). This phenomenon happened because the two methods have different abilities to adapt their sending rate under fluctuating environment. TFRC calculates sending rate only according to the loss rate and average RTT, while expert system also taking into consideration the traffic demand from client. When loss rate or RTT changes, that is, environment changes, TFRC hurriedly changes its sending rate according to the new loss rate and RTT values. Expert system, on the other hand, checks 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 buffer management strategy simultaneously, instead of changing the sending rate, in response to the changes of loss rate. Through this way, the client can receive a smoother media flow with less buffer overflow or underflow problems.

Besides the throughput, another important criteria need to be considered for media transmission is the client buffer occupancy (figure 12). As introduced in section V, expert system uses 60% of total allocated client buffer size. That is, 0.6MB in figure 12 although the allocated client buffer size is 1MB. It is obvious that the client buffer occupancy is quite smooth with expert system control while

the client buffer frequently encountered underflow under TFRC control although the throughput of TFRC is comparable with that of expert system (figure 11). This is because TFRC method adjusts sending rate according to loss rate without taking care of the client buffer usage. Even if underflow happens frequently, TFRC does not consider it as long as the intermediate loss rate does not change. On the other hand, expert system can detect such problem and adjust accordingly.

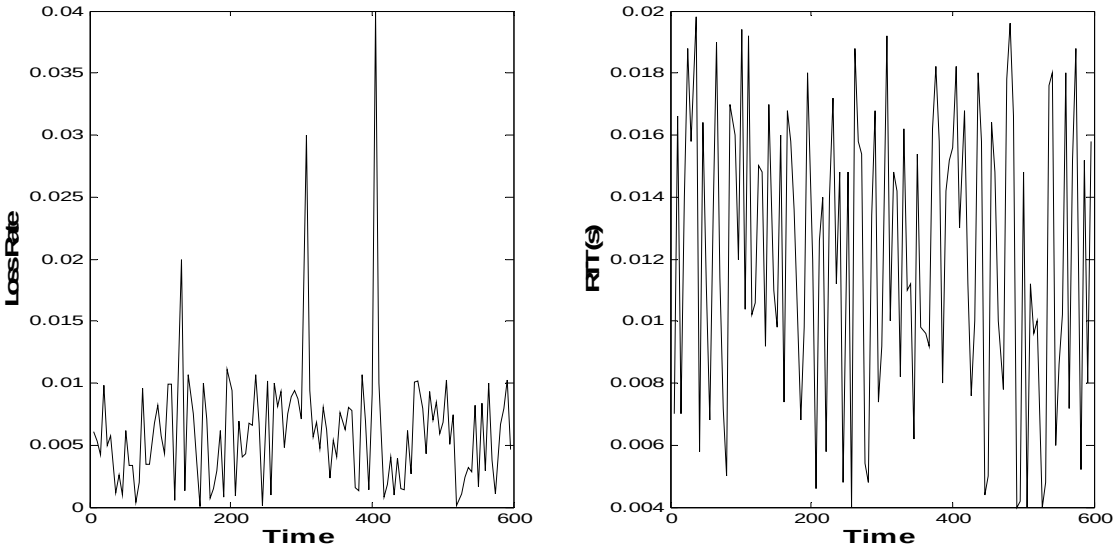


Figure 10 Loss rate and RTT characteristics

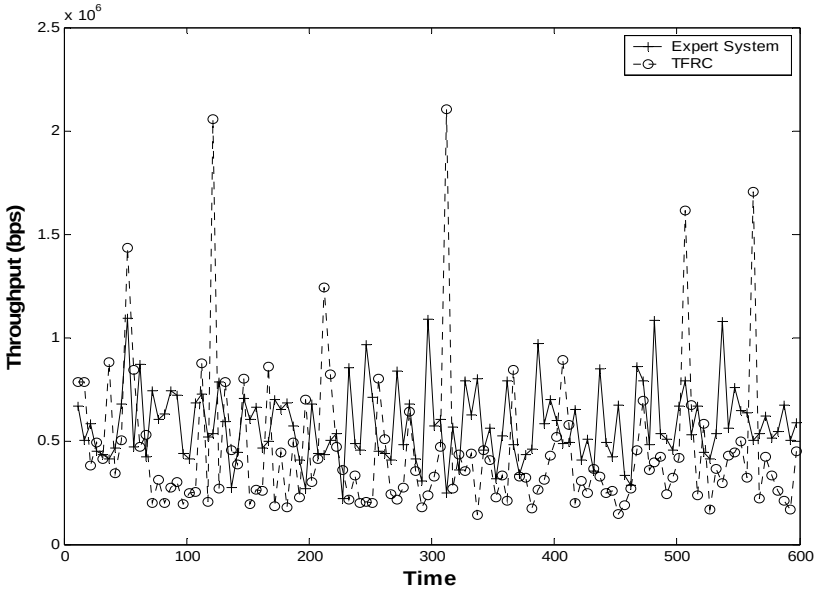


Figure 11 Client-side Throughput under TFRC or Expert System Control

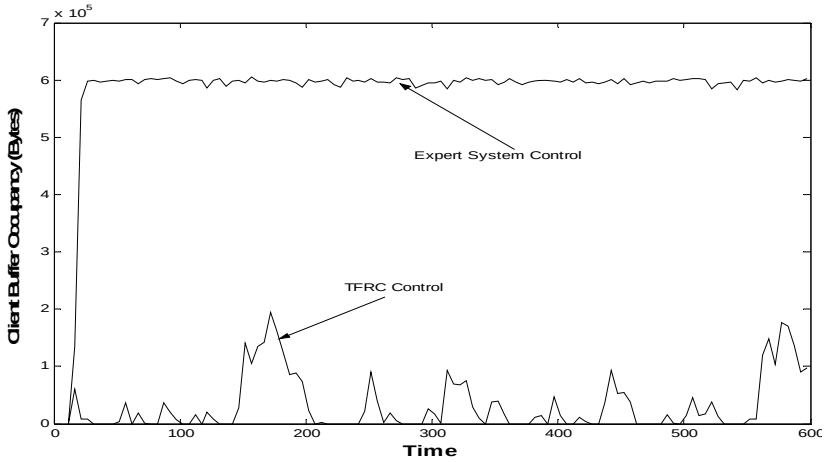


Figure 12 Client buffer occupancy under TFRC and Expert System control

6.2 Bandwidth probe

