THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Hong Kong Polytechnic University

Department of Computing

Soft Computing Based Dynamic Buffer Tuning for Better Response Timeliness and Fault Tolerance for Internet Channels

Wilfred Wan-Kei Lin

A Thesis Submitted in Partial Fulfilment of the Requirements
for the Degree of Doctor of Philosophy

May 2005

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

LIN WAN KEI WILFRED

ABSTRACT

The area of this PhD research is directed towards performance enhancement and fault-tolerance at client/server(C/S) interaction over a logical Internet channel. The aim is to effectively eliminate the user-level buffer overflow so that retransmissions can be reduced to shorten the service roundtrip time (RTT) in the interaction. Since a server may serve different clients simultaneously, the relationship is actually one-server-to-many-clients, alternatively known as the asymmetric rendezvous. The different streams of service requests from clients merge at the server's queue and this easily inundates the queue buffer to overflow at peak times. In fact, an asymmetric rendezvous involves two levels: the system/router level that includes all activities inside the TCP channel, and the user level that involves the client and the server. If the collective error probability for a client/server interaction path is ρ_{path}, then the average number of trials (ANT) to send a message successfully from one end of the C/S path to another is $ANT = \sum_{j=1}^{k\to\infty} j\rho_{path}^{j-1}(1-\rho_{path}) \approx \frac{1}{(1-\rho_{path})}$. Since ρ_{path} also encapsulates the user-level buffer overflow error, eliminating the latter definitely yields a smaller ANT and shorter end-to-end service roundtrip time (RTT).

My previous MPhil research concluded that dynamic buffer size tuning can indeed eliminate the chance of user-level buffer overflow. This was clearly demonstrated by the experimental results with the dynamic buffer controllers proposed. These original controllers developed in the MPhil thesis are:

1) PIDC (“*proportional (P) + integral(I) + derivative(D)*” *Controller*): It is algorithmic and always eliminates user-level buffer overflow but has two shortcomings: a) it locks unused memory, and b) it does not have a safety margin and therefore the queue length can get dangerously close to the buffer length, threatening possible overflow.

2) GAC (*Genetic Algorithm Controller*): It is the “PIDC + genetic algorithm (GA) + $\{0,\Delta\}^2$ objective function” combination. The GA moderates the PIDC process so that the outcome is always within the $\pm\Delta$ safety margins about the steady-state reference symbolically represented by “0” in $\{0,\Delta\}^2$. The GA eliminates the PIDC shortcomings but also produces occasional buffer overflow because it does not guarantee the global-optimal solution of the solution hyper-plane.

3) FLC (*Fuzzy Logic Controller*): It is the combination: “*PIDC + fuzzy logic +* $\{0,\Delta\}^2$ *objective function*” combination, which was proposed to preserve the GAC merits and eliminate the occasional buffer overflow. The fuzzy logic moderates the PIDC control process similar to the GA.

4) NNC (*Neural Network Controller*): It works with the $\{0,\Delta\}^2$ objective function but does not include PIDC. Its proposal was inspired by the successful experience of using neural networks in AQM (active queue management) algorithms, which prevent network congestion at the system/router level. AQM methods differ from the dynamic buffer size tuners by using a fixed-size buffer.

When experiments were conducted to verify the above four dynamic buffer tuners, it was observed that their performance was affected by the traffic patterns. The conclusion is that measures must be taken to neutralize the ill effects by traffic on

tuner stability and accuracy. My MPhil thesis left several unaddressed issues that form the backbone of this PhD research. The issues include:

1) In the aspect of traffic ill effects: a) Is it possible to calibrate the ill effects off-line so that the tuners can use these calibrations to ward off traffic changes by fine-tuning its dynamic buffer tuning process adaptively? b) If so, then how can the current Internet traffic pattern be deciphered on the fly (on-line) so that the off-line calibrations can be applied selectively?
2) For FLC: a) Is it possible to have an optimal design? b) Is it possible to make the tuner self-reconfigurable (especially with respect to traffic pattern changes)?
3) For NNC: a) Is it possible to prune the NNC configuration on the fly so that its control cycle time can be consistently and adaptively reduced? b) Is there a correlation between control accuracy and the number of hidden neurons in the NNC back-propagation architecture? (The procedure to provide the answer is called sensitivity analysis.)

The motivation of my PhD research is to provide answers to the above unaddressed issues. As a result the following solutions are proposed:

1) For real time traffic analysis: Two traffic filters have been proposed: *real-time modified QQ-plot* (or simply RT-QQ) and *self-similarity (S^2) filter*. These filters identify the Internet traffic patterns on the fly. The RT-QQ recognizes heavy-tailed distributions and the S^2 filter identifies self-similarity.
2) For FLC: a) an optimal design range is found for FLC design, and b) a way is

found to make the FLC adaptive/reconfigurable by squeezing the "*don't care*" state range threshold in a dynamic manner.

3) For NNC: a) the HBP (*Hessian Based Pruning*) approach was proposed for pruning or optimizing the NNC configuration on the fly and as a result its average execution time (i.e. control cycle time) is reduced, and b) sensitivity analysis was conducted and the results confirm that more hidden neurons do not necessarily mean better NNC performance.

The solutions proposed in my PhD research have contributed to 19 publications so far (5 journals and 14 conferences). All the stated PhD research objectives have been achieved. The research has also uncovered many relevant problems, which should be resolved in the future work: a) investigation of the issue of how to choose the limits for Gaussian tests effectively, b) deepening of the investigation into why "*heavy-tailedness*" is not a necessary condition of self-similarity, and c) investigation into how the dynamic buffer size controllers, especially the FLC, can best support pervasive computing based e-applications such a telemedicine.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to express my gratitude to the thesis supervisors Dr. Allan Wong and Prof. Tharam S. Dillon for their guidance and support throughout the study. I am thankful to the Hong Kong Polytechnic University for my PhD Tuition scholarship and clerical support from Ms. Miu Tai of the Department of Computing General Office. I am grateful to the technical teams of the Department of Computing (The Hong Kong Polytechnic University), Department of Computer Science & Computer Engineering (Latrobe University, Melbourne, Australia), and Faculty of Information Technology (University of Technology Sydney, Sydney, Australia) for their support for effective data collection in the different experiments. I also acknowledge the Visiting Scholar Appointment with University of Technology Sydney from 11th December 2004 to 7th February 2005. I would like to extend my gratitude to Mr. Richard Wu and Mr. Jason Lo, for discussions of their experience in the preliminary investigation phase for this thesis.

Finally, I would like to express my gratitude to my family and friends with their endless support.

Wilfred Lin

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AIMD	Additive Increase and Multiplicative Decrease
ANT	Average Number of Trials
AQM	Active Queue Management
ATM	Asynchronous Transfer Mode
A-FLC	Adaptive Fuzzy Logic Controller
CA	Convergence Algorithm
CAB	Continuous Aggregate Based
CMT	Consecutive Message Transmission
dQ/dt	Rate of change of queue length
E-RTPD	Enhanced Real-time Traffic Pattern Detection
FLB	Fixed Length Buffer

FLC	Fuzzy Logic Controller
FRP	Fractal Renewal Process
FSNDPP	Fractal-Shot-Noise-Driven Poisson Process
GAC	Genetic Algorithm Controller
GP	Given Percentage
HBP	Hessian Based Pruning
ICM	Integral Control Mechanism
IEPM	Internet End-to-End Performance Measurement
IET	Investigate & Experiment & Iterate Methodology
IETF	Internet Engineering Task Force
IP	Internet Protocol

LRD	Long-Range Dependence
M^2RT	Mean Message Response Time
M^3RT	Micro Mean Message Response Time
MD	Mean Deviation
MDS	Mobile Distributed Systems
NNC	Neural Network Controller
OTA	Offline Traffic Analysis
O-NNC	Optimized Neural Network Controller
PCI	Pervasive Computing Infrastructure
PIDC	Proportional (P) + Integral (I) + Derivative (D) Controller
QOB	Queue Length over Buffer Length
QOB_R	Queue Length over Buffer Length Reference

QoS	Quality of Service
RED	Random Early Detection
RFC	Request for Comments
R/S	Rescaled Adjusted Range Statistics
RTPA	Real-time Traffic Pattern Analysis
RTPD	Real-time Traffic Pattern Detection
RTT	Round Trip Time
R^2-FLC	Real-time Reconfigurable Fuzzy Logic Controller
RT-QQ-plot	Real-Time Modified QQ-plot
R/S	Rescaled Adjusted Statistics
S^2 filter	Self-Similarity filter

SAP	Service Access Point
SFF	Small Form Factor
SRD	Short-Range Dependence
TCP	Transmission Control Protocol
VLB	Variable Length Buffer

CHAPTER 1 BACKGROUND AND MOTIVATION

1.0 INTRODUCTION

The transport layer of the Internet supports two protocols: the connection-oriented TCP (Transmission Control Protocol) and the connectionless UDP (User Datagram Protocol) [Comer1995]. It is not easy to use the TCP for time-critical applications because of the inevitable channel error probability ρ at the system level that occurs due to the sheer size and heterogeneity of the underlying network. Sending a message/segment from one TCP end to another physically means traversing many different links and nodes of varying quality and capacities. Firstly, the Internet conceptually is a collection of large backbones (e.g. US backbone and European backbone) that are interconnected by the IP (Internet Protocol). In fact, it is not unusual that two IP peers are sandwiched by incompatible protocols such as the ATM (Asynchronous Transfer Mode). Then, the IP peers rely on the technique of *tunnelling* to communicate properly [Hassan2000]. In another scenario the IP peers may actually communicate in a transparent manner via the different wired and wireless parts of the Internet. Wireless and wired communications have very different requirements. For example, the wired part of Internet will opt to slow down transmissions when the possibility of network congestion is envisioned. On the contrary, in wireless communication packet loss due to congestion or other reasons will trigger even more aggressive transmissions by the sender. The aim is to make up for the lost messages quickly [Cen2003].

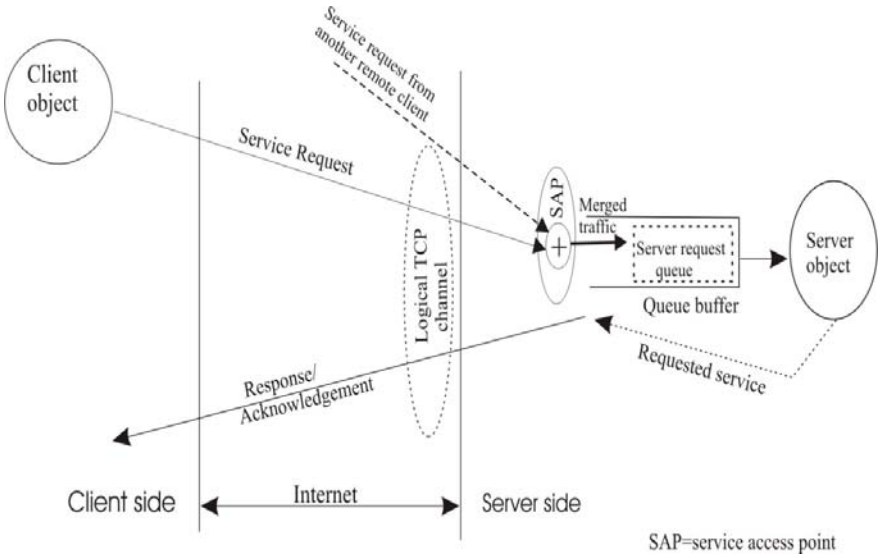


Figure 1.0.1 A simple representation of an asymmetric rendezvous

A client/server interaction is considered to have two levels: system and user. The system or router level (marked "Internet" in Figure 1.0.1) includes all the activities within the TCP channel, and the user level includes the client and the server that interact over the TCP channel in the end-to-end manner. Therefore the error probability for a client/server interaction path (referred to as the "*C/S path*" in this thesis) ρ_{path} is made up of two parts: the collective channel error probability ρ at the system/router level and the collective one at the user level ρ_U; $\rho_{path} = \rho + \rho_U$. Then, the average number of trials to send a message successfully from one end of the C/S path to another is $ANT = \sum_{j=1}^{k\to\infty} j\rho_{path}^{j-1}(1-\rho_{path}) \approx \frac{1}{(1-\rho_{path})}$. Therefore, lowering/eliminating either ρ or ρ_U, or both, yields a smaller ANT and thus shorter end-to-end service roundtrip time (RTT). There are many possible causes that

contribute to ρ and/or ρ_U such as hardware partial failures and buffer overflow. For example, network congestion at the system level may lead to router buffer overflow, which means message losses and timeouts by the respective senders, leading to widespread retransmission and more network congestion. The buffer at the user-level receiving end (i.e. server's end) may also be inundated by fast incoming messages to overflow unless the buffer can self-tune to ensure that buffer length always covers the queue size. There is a need for research to explore how to enable a reception buffer at the user level to self-tune on the fly, thereby eliminating the chance of overflow (i.e. dynamic buffer size tuning).