We now consider a situation with suddenly decrease of bandwidth. Usually, the reaction sequence for this case is like this: bandwidth decrease brings congestion; congestion causes higher loss rate; finally all edge users must decrease their sending rate greatly. In this subsection, we will see the reaction of expert system. When the loss rate p increases suddenly from around 0.4% to 15%~40% between 150 and 300 (Figure 13), the server detects the change and treats it as an indication of congestion or a sudden decrease of available bandwidth under both TFRC and the expert system control (figure 14). The following reactions differ for the TFRC and the expert system. The TFRC calculates a new sending rate from updated loss rate information. Obviously the rate decreases sharply in the figure. When bandwidth recovered, its sending rate was also grown up to the level before congestion like nothing had happened. In a word, the TFRC is a memoryless method that makes decision only based on the current loss rate. The expert system, on the contrary, checked other parameters first to confirm the occurrence of congestion. It suspected the truth of this phenomenon when it finds out that RTT remains stable and there is no historical sudden changes happened along this route. Actually the loss rate was changed by us manually to fake the congestion but the server did not know it. Due to the suspicion, the expert system reacted quite different with the TFRC. Instead of severely decreased the sending rate, it adjusted the sending rate to be more stable at the average level. Meanwhile, it switched on the buffer management mechanisms to diminish the traffic demand. These actions were taken to give some time to the intermediate networks to relieve the congestion. If the loss rate stays at a high level for a period, the expert system would consider to decreasing the sending rate further. Actually at the sample point 300, the expert server already started to decrease its sending rate because of the duration of high loss rate. However at this time, the network situation recovered to normal. Consequently, the expert system raised the sending rate but a little bit lower than the rate before congestion. This is because the congestion just happened has been remembered by the statistical log and the expert system would be careful on future adjustment due to this history until this record is discarded due to time out. Now let us examine the discard rate of the expert server buffer during this procedure.

Figure 15 gives the discard rate of server buffer along the transmission. When loss rate excess the threshold of p_low, prioritized-RED is switched on and layered drop is started. Expert system picks out the less important packets like B frames in MPEG2 and discards it according to discard probability calculated by RED. The rest more important packets are sent out with a lower sending rate derived

from congestion control algorithms. Benefit from the cooperation of buffer management and congestion control mechanism, expert system holds the client-side throughput at a reasonable stable level.

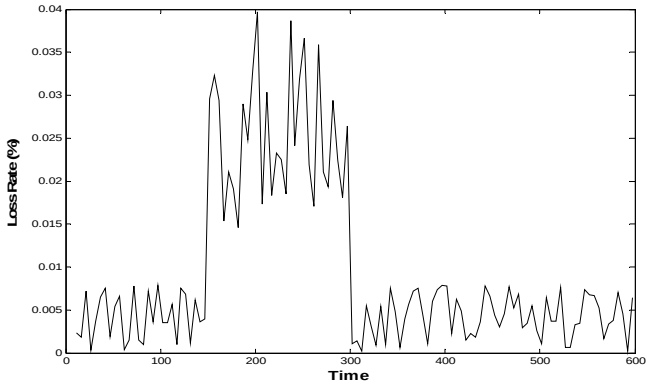


Figure 13 Available BW suddenly decrease (loss rate increases)

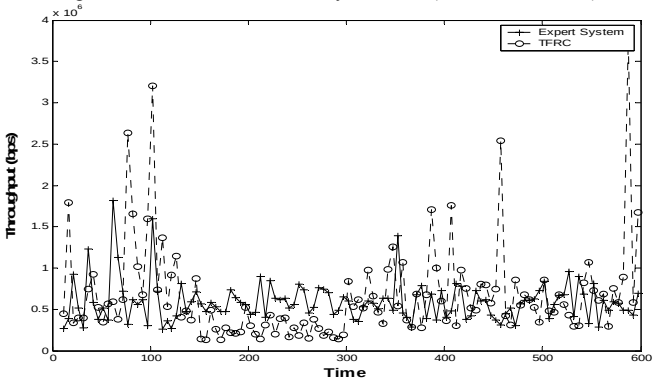


Figure 14 Throughput under TFRC and Expert System control when BW suddenly decreases