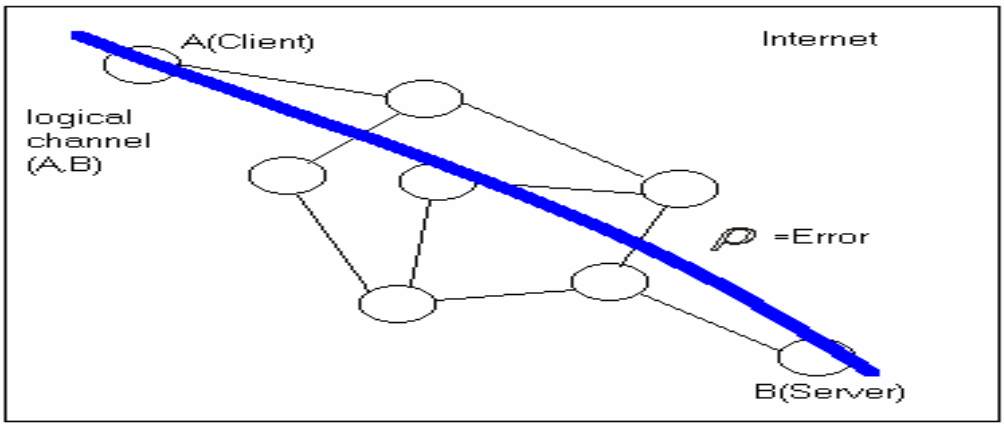


Figure 1.0.2 Client/server interaction over a logical channel with error probability ρ

The importance of reducing ρ_{path} for better service response is well recognized. The benefit of such reduction is best viewed from the point of system dependability [Avizienis2004], which is defined by the following attributes: reliability, availability, fault tolerance, security, integrity, and maintainability.

1.1 NETWORK CONGESTION PREVENTION

From the literature, ρ_{path} reduction in the area of network congestion prevention and buffer overflow control may be achieved as follows:

1) System-level sender initiative:

   a) Dynamic timeout window adjustment: The sender adjusts the timeout window on the fly with respect to the current values of some chosen parameters to avoid premature timeouts and unnecessary retransmissions (e.g. the *Adaptive and Aggressively Bounded Convergence Algorithm* [WongHC2001]).

   b) Dynamic congestion window tuning: The AIMD (Additive Increase and Multiplicative Decrease) is a well-known example proposed by Jacobson [Jacobson1988] to adjust the congestion window of a TCP connection. Another example is *adaptive congestion window tuning* for a Reno TCP [Padhye1998].

   c) Multiple copies of time-critical messages [Rama1992]: The sender sends multiple copies of the same message immediately one after another. The number of copies corresponds to the likelihood of congestion. The argument is that if the C/S path error probability for sending a message is ρ_{path}, then the chance for *mc* number copies to be erroneous at the same time is $(\rho_{path})^{mc}$, which is a smaller error.

2) Active Queue Management (AQM) by the system-level receiver/router [Braden1998]: If a router detects that its reception buffer is likely to overflow, then it throttles the sender to slow down transmission voluntarily. A router starts the throttling process by sending “choke” packets. If the sender does not respond to the throttling, then the router drops the incoming packets to facilitate smooth passage of those already queued. The message dropping process may follow different strategies, for example, “drop from front” [Lakshman1996]. In fact, dropping messages as a congestion and buffer overflow prevention mechanism is deleterious. Recently the IETF (Internet Engineering Task Force) proposed to use the RED (Random Early Discard) algorithm for AQM purposed in the RFC 2309 [Braden1998]. The subsequent analysis of RED found that it was unstable and this led to the different RED mutants (e.g. the algorithmic ones, FRED (Fair RED) [Kim1998], DS-RED [Zheng2001], LRU-RED [Reddy2001], M-RED [Koo2001], REM [Athuraliya2001]) and the intelligent non-RED-based versions (e.g. Fuzzy-PI [Ren2002]; P for proportional control and I for integral control). Floyd and Jacobson call those routers in packet-switching networks that adopt the RED algorithm the Random Early Detection Gateways [Floyd1993].
3) Using backup channels [Kris2003, Shin2000]: There is always a urgent need to control the message delivery/roundtrip time in real-time computing over the Internet so that tasks can be meaningfully executed before the deadline [Stankovic1998]. This is absolutely necessary for *hard* and *firm* real-time

applications and less stringent for the *soft* type. Using backup channels, which have the guaranteed level of reliability when congestion is detected, is the state-of-the-art solution. It may be more expensive to temporarily relinquish the normal channel and switch to the more reliable backup channel that guarantees the QoS (quality of service) to reduce ρ_{path}. The meaningful timely result, however, could be worth much more than the cost. Sometimes a reliable backup channel is time-shared by many normal channels.

4) Dynamic buffer size tuning [Wong1999A, WongHC2001, Wong2002GAC]: The principle is to tune the reception buffer size adaptively on the fly so that the buffer length always covers the queue size and therefore eliminates any chance of buffer overflow. So far all the dynamic buffer tuners, namely, PIDC, GAC, FLC and NNC are aimed at user-level applications.

To summarize, the three basic techniques to deal with buffer overflow are throttling, message dropping, and dynamic buffer tuning [Tanenbaum1996]. The four techniques that effect ρ_{path} reduction without tuning the buffer size are: a) tuning the timeout window adaptively, b) tuning the congestion window adaptively, c) sending multiple copies of the same message immediately one after another to logically reduce ρ_{path}, and d) using backup channels to bypass the bottlenecks.

The throughput of a communication channel depends on how efficiently by the supporting system can recycle usable memory. If too much memory is locked up in communication activities, then the whole system throughput may suffer because tasks

are suspended on memory shortage. Likewise, if a communication system is constantly starved of buffer memory random drop of messages [Lakshman1997, Paxson1999] and buffer overflow inevitably happen. The result is massive retransmissions by senders and widespread data traffic jams. Elimination of buffer overflow in client/server interaction [Lewandowski1998] is a significant and yet challenging balancing act in memory usage [Amir1995, Alvisi1998, Crawford2000, Cristian1999, Garbinato2000, Ip2001, Markatos1998, Mishra1998, Morin1997, Mukherjee1998, Ramani2000, Schmidt1995, Sobczak2001, Wong1999A, Wong2000B, Wuytack1999]. For commercial applications such as ISP (Internet Service Provider) setups any excessively long response delay/latency due to retransmissions would cause business loss because it taxes customers' patience and drives them away. In such cases it is justified to aggressively apply one or a combination of the aforementioned basic techniques to reduce the response time to make customers happy.

1.2 BUFFER TUNING SCHEMES

The fact that buffer overflow prevention shortens TCP channel RTT has spurred development and deployment of different algorithmic and expert approaches for applications at the system and user levels [Fisk2001, Dunigan2003, Aweya2002]. These algorithms can be classified in different ways by various attributes, as follows:

a) Open loop versus closed loop: Open loop algorithms do not require behavioral feedback for controlling the future trend, while feedback is mandatory for closed loop systems [Yang1995].

b) FBL (fixed buffer length) versus VBL (variable buffer length): For FBL algorithms the ultimate overflow prevention solution is to drop packets. This may occur in two stages: i) firstly, the receiver throttles the sender to reduce transmission, and ii) if this does not help then incoming packets are dropped either “front on full” or “random on full” [Lakshman1996]. VBL algorithms prevent overflow by dynamic buffer size adjustment without the necessity of throttling the sender first [Wong1999A].

c) Algorithmic versus expert: Algorithmic approaches do not use soft computing techniques but expert systems do [Karray2002, Ravindran2001].

d) System level versus user level: Algorithms at the system level operate without user intervention, for example, the AQM operations [Braden1998]. If they operate in the client and server domains independent of the system, they are working at the user level [Wong1999A].

e) Implicit versus explicit: In implicit control the remedy is negotiable, for example, the voluntary reaction by the sender when throttled by a router [Ren2002]. If the remedial response is instantaneous and involuntary, it is explicit control (e.g. [Wong2002GAC]).

f) Direct versus indirect: Direct control invokes immediate action, for example, tuning the buffer size spontaneously [Ip2001]. Indirect control depends on voluntary reaction.

1.3 BUFFER OVERFLOW MANAGEMENT

The motivation of the research is to explore how soft computing techniques [Pedrycz1997, Zadeh1994] can be used to gain efficacious user-level dynamic buffer overflow control for Internet channels for better response timeliness [Kang2002, Stankovic1998] and fault tolerance [Avizienis2004, Elnozahy1999, Gartner1999, Jalote1994, Laprie1995]. This research project is a deeper continuation of my MPhil thesis [Lin2002], in which four original dynamic buffer overflow controllers were proposed [Appendix 1]. One of them, namely, the PID or "P+I+D" controller (*Proportional + Integral + Derivative controls*) is algorithmic. The control parameters of the PID controller or PIDC remain unchanged once the control process has started. The other three controllers are soft computing based and work with the $\{0, \Delta\}^2$ objective function, where Δ is the safety margin to be maintained about the reference symbolically represented by "0". In reality the reference is a given queue length over buffer length (QOB) ratio known as the QOB_R. The three intelligent dynamic buffer controllers for user-level applications are: the GAC (Genetic Algorithm Controller [Wong2002GAC]), the FLC (Fuzzy Logic Controller [Lin2002FLC]), and the NNC (Neural Network Controller [Lin2001NNC]). The PIDC was based on the "P+D" dynamic buffer size tuner controller for user-level application [Wong1999A]. The "P+D" controller was the first of its kind but failed frequently in actual deployment over the Internet. The cause was the unrealistic expectation of using a set of static parameters to control the whole spectrum of changes in TCP channel dynamics. The PIDC rectifies the "P+D" problem by adding

the integral (I) control. It differs from the GAC, FLC and NNC by having no safety margin (i.e. Δ) at all, and the accumulated performance data shows that the danger of buffer overflow is still there under serious perturbations. The basis and evolution process in my MPhil research is summarized in Figure 1.3.1.