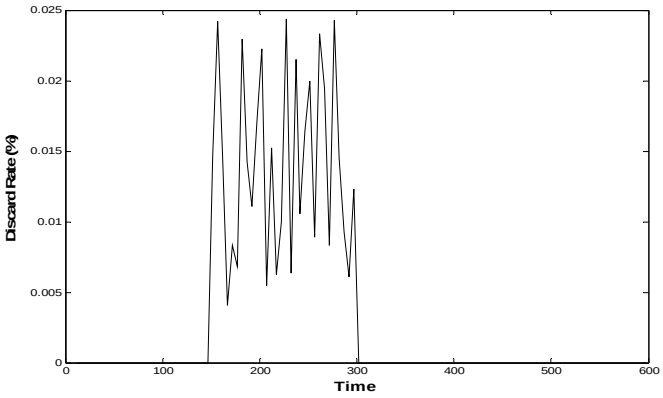


Figure 15 Discard rate on server buffer when BW suddenly decreases

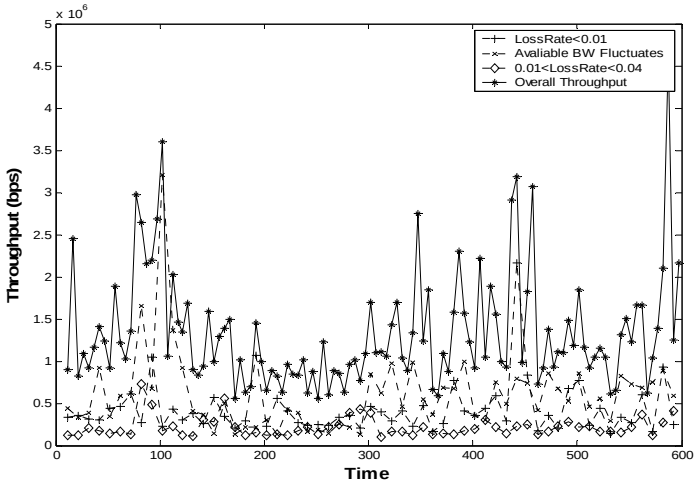


Figure 16 Throughput for three streams under TFRC control

6.3 Throughput for combined traffic

Expert system has the flexibility to carry on per flow control. That is, it treats each session uniquely and allocates proper buffer management and congestion control methods to each one according to the run time parameters associated with the session. Figure 16 and figure 17 represent the throughput for three traffics with different properties under TFRC and expert system control respectively. The overall throughput with TFRC control has a mean value of nearly 1.5Mbps but fluctuates from 0.5Mbps to 4.7Mbps (figure 16).

Comparatively, expert system achieves average 1.75Mbps throughput rages from 1.2Mbps to 2.5Mbps (figure 17). The results represent clearly that expert system performs better on global control than a single session control method. Comparing with conventional server, the per session management mechanism adopted in it enhance the overall performance of transmission. Additionally, benefit from the small fluctuation of throughput achieved, the expert system can send just enough data for each flow, saving the bandwidth for more users.

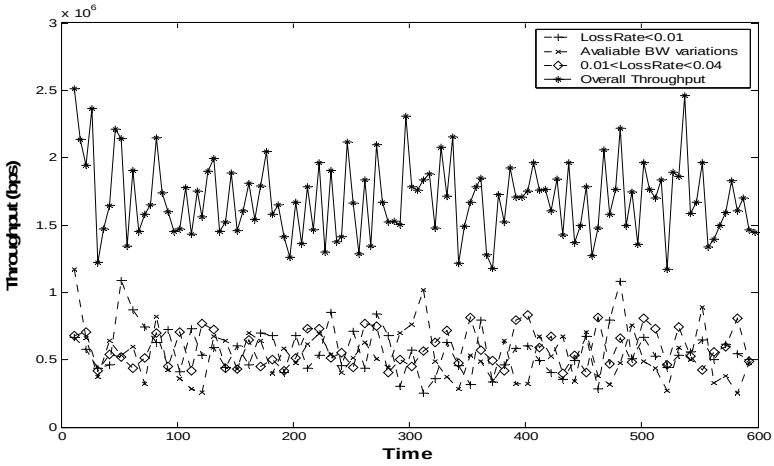


Figure 17 Throughput for three streams under expert system control

6.4 Server resource utilization

Previous results concentrate on sending rate and the client side parameters. Now we come back to the parameters of server itself. Since bandwidth utilization is already discussed together with sending rate in previous paragraphs, here we only investigate the CPU utilization and server buffer usage of the expert system (figure 18 and figure 19). In our experiment, the whole server is devoted into media streaming service without other type of background applications. The figure 18 is obtained by setting new request arrival rate as 0.15 per second under Poisson distribution. We change rule base size from 20 to 2000 rules and recording the CPU time for transmission. Of course the 2000 rules are not all meaningful. Quite a number of them are dummy rules are just added for testing the CPU time occupied by expert system. From the figure, the size of rule base does not influent the CPU time for transmission much. Generally, 90% of the CPU time is allocated for media delivery, receiving feedbacks and performing transmission. Less than 10% of CPU is used for expert server control.