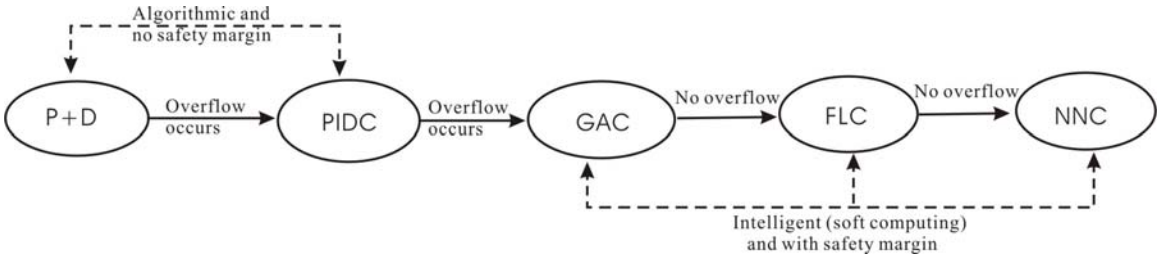


Figure 1.3.1 Summary of the basis and evolution of my MPhil project

The previous Internet based experimental results with the four novel controllers indicate that they represent the right direction to eliminate user-level buffer overflow along the client/server interaction path. This path over a TCP (Transmission Control Protocol) channel is also known as the asymmetric rendezvous. The GAC was proposed to preserve the PIDC merits minus its shortcomings. Nevertheless, as a result of the very nature of the genetic algorithm (GA), which does not guarantee the global-optimal solution of the solution hyperplane [Mitchel1999], the GAC produces occasional though rare buffer overflow. The GAC results do verify that the $\{0, \Delta\}^2$ objective function is a powerful concept, and it can serve as a solid basis for other intelligent solutions. This led to the FLC proposal and subsequently the NNC development. What I had achieved in my MPhil thesis can be summarized as follows:

a) Four novel dynamic buffer overflow controllers for user-level applications were proposed, one algorithmic (i.e. the PIDC) and three intelligent ones (i.e. GAC, FLC and NNC).

b) The GAC was thoroughly tested and found to be unacceptable because it yields occasional buffer overflow.

c) The FLC was proposed and two designs, namely, FLC [4x4] and FLC [4x6] were tested. The results indicated this direction is the right one because of the following: i) it eliminates buffer overflow completely, ii) its execution time is comparable to the simpler PIDC's due the presence of the "*don't care*" state [Lin2002FLC], and iii) it always maintains the control output within $\pm\Delta$ about the chosen QOB_R reference. Yet, its convergence to QOB_R can be oscillatory.

d) The success of using $\{0,\Delta\}^2$ as the operational principle and the desire to have a smoother QOB_R convergence led to the proposal of the NNC. The NNC differs from the GAC and the FLC because it does not include the PIDC as a component. The NNC, however, has a much longer control cycle time compared to the PIDC, GAC and the FLC and this is prone to deleterious effects. The argument is that by the time the remedy is computed the actual problem has already passed. Using the computed remedy to resolve a spurious problem may lead to undesirable consequences or deleterious effects. The NNC prototype, which works by backpropagation with supervised training, has 10 input neurons, 20 neurons in the hidden layer, and one output neuron.

e) Timing analyses confirmed that the four novel dynamic buffer size tuning models are indeed suitable for time-critical applications over the Internet.

The area of user-level dynamic buffer size control, which tries to ensure that the buffer length always covers the queue size on the fly, is pristine. For this reason my MPhil research is able to produce 12 refereed publications (4 journal papers and 8 conference papers). The MPhil research, however, also left some important, unaddressed issues:

a) Does the Internet traffic impede the controllers' stability and accuracy? If so how can the impedance be alleviated or neutralized? In fact, the internet traffic can change without warning, for example, from LRD (long-range dependence) such as heavy-tailed and self-similar to SRD (short-range dependence) such as Poisson [Molnár1999]. Such changes may have a serious impact on the controllers' performance.
b) Is it possible to have an optimal (cost effective) FLC design?
c) Is there a correlation between the accuracy and the number of neurons in the hidden layer of the NNC? In my PhD research finding such a correlation is called *sensitivity analysis*.
d) Is it possible to cut down the NNC control cycle time and lower the chance of a deleterious effect?

1.4 SCOPE OF THESIS

The motivation to address the above issues becomes the problem statement of my PhD project, with the aim to achieve the following objectives:

a) Study the impact of traffic on the stability and accuracy of the FLC and the NNC, and propose methods to counteract the negative impact effectively.

b) Explore and define the possible optimal range for the FLC design and implementation.

c) Define the correlation between the number of neurons in the NNC hidden layer and the control accuracy.

d) Propose a method(s) to optimize the NNC configuration to lower its control cycle time.

e) Perform timing analyses of the improved or new FLC and NNC models to confirm that they indeed suitable for time-critical applications over the Internet.

Figure 1.4.1 accentuates the importance of buffer overflow control over the path of asymmetric rendezvous (one-server-to-many-clients relationship) over a TCP channel. Efficacious buffer overflow control is the prelude for running time-critical applications over the Internet successfully [Stankovic1998] because it reduces the *service roundtrip time* (SRTT or simply RTT). As a result the response timeliness is enhanced. The server at the user level in an asymmetric rendezvous usually serves many clients simultaneously [Lewandowski1998]. Any sudden influx of requests from these clients to be queued at the server's buffer could cause buffer overflow,

which means request losses and possible widespread retransmissions [Lakshman1997, Paxson1999, Jamjoom2004]. The *average number of trials* (ANT) to get a transmission success depends on the *C/S path* error probability ρ_{path}. If the P_j is the probability for a transmission success at the j[th] trial, then $P_j = \rho_{path}^{j-1}(1-\rho_{path})$ leads to $ANT = \sum_{j=1}^{k\to\infty} jP_j$ or $ANT = \sum_{j=1}^{k\to\infty} j\rho_{path}^{j-1}(1-\rho_{path}) \approx \frac{1}{(1-\rho_{path})}$. *S*ince the overflow probability is part of the overall ρ_{path} any overflow elimination along the client/sever interaction path yields a smaller ANT and thus a shorter service RTT.

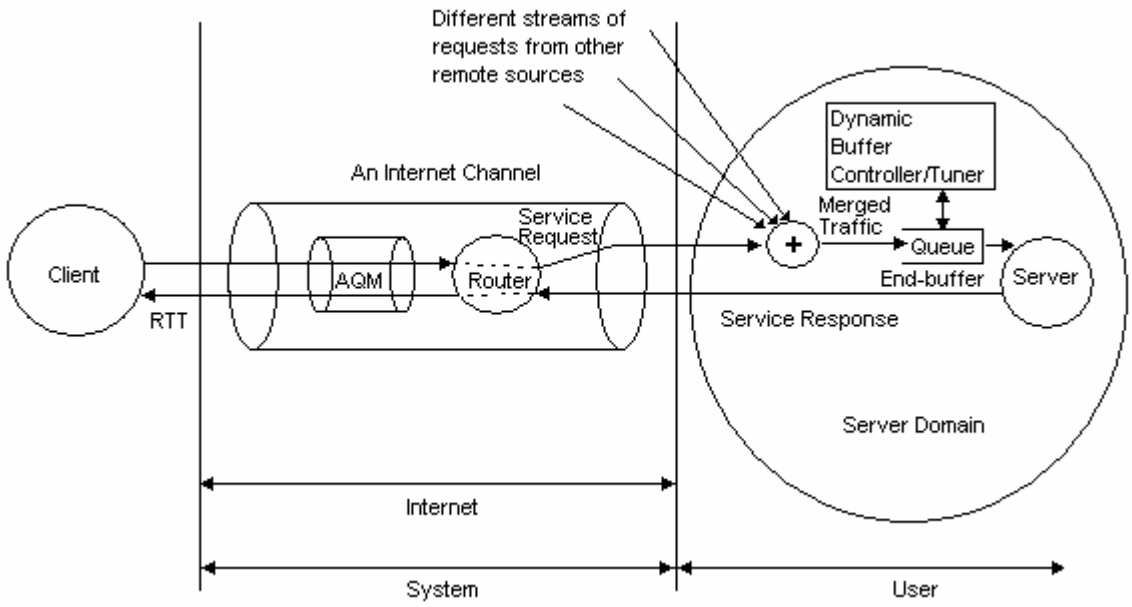


Figure 1.4.1 End-to-end logical channel between client and server

In reality the buffer overflow can occur at both the system/router and user levels. The system/router level includes all the routing activities within the TCP. Here the sender and the receiver can contribute to prevent network congestion, which is

manifested as buffer overflow at the congested routers or bottle-necks. The AIMD (*Additive Increase Multiplicative Decrease*) approach proposed by Jacobson [Jacobson1988], for example, is a measure for a sender within the TCP to control the congestion window adaptively. This lowers the transmission rate thereby alleviating congestion. The router can also choose to actively throttle any sender that sends too much data in a short time. The throttling act is called AQM (active queue management) [Braden1998]. Since the throttled sender reacts only voluntarily, the AQM process may fail and the router then may to have to drop new incoming packets. The goal is to ensure that those already queued have a smooth passage. Using message dropping as a strategy [Floyd1993] to prevent network congestion is deleterious even though it prevents router buffer overflow because on the other hand it increases retransmission, which causes more congestion. System-level buffer overflow or congestion prevention alone, however, cannot prevent the user-level overflow. The reason is that "*merged traffic*" from the combined client requests streams (Figure 1.4.1) can still inundate the buffer easily to overflow. My MPhil research indicates that the inundation is definitely caused by the high traffic rate and possibly by the embedded traffic pattern. Yet, the effect of the embedded traffic pattern was not explored and studied. The buffer inundation problem can be alleviated if the buffer is provided with the capability to self-tune and assure that the buffer length always covers the queue size. The assurance is called *dynamic buffer size tuning* in both of my MPhil and PhD research. If user-level buffer overflow is allowed to occur after the system has dished out expensive congestion prevention effort, the consequence could be disastrous. Not only are valuable resources wasted

but the system also loses the chance of rectifying a serious problem earlier. Therefore, user-level dynamic buffer tuning and system congestion prevention together is a unified solution to stifle the chance of buffer overflow along the client/server interaction path.

The potential of shorter service RTT in an asymmetric rendezvous by having buffer overflow control has inspired the emergence of different strategies [Chatranon2004]. These strategies are divided into two basic categories, namely, *fixed length buffer* (FLB) [Aweya1998, Feng1999] and *variable length buffer* (VLB) [Ip2001, Lin2001NNC, Lin2002FLC, Wong2002GAC]. The FLB approach is naturally deleterious because dropping incoming messages as the ultimate solution to prevent congestion and buffer overflow would cause widespread request retransmissions [Grinnemo2004, Jamjoom2004]. At this moment all the known AQM approaches from literature to prevent network congestion and router buffer overflow are exclusively FLB in nature. The VLB approach is relatively recent and the only examples that can be identified from literature include the PIDC, FLC, GAC, and NNC. These four controllers are designed for user-level applications. The desire to eliminate the two PIDC shortcomings [Ip2001] led to the development of the intelligent FLC, GAC and NNC. These shortcomings are: a) the controlled queue length can get dangerously close to the buffer length leading to possible overflow under serious perturbations, and b) too much buffer space is locked up even when it is no longer needed for remedial action.

CHAPTER 2 EVALUATION OF PREVIOUS RESEARCH

2.0 INTRODUCTION

In the last chapter, we identified the importance of controlling network congestion on the Internet, in the presence of different traffic patterns.

We noted that network congestion prevention and buffer overflow control can be carried as follows:

1) Initiated by system-level sender

2) Active Queue Management

3) Using backup channels

4) Dynamic buffer size tuning

Firstly we will show the taxonomy of the techniques being utilized for network congestion control, and then we will discuss each of these techniques and evaluate their effectiveness.

2.1 CLASSIFICATION OF CONGESTION MANAGEMENT TECHNIQUES

Table 2.1.1 shows a few overflow controllers and their attributes, and Figure 2.1.1 is the brief taxonomy of different queue buffer management techniques.

	Algorithmic	Expert
System level	The RED AQM algorithm [Braden1998] (closed loop, implicit, FBL, indirect)	The PI fuzzy controller [Ren2002] (closed loop, implicit, FBL, indirect)
User level	The basic PID controller [Ip2001] (closed loop, explicit, VBL, direct)	The genetic algorithm controller [Wong2002GAC] (closed loop, explicit, VBL, direct)

Table 2.1.1 A few overflow controller examples for illustration

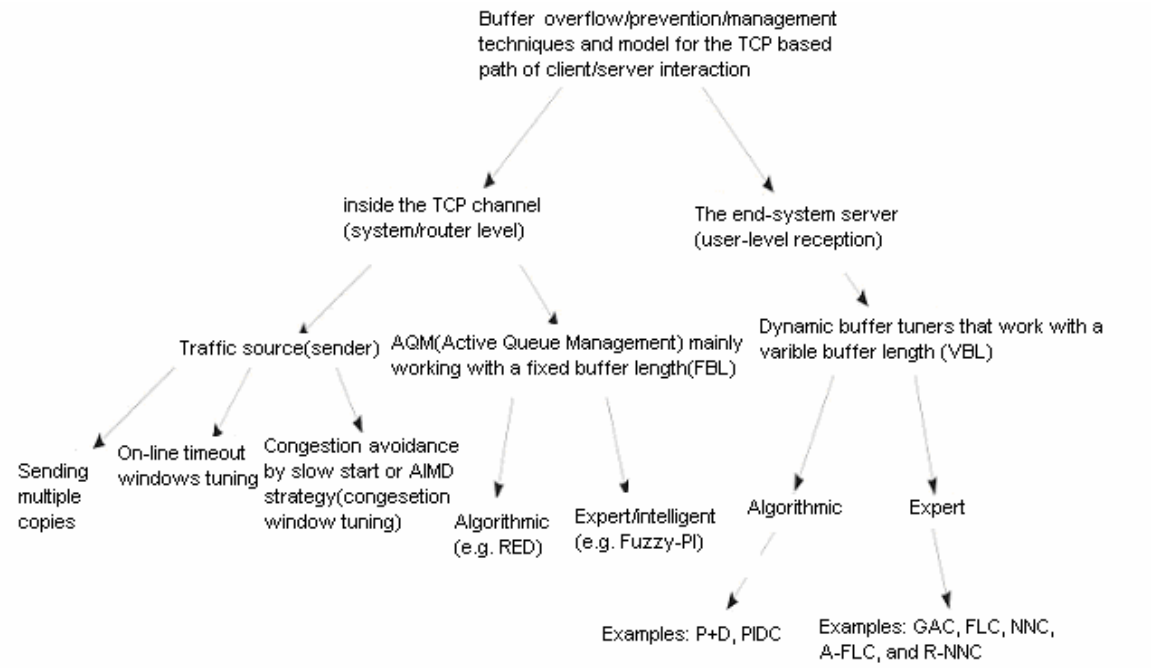


Figure 2.1.1 Brief taxonomy of different queue buffer management techniques

2.2 MECHANISMS INITIATED BY THE SYSTEM-LEVEL SENDER

The aim is to prevent premature timeouts to maximize the TCP channel bandwidth utilization. The mechanisms initiated by the system-level sender include the following:

a) Dynamic timeout window adjustment: The sender adjusts the timeout window T_{out} on the fly with respect to the currently measured values of some chosen parameters. The goal is to prevent premature timeouts and unnecessary retransmissions. How the TCP manages its retransmission timer (i.e. T_{out}) adaptively provides a good example. Unlike the data link protocols, which usually have predictable roundtrip times (RTT) with a low variance, the TCP (an Internet transport layer operation) has a large RTT variability spread. This makes the dynamic T_{out} adjustment process non-trivial [Jacoson1988]. Most TCP implementations adjusts T_{out} on the fly by using three parameters: the predicted RTT (PRTT), the currently measured RTT (MRTT), and the deviation D defined by $D = \varsigma D + (1-\varsigma)\,|\,PRTT - MRTT\,|$, where ς is a smoothing factor typically set to 7/8. The PRTT value is predicted by $PRTT = \varsigma PRTT + (1-\varsigma)MRTT$. Finally the next timeout interval for the retransmission timer is set to $T_{out} = PRTT + 4 * D$, where 4 is the commonly used figure for better performance, as determined from experience [Tanenbaum2003].

b) Dynamic congestion window tuning: The TCP is a full-duplex, connection-oriented Internet transport set up that strives to provide a reliable end-to-end byte-

stream based client/server interaction. A TCP connection is, however, considered "point-to-point" because the client/server interaction is "port-to-port". The client and the server can communicate only via two specific end points or ports, which are also known as the TSAP (Transport Service Access Points). A TCP connection is established if the server successfully responds to the CONNECT protocol primitive executed by a client. The server response includes the execution of the two primitives: LISTEN and ACCEPT. In the CONNECT primitive a client/sender specifies the IP address (i.e. the Network Service Access Point), the target port for connection, and the maximum TCP segment size expected. In the connection establishment process the server advertises the size of the sliding window for flow control [Tanenbaum2003]. The actual management of this size in the TCP, however, is decoupled from the acknowledgements. Since a segment sent through a TCP channel may be fragmented into smaller packets to be routed through the network layer, whether all the packets can be received correctly for assembly depends on the network capacity. Some packets may get lost or become corrupted.

The major cause for packet loss in the wired Internet is router congestion. A congested router drops the new incoming packets to prevent local buffer overflow and ensure the smooth passage for those already queued. Segment/packet retransmission can deleteriously aggravate network congestion and rapidly consume the bandwidth leading to poor system throughput. Therefore, the TCP needs to contribute actively to network congestion control. For example, the ***slow start*** algorithm that adaptively tunes the congestion window is supported by all TCP implementations [Jacobson1988]. In the algorithm if the receiver advertises a

reception window of 10K bytes but timeout occurs at 4K bytes, then the congestion window is set at 2K bytes to prevent any segment larger than this size to be sent. This is independent of what the receiver advertises. Initially the sender sets the congestion window to the size of the maximum segment in use by the connection, and then sends one maximum segment. If the corresponding acknowledgement is received before the timeout is triggered, it resets the retransmission timer and then sends a burst of two maximum segments. This process repeats and the congestion window grows exponentially by the factor of 2^P until it encounters a timeout or hits the receiver's window. The exponent P indicates the successive successful acknowledgements, for $P = 1,2,3,...,n$. To make the network congestion control by ***slow start*** even more effective a ***threshold***, which is typically set at 64K bytes initially (approximately the IP payload size), is used together with the receiver and congestion windows. This is the AIMD (*Additive Increase and Multiplicative Decrease*) approach. When a timeout or "choke packet" is sent by the router, the threshold becomes half of the current congestion window, which is set to one maximum segment size. ***Slow start*** then determines how much transmission the network can handle, but the exponential growth of the congestion window size stops once the threshold is hit. After this point the congestion window can grow only linearly by one maximum segment in every new burst. The AIMD approach, however, may create problems as follows:

i) Self-similar traffic generation: The AIMD encourages the maximum bandwidth usage by allowing burst sending behaviour in an exponential manner. This may create self-similar burst traffic that affects the receiver's stability as some of my experiments have revealed (Chapter 3 and section 6.3).

ii) Impoverished bandwidth utilization: The approach is useful for short-haul interactions, where the distance in terms of network latency between the client and server is not serious. For long-haul operations such as the "long-fat-pipes" [Nakamura2004] in high-bandwidth-high-latency networks the acknowledgements can be seriously delayed leading to spurious timeouts by the sender. As a result the subsequent decrease of the congestion window to one maximum segment can inadvertently impoverish the client/server interaction bandwidth utilization. This has inspired different solution proposals and the RRTP (Reconfigurable and Reliable Transport Layer Protocol) [Balakrishnan1997, Wang2004] is one example.

In fact, the quest for more effective *adaptive congestion window tuning* models is continuing, for example, the model proposed by Padhye et al [Padhye1998] for the Reno TCP applications.

c) Multiple copies of the same time-critical message [Rama1992]: The sender sends multiple copies of the same message immediately one after another. The number of copies ties with the likelihood of congestion. The argument is that if the C/S path error probability for sending a message is ρ_{path}, then the chance for *mc* number copies to be erroneous at the same time is $(\rho_{path})^{mc}$; it is much reduced. A design that is based on the $(\rho_{path})^{mc}$ criterion is also called the Consecutive Message Transmission (CMT) approach [Wong1999A]. The evaluation of some recent CMT findings [Wong1999A] has revealed the following:

i) Possible *orphan executions*: If the server/receiver side is designed to receive multiple copies of the same messages/requests, it must be able to support remote invocations of the *exactly-once* semantics. This is necessary for services that are not *idempotent*. An idempotent service differs by producing the same effect even when it is repeatedly and inadvertently invoked. If a non-idempotent service is invoked by the *at-least-once* semantics, then the additional copies of the same request would lead to orphan executions and possibly disastrous side effects. This requires the server to possess the power to differentiate the exactly-once semantics from the at-least-once invocations. Judging from the complexity and size of the Internet operation precise differentiation is not easy to accomplish.

ii) Balanced protocol design needed: The $(\rho_{path})^{mc}$ alone is not enough for an efficient CMT protocol. The number of copies in a CMT scheme should be supported by a proper "acceptance criterion", which is the number of copies of the same message received correctly before the transmission of the "original/intended" message is considered a successful reception. This can be demonstrated by comparing the following two CMT schemes:

A) Scheme 1: In a 4-copies CMT scheme, the receiver acknowledges correct reception of a message provided that it has received any 2 correct copies out of the 4. Every copy however has an error probability of $\upsilon = 0.4$ or 40%. This implies the probability for a transmission success to be $\upsilon_4 = PR_2 + PR_3 + PR_4 = 0.8208$. PR_X is the error probability for having X number of correct copies for $X = 1,2,3,4$; $PR_2 = \binom{4}{2}\upsilon^2(1-\upsilon)^2$, $PR_3 = \binom{4}{3}\upsilon(1-\upsilon)^3$, $PR_4 = (1-\upsilon)^4$. Then, the average number of trials (ANT) to send a message successfully from one end to

another can be calculated in the same way as for a C/S path with the ρ_{path} error probability: $ANT_{Scheme-1} = \sum_{j=1}^{k\to\infty} j\rho_{path}^{j-1}(1-\rho_{path}) \approx \frac{1}{(1-\rho_{path})}$. In scheme 1 the ANT for sending a message successfully means $\rho_{path} = (1-\upsilon_4)$ and $ANT_{Scheme-1} = 1/\upsilon_4 \approx 1/0.82 \approx 1.22$.

B) Scheme 2: In this CMT scheme three copies of the same message are sent and the receiver acknowledges reception as long as one of the copies is correctly received. This scheme is of speculative nature but is useful when the heavy-tailed traffic persists. If one of the copies can get through the channel quickly, then the service roundtrip time can be shortened. Assuming $\upsilon = 0.4$ the probability for a message transmission success is $\upsilon_3 = PR_1 + PR_2 + PR_3 = 0.936$, for $PR_1 = \binom{3}{1}\upsilon^2(1-\upsilon)$, $PR_2 = \binom{3}{2}\upsilon(1-\upsilon)^2$ and $PR_3 = (1-\upsilon)^3$. Then, the resultant $ANT_{Scheme-2} \approx \frac{1}{(1-\rho_{path})} \approx \frac{1}{\upsilon_3} \approx \frac{1}{0.936} \approx 1.07$ implies a shorter RTT than scheme 1.