Another phenomenon observable from the figure is that before time 150 when the number of sessions under service is less than 25, the CPU time for transmission is nearly identical with different rule size. It means that when rule size increases, the expert system takes more CPU time only under heavy load. The explanation lay on the per session control mechanism adopted by expert system. The more session accepted; the more efforts should be put on taking care of them.

As for the server buffer allocation, the expert system allocates buffers for each flow depending on the amount of data that going to be transmitted in next time slot. Notice the buffer allocation for the server is different from that for intermediate router as the amount of incoming packets for a router is not controllable. For media server, it can determine how much data should be moved from the database to its buffer according calculated sending rate. The usage of our expert system is shown in figure 19.

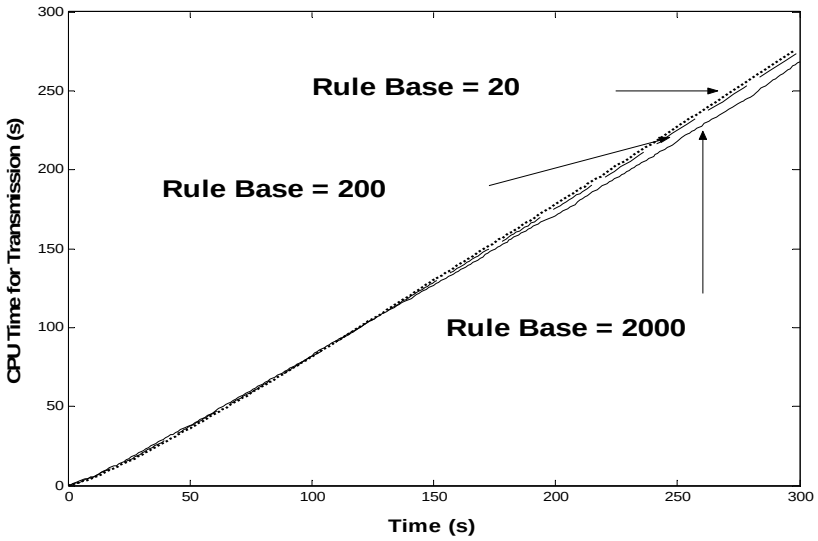


Figure 18 CPU Utilization with Different Size of Rule Base

Obviously, server under TFRC control need at least a 110KB buffer to cater the data for the peak sending rate while a smaller buffer of 70KB is enough for expert system. With the expert control, the space saved is up to 57.1%.