The comparison between the two schemes above indicates that the designer should strike a balance between the number of multiple copies and acceptance criterion for an efficacious CMT scheme.

2.3 ACTIVE QUEUE MANAGEMENT

In fact, prevention of TCP channel congestion by the sender's effort alone may not be effective in many cases. This led to the proposal of the Active Queue

Management (AQM) concept in the RFC 2309 for system-level receiver/router applications [Braden1998]. In this concept if a router detects that its reception buffer is likely to overflow, then it throttles the sender to slow down transmission voluntarily [Chatranon2004]. A router starts the throttling process by sending "choke" packets. If the sender does not respond to the throttling, then the router drops the incoming packets to facilitate smooth passage of those already queued. The message dropping process may follow different strategies, for example, "drop from front" [Lakshman1996]. In fact, dropping messages as a congestion and buffer overflow prevention mechanism is deleterious. That is why the IETF (Internet Engineering Task Force) proposes to use the RED (Random Early Discard) algorithm for AQM purposes in the RFC 2309 [Braden1998]. The subsequent analyses of RED by different researchers, however, found that it was unstable and this led to the different RED mutants (e.g. the algorithmic ones FRED (Fair RED) [Kim1998], DS-RED [Zheng2001], LRU-RED [Reddy2001], M-RED [Koo2001], REM [Athuraliya2001]) and the intelligent non-RED-based versions (e.g. Fuzzy-PI [Ren2002]; P for proportional control and I for integral control). Floyd and Jacobson call those routers in packet-switching networks that adopt the RED algorithm the Random Early Detection Gateways [Floyd1993].

2.4 USING BACKUP CHANNELS

Internet based time-critical applications, which require communication service with guaranteed quality of service (QoS) in terms of timeliness and fault tolerance,

are emerging quickly. They include different areas such as video-on-demand, video-conferencing, and telemedicine. From the "average number of trials (ANT) to get a transmission success" point of view, fault tolerance measures can reduce the channel/connection error ρ_{path} and thus the ANT value leading to a shorter RTT that can satisfy the QoS requirements. The ρ_{path} value actually encompasses different errors that could occur along the C/S path including partial failures and router buffer overflow. Therefore, some researchers argue that the ordinary TCP channels for best-effort traffic (i.e. no real-time constraints) may not be good enough. They suggest the use of backup channels [Kris2003, Shin2000] to create more dependable real-time protocols. An ordinary *primary* TCP channel together with a backup connection makes a more dependable connection referred to as DP-channel/connection here. How dependable the backup channel is depends on the resources being reserved. For example, [Han1998] proposes to reserve dedicated system resources as support for backup channels. This means that the backup dependability depends on the quality of the reserved resource, which is tied with the cost. The DP-connection is a basically a *primary-backup* approach that involves three basic steps: i) establishing the primary channel and backup, ii) detecting channel problems (e.g. network congestion, partial failure, etc.), and iii) channel switching from primary to backup. Some researchers argue that a backup channel should not be dedicated but shared for better system throughput. The sharing can be as follows: i) *backup multiplexing* (BM) that lets two or more primary channels share a backup, ii) *backup-primary multiplexing* (BPM) that lets the backup be a temporary ordinary primary channel without real-time

constraints. The pros and cons of the three basic schemes above are evaluated as follows:

a) Dedicated scheme: The dedicated backup is expensive but it can better support the primary channel to provide a high-quality DP-channel. Unless the network always has extra resource to be reserved for backup channels the reservations can consume system resources rapidly. As a result the bandwidth utilization is impoverished leading to poor system throughput.

b) BM scheme: If a backup channel is shared, the dependability of the DP-connections (i.e. primary channel plus backup) can be evaluated by equation (2.4.1) [Kris2003, Shin2000]. The parameter *n* is the number of sampling operations within the period/window of interest T. PTD_i is the "*product of the* i^{th} *sampling period and the number of DP-connections*", and PTC_i is the "*product of the* i^{th} *sampling period and the total number of primary and backup channels involved*". Table 2.4.1 illustrates some statistics for the two separate sampling operations within the interval T. In the 1st sampling operation that lasted 10 time units the original primary connection 1 in the DP-channel 1 shared the backup in the time-multiplexing manner with the original primary 2 connection. The backup channel was used 5 times to support either DP-channel 1 or DP-channel 2. In the 2nd sampling operation the backup channel was used for similar support 4 times. By equation (2.4.1) the dependability of the DP-

connections in T is $\frac{(2*10+2*8)}{(2*10+5*10+2*8+4*8)} = \frac{36}{118} \approx 0.31$ or 31%.

The important connotation from equation (2.4.1) is that the dependability is 100% if the backup is not invoked at all.

$$Dependability = \frac{\sum_{i=1}^{n} PTD_i}{\sum_{i=1}^{n} PTC_i} \text{.....(2.4.1)}$$

i^{th} sampling operation (duration)	*number of connections*	*connection type*	*remarks*
1st (10 time units)	1	original primary 1	as main part of DP-channel no. 1
	5	backup	shared by DP-channels 1 and 2
	1	original primary 2	as main part of DP-channel no. 2
	3	backup became a temporary "ordinary" channel	
2nd (8 time units)	1	original primary 1	as main part of DP-channel no. 1
	4	backup	shared by DP-channels 1 and 2
	1	original primary 2	as main part of DP-channel no. 2
i.e. $n = 2$ for equation (2.4.1)	2	backup became a temporary "ordinary" channel	

Table 2.4.1 The connections sampled in two sampling periods within T

c) BMP scheme: In this scheme the backup channel can become a temporary "ordinary" channel with no real-time quality as illustrated in Table 2.4.1. The

argument is that when the backup channel is not needed as backup it may be used temporarily to boost up the communication throughput. The drawback of this approach is that it also makes the DP-channels temporarily insecure/undependable. There are still many issues that need to be addressed for designing efficacious BMP schemes, for example:

i) When should a backup be allowed to become a temporary non-real-time channel?

ii) How could an "ordinary" channel be terminated gracefully when a DP-channel needs it for support suddenly?

In general it is difficult to harness the TCP channel RTT for time-critical applications because of the sheer size and heterogeneity of the underlying network. This leads to the use of backup channels for more dependability [Kris2003, Shin2000]. This approach helps shorten the service RTT in critical applications over the Internet. As a result tasks can be meaningfully executed before the deadline [Stankovic1998]. This is absolutely necessary for *hard* and *firm* real-time applications and less stringent for the *soft* type.

2.5 DYNAMIC BUFFER SIZE TUNING

The system-level channel congestion prevention methods cannot prevent user level buffer overflow from happening. Firstly, the server may serve many different clients simultaneously, and the merged traffic in the asymmetric rendezvous can

create overflow because of its high rate. Secondly, the traffic pattern embedded in the merged traffic is unpredictable and this can cause overflow because the pattern affects the efficacy and stability of the reception buffer. One effective solution is dynamic buffer tuning, which means tuning the reception buffer size adaptively on the fly so that the buffer length always covers the queue size and thus eliminates any chance of buffer overflow [Wong1999A, WongHC2001, Wong2002GAC]. The first dynamic buffer size tuning scheme, "P+D" scheme, was proposed by Wong and Dillon [Wong1999]. It is based on the concept of proportional (P) and the derivative (D) control. The two parameters used in this approach are: the ratio of "queue length over the buffer length" and the rate of change of queue length over time. The instability of the "P+D" controller in real application led to the development of the PID controller or PIDC [Ip2001], which is the "P+D" approach augmented by integral (I) control. So far all the known dynamic buffer tuners, namely, PIDC, GAC, FLC and NNC are aimed at user-level applications.

2.6 PREVIOUS MPHIL RESEARCH

In my previous MPhil research four original dynamic buffer overflow controllers/tuners were proposed, namely, the PID Controller (PIDC), the GAC, the FLC and the NNC. Figure 1.3.1 illustrated the course of evolution from the PIDC to the NNC, and in fact, the intelligent tuners, GAC, FLC, and NNC were intended to preserve the PIDC merits minus its shortcomings [Ip2001]. In the rest of this section

each of the four tuners will be concisely presented, and their performance in terms of their control cycle time will also be compared.

2.6.1 THE PID CONTROLLER (PIDC)

Buffer size tuning, similar to industrial control processes, may involve P, I and D control elements [Karray2002]. In the PIDC the proportional (P) control is the ratio of "*queue length over buffer length (QOB)*" to predict the chance of overflow. The rate or derivative (D) control, which decides how fast the buffer would become full, is the rate of change in the queue length (i.e. dQ/dt). The integral control is the history of changes in the queue length. The P, I and D control elements in the PIDC should be construed on a conceptual basis [Wong2000A], They are different from the traditional meanings in process control theory because the PIDC does not consider the feedback system gain. The PIDC is formed by adding integral control to the previous "P+D" tuner, which was the first of its kind [Wong1999A]. Although the "P+D" tuner worked well in simulations with selected datasets, it failed in real situations over the Internet. The main cause of failure is the "*hard-coded*" nature of the "P+D" control parameters. Since these parameters do not register new knowledge the controller does not have enough power to anticipate what may happen in the future proactively. The use of the history in predicting the trend of change is the basis of integral (I) control in a general sense. The PIDC development needs to address the following two issues:

a) Would the P+D perform better if I control is incorporated to make it a PID controller (PIDC)?
b) How should the I control be implemented, especially when direct data measurement is the basis of the PID control process?

The need for direct data measurement and the impossibility of monitoring the overwhelming number of dynamic network parameters in the sizeable Internet directly require a new technique. This led to the consideration of using the IEPM (Internet End-to-End Performance Measurement) approach [Cottrel1999]. The core idea in this approach is to gauge the channel dynamics by measuring its mean roundtrip time (RTT). The IEPM concept is relatively new [Prasad2003, Barford2004], and the only known IEPM method that has its accuracy independent of any type of distribution/waveform is the *Convergence Algorithm* (CA) [Wong1999B]. The CA was successfully implemented and tested as the M^2RT [Wong2001] macro IEPM tool. The waveform independent property of CA/M^2RT is attributed to the fact that it is derived from the *Central Limit Theorem* [Aloisio1980]. In its macro version the M^2RT must be installed at two nodes that represent the ends of a logical channel. The micro implementation of the CA is known as the M^3RT [Ip2002], which runs as a logical object that can be invoked for service by message passing anytime and anywhere.

The accumulated PIDC experience, however, shows that this tuner has two distinctive shortcomings (Figure 2.6.1.1):

a) The queue length can get dangerously close to the buffer length, and in very serious dynamic traffic perturbations there could be a chance of overflow.

b) The buffer length lingers at the high value after every correction and this wastes valuable memory and impedes system performance.

The desire to eliminate these shortcomings led to the introduction of the safety margin Δ concept. The three intelligent tuners, GAC, FLC and NNC, adaptively maintain Δ on the fly about the chosen reference (i.e. "0") of the $\{0,\Delta\}^2$ objective function. The difference between the controlled buffer length and the current queue length should stay inside the user-specified tolerance band of $\pm\Delta$ about the reference point. This reference point is the "queue length over buffer length" ratio chosen by the user, namely, QOB_R.

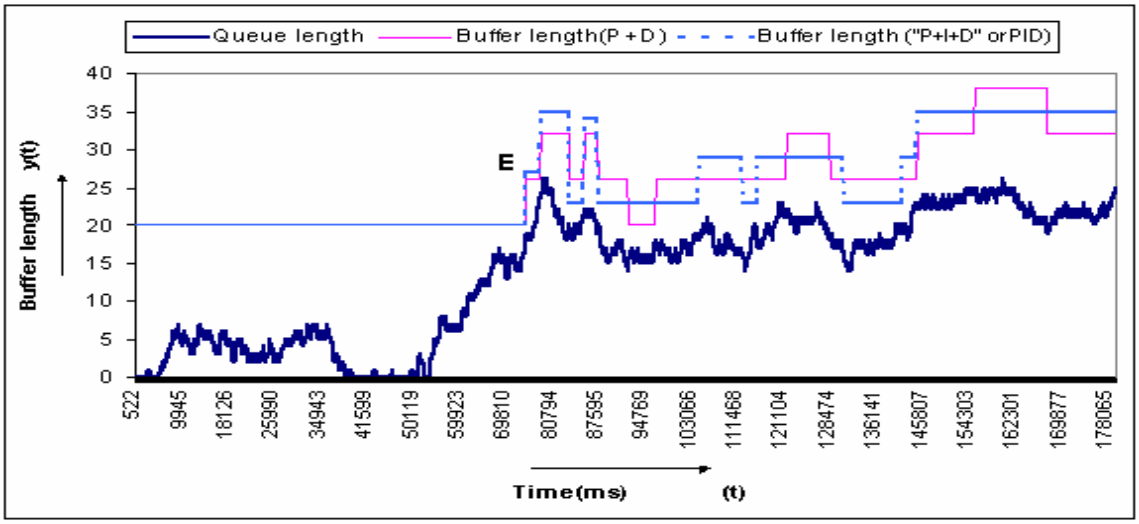


Figure 2.6.1.1 Illustration of the PID shortcomings

Figure 2.6.1.1 shows the following: a) the trace of queue length values for the experiment, b) for this trace the "P+D" tuner buffer overflows at point E, and c) the

PIDC working with same trace produces no overflow at all. In fact, the PIDC consistently produces no overflow with all the deployment cases, but its two shortcomings are always present.

2.6.2 THE GENETIC ALGORITHM CONTROLLER (GAC)

Genetic algorithms are a form of evolutionary computing [Michalewicz1996] that mimick natural evolution in the reproduction process, including chromosome crossover and mutation of genes. From the perspective of conventional or algorithmic PID control genes are the threshold values. The goal of the GAC is to eradicate the shortcomings of the algorithmic PIDC by adjusting the set of thresholds. The safety margin concept Δ and the $\{0,\Delta\}^2$ objective function were first introduced in the GAC proposal and development [Wong2002GAC]. Conceptually the GAC is the "PIDC plus GA plus the $\{0,\Delta\}^2$ objective function" combination.

Since in the buffer length tuning process the GA treats the PIDC parameters as genes every parameter set is a chromosome in the GA context. The buffer size estimated by the refined PIDC is fed to the objective function $\{0,\Delta\}^2$ to check its fitness. In effect, $\{0,\Delta\}^2$ is the fitness function. The aim is to ensure that the difference between the controlled buffer length and the current queue length stays within the $\pm\Delta$ tolerance band about QOB_R. If there is an indication that the criterion of $\pm\Delta$ may not be satisfied, the GA immediately reproduces new chromosomes by mutation and crossover. It then selects the fittest chromosome to replace the existing set of PIDC parameters as a refinement process. By doing so the GA tries to prevent

the QOB deviating from QOB_R by more than $\pm\Delta$. The operation of the GAC, which is a GA-augmented PIDC, is shown in Figure 2.6.2.1.

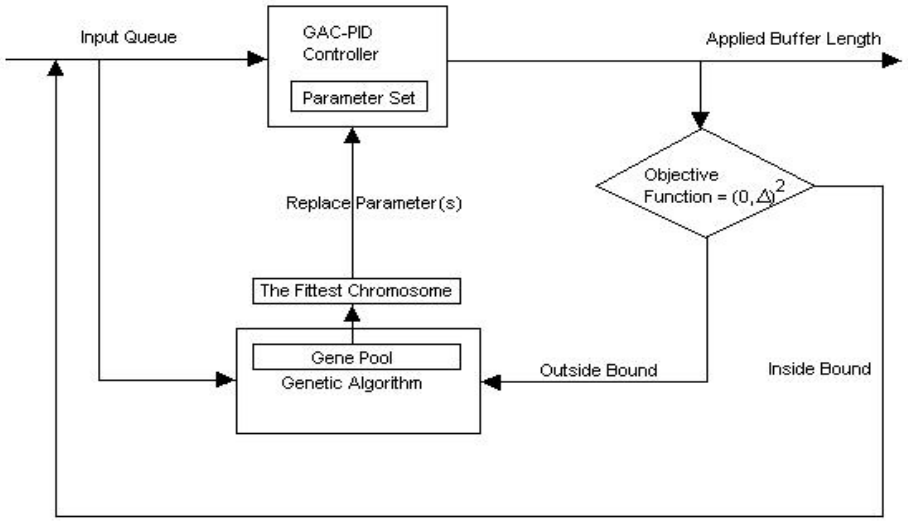


Figure 2.6.2.1 The GAC model for marginal buffer control

It was observed in different tests that oscillations might occur right after replacements with new chromosomes, and this leads to subsequent system instability and occasional buffer overflow. A solution to alleviate such vicious oscillations is to give the GAC enough time to adjust to the new parameters. We call this grace period the *adaptation time window* (ATW). The subsequent deeper analysis shows that the ATW helps but cannot eliminate the chance of buffer overflow at all. The analysis of this phenomenon as part of my PhD research overview had revealed that this is caused by the very nature of the GA not to guarantee the global-optimal solution of the *hyperplane* [Mitchel1999]. In the GAC the PIDC and the GA mechanism work in parallel with the same data input. The chromosome is made up of two different sets of

thresholds, namely, QOBL (Lower) and QOBU (Upper). The GA logic is illustrated in the flowchart shown in Figure 2.6.2.2.

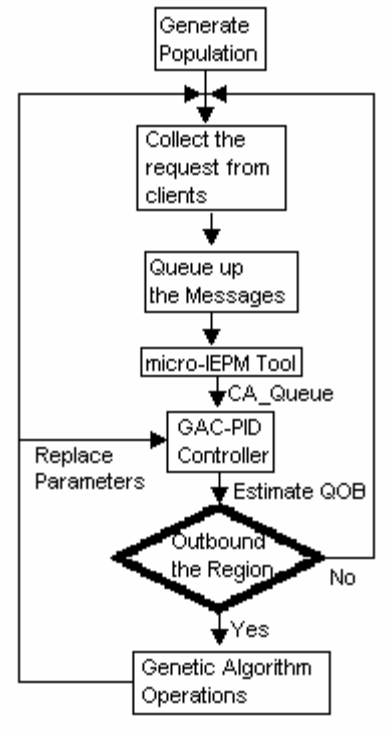


Figure 2.6.2.2 The GA logic flowchart

Figure 2.6.2.3 shows that for the given queue length trace the GAC eliminates the shortcomings of the PIDC (non-GA) by yielding more responsive buffer overflow control. For the same trace, however, it also produced an overflow as shown in Table 2.6.2.1. The overflow occurred at the time point of 548980 after chromosome replacement at the time point 547781.

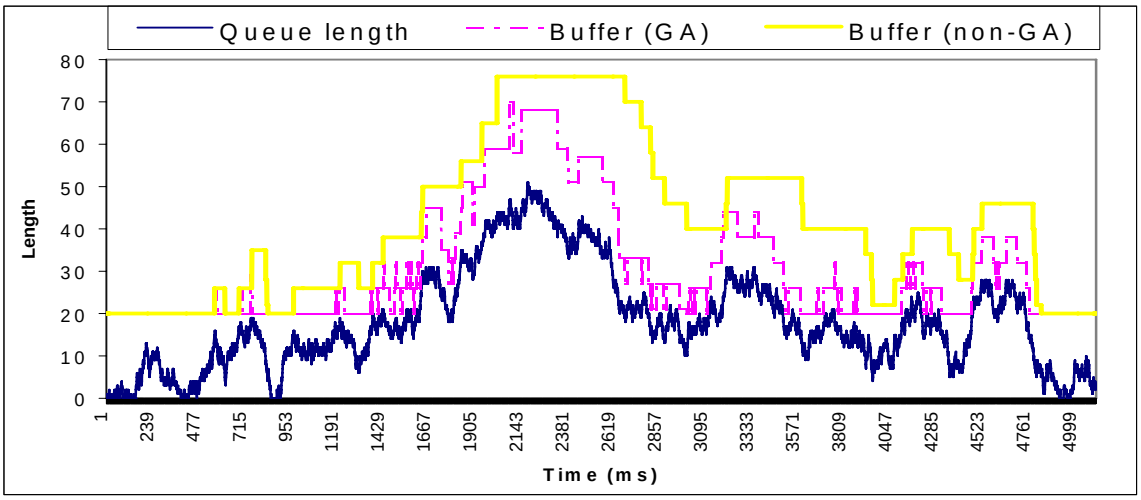


Figure 2.6.2.3 The GAC yields a more responsive result than the PIDC

Time	Queue length	Buffer (GA)	Buffer (non-GA)	RLQI (GA)	RLQD (GA)	RLQI (non-GA)	RLQD (non-GA)	Action
588	0	20	20	0.716605	0.574195	0.716605	0.574195	
1174	1	20	20	0.716605	0.574195	0.716605	0.574195	
1379	0	20	20	0.716605	0.574195	0.716605	0.574195	
					⋮			
546589	18	20	26	0.856595	0.583624	0.716605	0.574195	
547178	19	20	26	0.856595	0.583624	0.716605	0.574195	
547781	20	20	26	0.856595	0.583624	0.716605	0.574195	REPLACE
547941	19	20	26	0.823966	0.539837	0.716605	0.574195	
548374	20	20	26	0.823966	0.539837	0.716605	0.574195	INSIDE_ADAPTATION_PERIOD
548980	21	20	26	0.823966	0.539837	0.716605	0.574195	INSIDE_ADAPTATION_PERIOD
549391	20	26	32	0.823966	0.539837	0.716605	0.574195	
549509	19	26	32	0.823966	0.539837	0.716605	0.574195	
549587	20	26	32	0.823966	0.539837	0.716605	0.574195	

Table 2.6.2.1 A record of buffer overflow after chromosome replacement

2.6.3 THE FUZZY LOGIC CONTROLLER (FLC)

Although the GAC produces occasional buffer overflow it eliminates the PIDC shortcomings completely and confirms that the $\{0,\Delta\}^2$ objective function is a sound

and powerful concept. The FLC proposal represents the quest for a better intelligent model that can: a) work with the PIDC as a component minus its shortcomings, b) use the $\{0,\Delta\}^2$ objective function as the operational basis, and c) produce no overflow at all. In this sense the FLC should be more powerful and accurate than the GAC. The FLC conceptual framework is the "PIDC plus fuzzy logic plus the $\{0,\Delta\}^2$ objective function" combination. The fuzzy logic divides the PIDC control domain into many smaller fuzzy control regions (e.g. Table 2.6.3.1) and supports each of them with a predefined fuzzy rule or a "*don't care*" state. The "*don't care*" state requires no action/computation and in this way it offsets the FLC computation complexity and reduces its control cycle time. Therefore, the FLC is a *fuzzy region based* (FRB) approach [Berkan1997, Zadeh1994]. The control domain, which now consists of many fuzzy regions, is known as the *fuzzy knowledge base*. The adaptive adjustment of the buffer length, by addition or subtraction, depends on the current fuzzy region of operation. In effect, the original algorithmic PIDC has only two fuzzy regions if compared to the FLC approach. The "*don't care*" is marked by X in the FLC[4x6] design shown in Table 2.6.3.1.