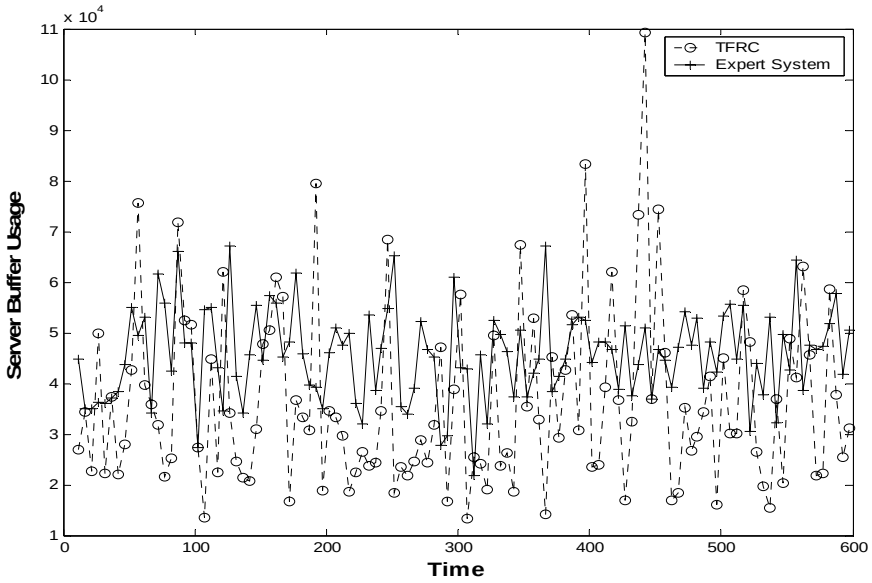


Figure 19 Usage of Server Buffer with TFRC or Expert System Control

7 Conclusion

In this paper, we present a novel expert server system for multimedia transmission. Results showed that the expert system can deliver smooth streams to clients under various situations efficiently. When congestion happened, the expert reacted smartly. In stead of fluctuating with the changes of loss rate aimlessly, it verified and took cooperative regulations to cope with the congestion. When congestion past, the expert system recorded it in its historical statistics and acted more careful in future control to prevent the congestion happening again. Attractively, these enhanced performances are achieved by taking only a small portion of CPU time to the execution of expert server control program.

The performances obtained in the case study section depend 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. It would become a practical and promising technique on media streaming server design.

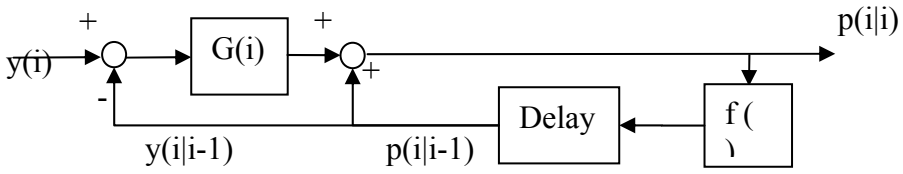
References

1. Forsyth and Richard, "Expert systems: principles and case studies", ISBN: 0412304600
2. Robert Keller, "Expert system technology: development and application", NJ: Yourdon Press, c1987, ISBN: 0132955776
3. Suzanne Smith; Abraham Kandel; Boca Raton, "Verification and validation of rule-based expert systems", CRC Press, c1993, ISBN: 084938902X
4. 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
5. 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

6. 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
7. Jin Wu; Djemame, K.; "An expert-system-based structure for active queue management", Int. Conf. on Machine Learning and Cybernetics, Volume 2, 2-5 Nov. 2003 Page(s):824 - 829 Vol.2
8. 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
9. Liang Ji; Arvanitis, T.N.; Woolley, S.I., "Fair weighted round robin scheduling scheme for DiffServ networks", *Electronics Letters*, Volume 39, Issue 3, 6 Feb. 2003 Page(s):333 - 335
10. Zhou, X.F; Ong, K., "Discrete LQ rate control schedule system for multimedia transmission", *International Conference on Advanced Information Networking and Applications*, Vol.1, Page(s): 6,18-20, Vienna April 2006
11. 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
12. Zhu, L.; Ansari, N.; Liu, J., "Throughput of high-speed TCP in optical burst switching networks", *IEE Proceedings-Communications*, Volume 152, p. 349-352, Issue 3, 3 June 2005
13. Lisong Xu, "Extending equation-based congestion control to high-speed and long-distance networks", *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Volume 51, Issue 7 (May 2007) Pages: 1847-1859
14. 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*, 49:237--252, 1998.
15. Y.P.Zhang; D.Loguinov, "Oscillations and Buffer Overflows in Video Streaming under Non-Negligible Queuing Delay", *ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Pages: 88 – 93, Cork, Ireland, June 2004
16. M. Dai and D. Loguinov, "Analysis of Rate-Distortion Functions and Congestion Control in Scalable Internet Video Streaming", *ACM NOSSDAV*, Pages: 60 – 69, Monterey, CA, USA, June 2003
17. Y.P.Zhang, S.R.Kang, and D.Loguinov, "Delayed Stability and Performance of Distributed Congestion Control", *ACM SIGCOMM*, Pages: 307 - 318, Portland, Oregon, USA, August, 2004
18. 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'05.
19. S.R.Kang, Y.P.Zhang, M.Dai, and D.Loguinov, "Multi-layer Active Queue Management and Congestion Control for Scalable Video Streaming", *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, Pages: 768 – 777, 2004.
20. M. S. Grewal and A. P. Andrews, "Kalman filtering: Theory and practice using MATLAB, 2nd Edition," John Wiley & Sons, 2001.