QOB		dQ/dt					
		NL	NM	NS	PS	PM	PL
0.7 0.8 0.9	ML	-	-	-	-	-	-
	L	-	-	X	X	+	+
	G	-	-	X	X	+	+
	MG	+	+	+	+	+	+

Table 2.6.3.1 A FLC [4x6] design example

The "*dot*" in Table 2.6.3.1 marks the QOB_R, which in this case is equal to 0.8 (80%). The FLC experimental result presented in this section is based on this design,

and this implies Δ to be 0.2 or 20%. The linguistic variables (representations), which are used for the fuzzy regions of the FLC design in Table 2.6.3.1, are defined as follows:

a) For the Ratio of Queue Length Over Buffer Length (QOB)

- ML - *M*uch *L*ess than optimal point
- L - *L*ess than optimal point
- G - *G*reater than optimal point
- MG - *M*uch *G*reater than optimal point

b) For the Rate of change of queue length (dQ/dt)

- NL - *N*egative *L*arge than optimal point
- NM - *N*egative *M*edium than optimal point
- NS - *N*egative *S*mall than optimal point
- PS - *P*ositive *S*mall than optimal point
- PM - *P*ositive *M*edium than optimal point
- PL - *P*ositive *L*arge than optimal point

The control action to be taken by the FLC depends on the two input parameters, namely, QOB (i.e. proportional or P control) and dQ/dt (i.e. derivative or D control). The three possible FLC decisions/actions are: a) *Addition or* "+", b) *Subtraction or* "-" and *don't care or* "*X*". The *X* state prevents unnecessary oscillation in the buffer length control process. The *quantum* for *addition (*buffer elongation*)* or *subtraction (*buffer shortening*)* is still computed by the refined PIDC mechanism, which is a component of the FLC. Refinement here means that the operation of the PIDC algorithm depends on the fuzzy region that the controller is currently operating

in. The FLC Java prototype for the FLC[4x6] design shown by Table 2.6.3.1 has the following fuzzy rules, which moderate the integral control represent by RICM (*Refined Integral Control Mechanism* [Lin2002FLC]):

Rule 1: If (QOB is ML) AND (dQ/dt is NL) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 2: If (QOB is ML) AND (dQ/dt is NM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 3: If (QOB is ML) AND (dQ/dt is NS) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 4: If (QOB is ML) AND (dQ/dt is PS) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 5: If (QOB is ML) AND (dQ/dt is PM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 6: If (QOB is ML) AND (dQ/dt is PL) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 7: If (QOB is L) AND (dQ/dt is NL) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 8: If (QOB is L) AND (dQ/dt is NM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 9: If (QOB is L) AND (dQ/dt is NS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$

Rule 10: If (QOB is L) AND (dQ/dt is PS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$

Rule 11: If (QOB is L) AND (dQ/dt is PM) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

Rule 12: If (QOB is L) AND (dQ/dt is PL) Then Action is "+" (Addition) AND $L_{new} = L_{old}$ + RICM

Rule 13: If (QOB is G) AND (dQ/dt is NL) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 14: If (QOB is G) AND (dQ/dt is NM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - RICM

Rule 15: If (QOB is G) AND (dQ/dt is NS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$

Rule 16: If (QOB is G) AND (dQ/dt is PS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$

Rule 17: If (QOB is G) AND (dQ/dt is PM) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

Rule 18: If (QOB is G) AND (dQ/dt is PL) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

Rule 19: If (QOB is MG) AND (dQ/dt is NL) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

Rule 20: If (QOB is MG) AND (dQ/dt is NM) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

Rule 21: If (QOB is MG) AND (dQ/dt is NS) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

Rule 22: If (QOB is MG) AND (dQ/dt is PS) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

Rule 23: If (QOB is MG) AND (dQ/dt is PM) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

Rule 24: If (QOB is MG) AND (dQ/dt is PL) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + RICM

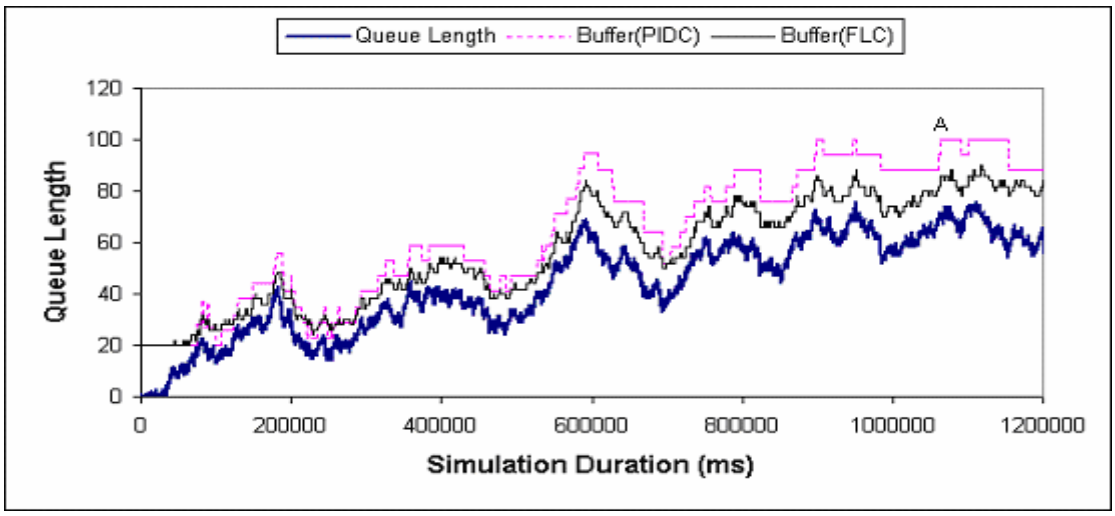


Figure 2.6.3.1 A case of performance comparison between the PIDC and the FLC ($QOB_R = 0.8, \Delta = 0.2$)

The FLC always yields more a responsive buffer tuning operation than the PIDC minus the latter's shortcomings. Figure 2.6.3.1 is one case in which the FLC consistently maintains the safety margin Δ for the $\{0, \Delta\}^2$ objectivity function. Point A in Figure 2.6.3.1 indicates that the FLC has eliminated the PIDC shortcoming of staying at the high buffer length value (i.e. unused buffer space still being locked by the controller).

2.6.4 THE NEURAL NETWORK CONTROLLER (NNC)

Although the FLC preserves the PIDC merits minus its shortcomings, its convergence towards QOB_R can be oscillatory. The desire to attain a smoother QOB_R convergence for the $\{0, \Delta\}^2$ objectivity function led to the NNC proposal, which was also inspired by the success of applying the neural network (NN)

techniques in AQM (Active Queue Management) algorithms (e.g. [Aweya1998]). After some detailed preliminary investigation it was concluded that the NN approach should base on backpropagation (BP). The argument is that the BP approach is simpler and the NN controller can be trained efficiently with Δ as the teacher signal. Therefore, the NNC is proposed as a feed-forward BP perceptron [Lin2001NNC] with supervised training [Rumelhart1986], as shown in Figure 2.6.4.1. The NNC prototype's configuration consists of: a) a single input layer of 10 neurons, b) a single hidden layer of 20 neurons, and c) a single neuron in the output layer. The training with Δ is based on the Sigmoid function represented by $f(x) = 1.0/(1.0 + e^{-x})$. The activation energy (value) for the neurons in the hidden and output layers are computed respectively as follows:

a) Sigmoid($\sum$ InputActivation * weight (input-hidden))

b) Sigmoid($\sum$ OutputActivation * weight(hidden-output))

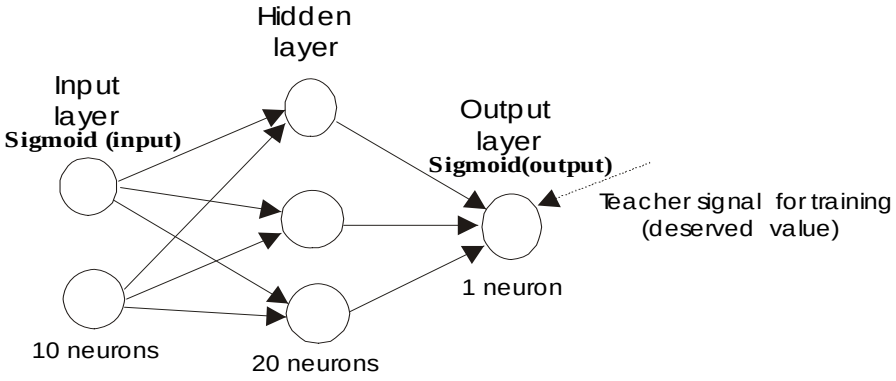


Figure 2.6.4.1 Backpropagation NNC control

The NNC has an input vector Q_{vector} of ten variables with the following properties:

a) *Nine queue-length samples*: They are sampled at equal time distances within the chosen renewal window or period of W. If we divide W equally into nine portions then each sample is denoted by Q_{tX} , where X is 1,2,…9.

b) *10^{th} element*: This is the queue length estimated by the M^3RT at the t9. That is, at the time point t9, which is the end point of the current W cycle, two samples are included, namely, $Queue_{CA\_estimate}$ and the queue length at that point. This means that M^3RT must run in parallel with the NNC as a logical entity.

The inclusion of $Queue_{CA\_estimate}$ (output from the M^3RT) is the basis for the argument that the NNC has the capability to proactively maintain the safety margin Δ. The rationale is that the main function of M^3RT is to predict the trend of the queue length distribution and therefore the instantaneous value of $Queue_{CA\_estimate}$ at t9 should reflect the next move of the queue length accurately. The output from the NNC is the predicted buffer length required to ensure that the queue is completely covered so that the safety margin Δ criterion will be met in the next W. The computation approach for the NNC is given by equations (3.4.1) and (3.4.2). The predicted buffer length $L(W+1)$ for the W+1 cycle is a function ($\int_{NN}(..)$ *is the symbolic representation*) of the variables vector Q_{vector} (W) and $Queue_{CA\_estimate}$ ($t9_W$).

$$L(W+1) = \int_{NN} [Q_{vector}(W), Queue_{CA\_estimate}(t9_w)] \ldots\ldots (3.4.1)$$

$$Q_{vector} = \{Q_{t1_W}, Q_{t2_W}, Q_{t3_W}, Q_{t4_W}, Q_{t5_W}, Q_{t6_W}, Q_{t7_W}, Q_{t8_W}, Q_{t9_W}\} \ldots..(3.4.2)$$

The NNC operation is divided into two phases, namely, *training* and *prediction*. The first is the training process for the BP approach to learn to respond properly by yielding the deserved value with respect to the teacher signal; in this case the teacher

is the given Δ. Before training starts the weights of the NN arcs are randomized. The error, which is the difference between the predicted output and the deserved value (DV) defined by the $(QOB_R - \Delta) \le DV \le (QOB_R + \Delta)$ range, should gradually decay as the learning process is progressing. The training is considered to be completed if the controlled output is consistently within the DV range. The simulation performance by the NNC and that by the "NNC+M^{3}RT", namely, the NNC supported by the M^{3}RT in the form of $Queue_{CA\_estimate}$ or Q_{10} element in the Q_{vector}, are depicted in Figure 2.6.4.2 and 2.6.4.3. M^{3}RT is synonymous with CA (*Convergence Algorithm*) because the former is the validated micro version of the CA *macro* Java implementation known as the M^{2}RT.