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

This appendix gives the Kalman filter diagram, loss rate state function, and implementation algorithm. The aim of Kalman Filter (with reference to the above Kalman filter block diagram) is to get the best estimation of state variable 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(i). It tries to get a filter gain G(k) so that the prediction $p(k \mid k) = p(k \mid k-1) + \mathrm{G}(k)[\mathrm{y}(k) - p(\mathrm{i} \mid \mathrm{i}-1)]$ minimizes the predict error covariance $\Phi(k \mid k) = E[(p(k) - p(k \mid k))(p(k)\, p(k \mid k))^T]$.



Kalman Filter Block Diagram

The overall filter procedure is: the Kalman Filter uses state function as the a priori state estimate for p(k+1) given knowledge of p prior to step k. After getting the observation y(k), it performs a posteriori state estimation for p(k|k) and meanwhile update the control gain G(k). Parameters and their meaning are listed in the following table.

Variable Name	Meaning
k	Time step
y(k)	Measured loss rate at time step k
p(k)	Accurate loss rate at time step k
p(k|k-1)	A priori prediction of p before giving y(k)
p(k|k)	A posterior estimation of p given y(k)
G(k)	Filter gain at time step k
Φ(k|k)	Covariance of a priori prediction error
Φ(k|k-1)	Covariance of a posterior estimation error

Table. Parameters used in Extended Kalman Filter Design in Kelly's Control

Discrete Kelly's rate control function [10]:

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

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. Kalman Filter will be used to get a more accurate loss rate under noise.
Assumptions:

- Fairness among traffics 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:

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

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

Where 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:

$$\begin{cases} p(k+1) = 1 - \dfrac{r(k)(1-p(k))}{r(k) + \alpha - \beta r(k)\, p(k)} & \text{---(A.4)} \\ y(k) = p(k) + n(k) & \text{---(A.5)} \end{cases}$$

From (A.4), state function for loss rate is non-linear and 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 can be ignored here because the rate control performs at the time interval that is small enough, 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:

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

Where $\nabla f\big|_{p(k|k)} = \dfrac{\partial p(k+1)}{\partial p(k)}\bigg|_{p(k|k)} = \dfrac{(1-\beta) r^2(k) + \alpha r(k)}{(r(k) + \alpha - \beta r(k)\, p(k))^2}$

The problem is changed to design a Kalman filter for a linear system. The detailed deduction of Kalman gain is shown in [20] and therefore omitted here. The implementation algorithm is given below.

Given the initial values of p(0) and Φ(0)
WHILE (transmitting){

$p(k \mid k-1) = 1 - \dfrac{r(k-1)(1 - p(k-1 \mid k-1))}{r(k-1) + \alpha - \beta r(k-1)\; p(k-1 \mid k-1)}$ /*Predict the state p(k|k-1)*/

$\Phi(k \mid k-1) = \Phi(k-1 \mid k-1)$ /*Predict the error covariance*/

$G(k) = \dfrac{\Phi(k \mid k-1)}{\Phi(k \mid k-1) + R(k)}$ /*Compute the Kalman Gain*/

Waiting for y(k) from client feedback…

$p(k \mid k) = p(k \mid k-1) + G(k)[y(k) - p(k \mid k-1)]$ /*Estimate p(k) with measured y(k)*/

$\Phi(k \mid k) = (1 - G(k))\; \Phi(k \mid k-1)$ /*Update the error covariance for time-step k+1*/

Increase k

}