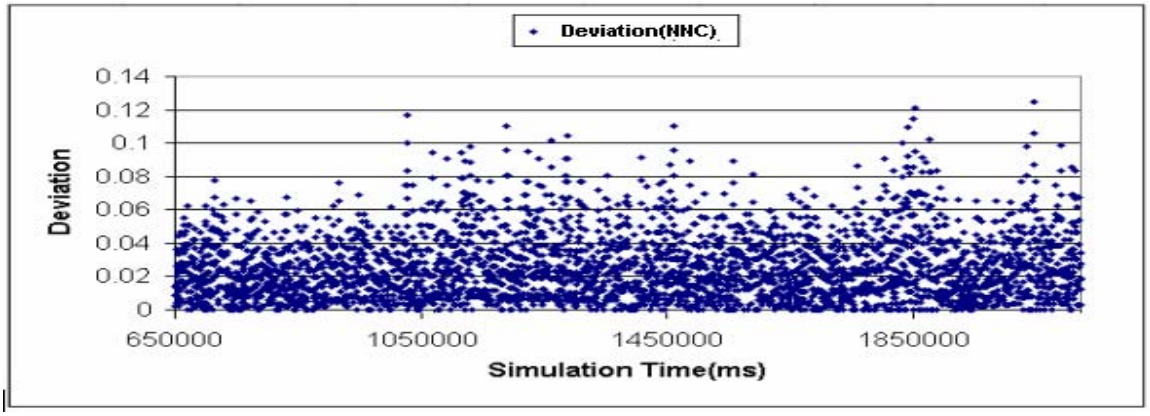


Figure 2.6.4.2 Deviation by NNC from objective function with QOB=0.8 – test case 1

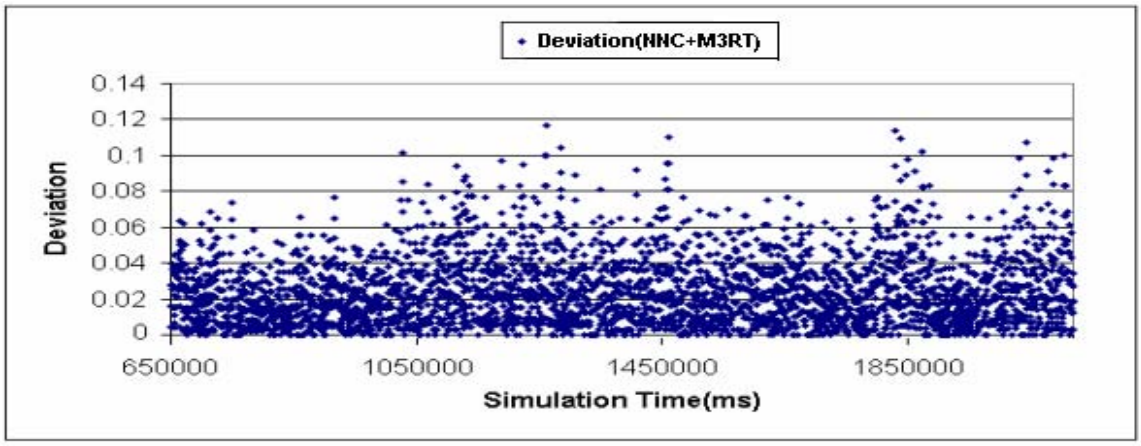


Figure 2.6.4.3 Deviation by (NNC+ M^3RT) from objective function, QOB=0.8 – test case 2

The average of the deviation of the NNC output is measured by the following equation: $\left[\sum_{i=1}^{k} | \Delta - QOB_i | \right] \Big/ k$. Table 2.6.4.1 (Δ = 0.2, sample size k = 7200) shows three cases out of the many simulations, which confirm that the "NNC+M^{3}RT" consistently has at least 5 percent or less of deviation from the QOB_R than the NNC working alone. Without the M^{3}RT incorporation the deviation can be as large as 25% in some cases, and our analysis indicates that this phenomenon is due to the fact that the knowledge from the last training of the NNC is not enough to deal with unexpected new situations. With the M^{3}RT presence the largest deviation by the "NNC+ M^{3}RT" is around only 15%. This is the result of the integral effect provided by the M^{3}RT convergence process.

Figure 2.6.4.4 compares the smoothness of convergence towards QOB_R by the PIDC, FLC and NNC (supported by CA/ M^3RT). The NNC and the FLC eliminate

the PIDC shortcomings and consistently maintain the safety margin Δ. The maintenance by the NNC is much smoother than the FLC.

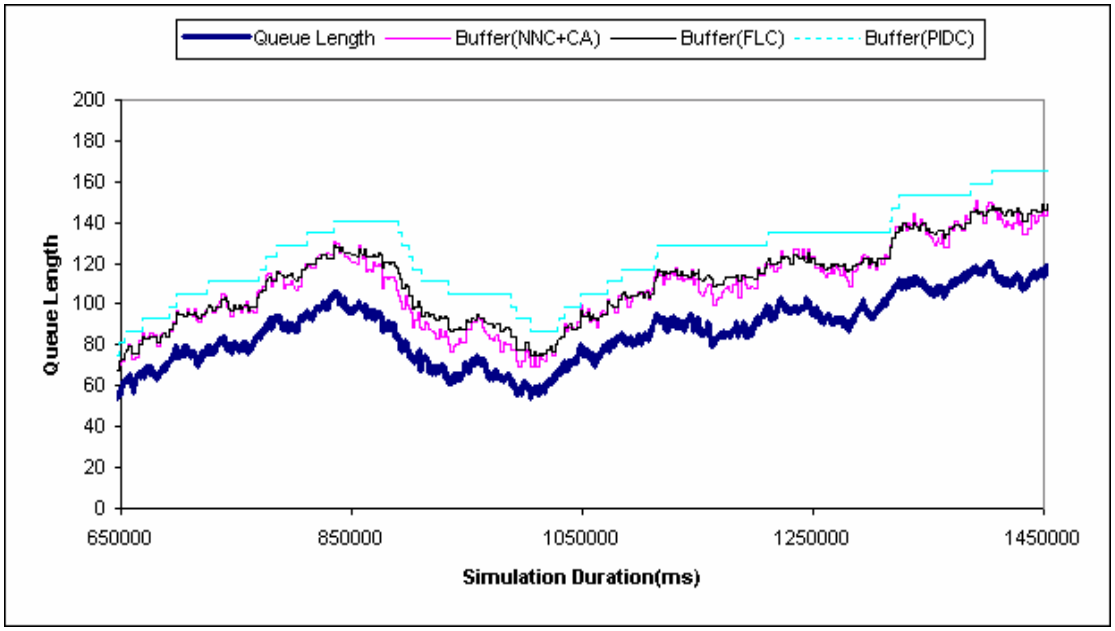


Figure 2.6.4.4 Comparing the PIDC, FLC and NNC ($QOB_R = 0.8$)

Controller	Case 1	Case 2	Case 3
NNC	0.02386	0.01853	0.02245
NNC+ M^3RT	0.02260	0.01655	0.02115
Performance improvement: $\frac{NNC-"NNC+M^3RT"}{NNC}*100\%$	5.28%	10.7%	5.8%

Table 2.6.4.1 Comparing three cases of deviations between NNC and "NNC+ M^3RT"

2.6.5 TIMING ANALYSES OF THE DIFFERENT CONTROLLERS

Timing analysis of the individual dynamic buffer tuners, namely, PIDC, GAC, FLC and NNC is an essential part of my MPhil research. A good dynamic buffer tuner should be quick and accurate. If its control cycle time is too long, then it may

yield deleterious effects because the computed remedy ends up correcting a long-passed spurious problem. The timing analysis is carried out with the *Intel's VTune Performance Analyzer* [VTune2002]. The control cycle time or controller execution time is measured in terms of the number of neutral clock cycles. Some of the results are presented in Figure 2.6.5.1 to 2.6.5.4. The clock cycles can be converted into the actual physical time for the chosen platform. For example, if the platform is operating at 100 MHz, the control cycle time of 500 clock cycles yields $500/(100*10^6) \approx 5$ micro seconds.

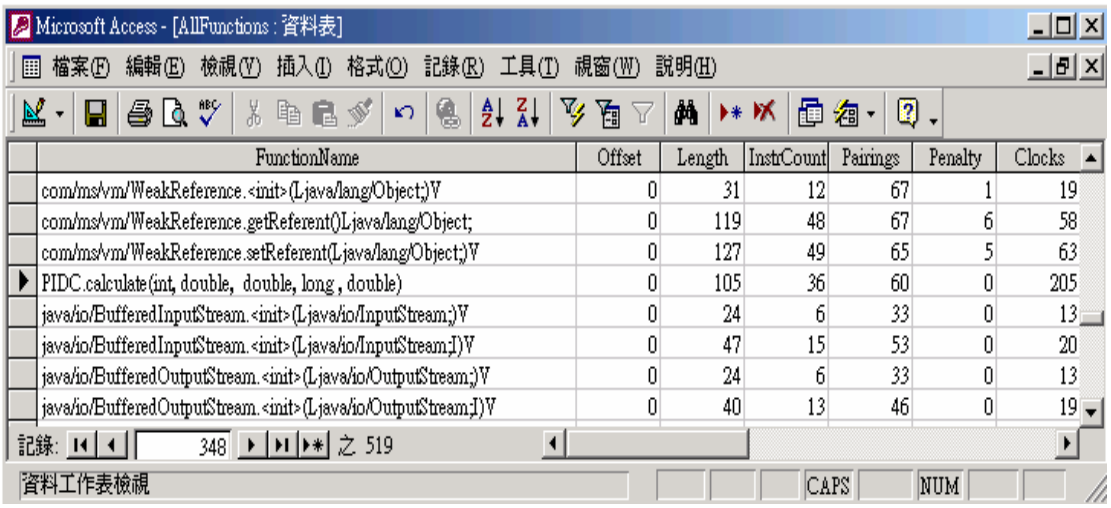


Figure 2.6.5.1 PIDC VTune Analysis (control cycle time is 205 clock/T cycles)

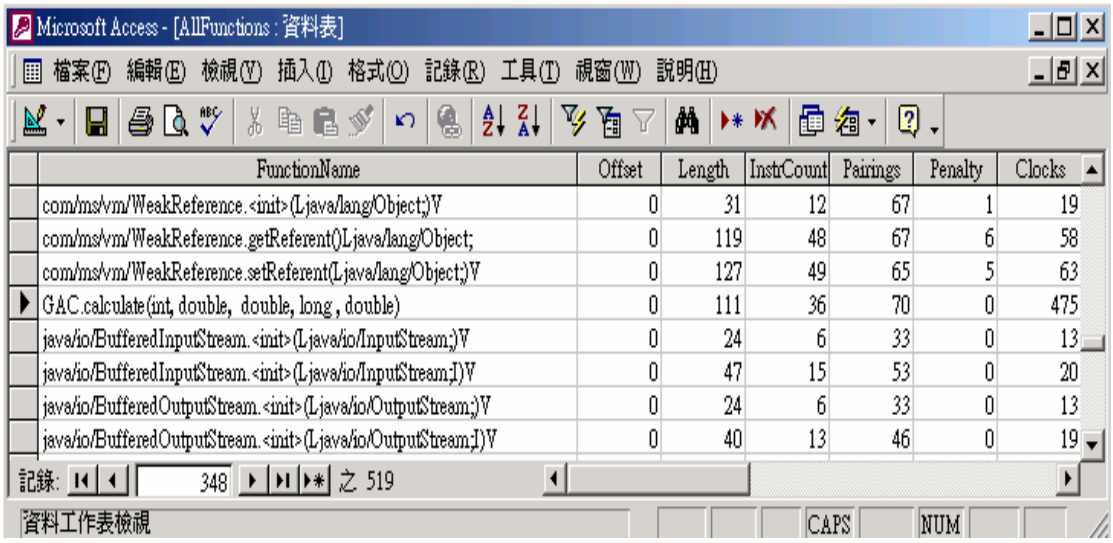


Figure 2.6.5.2 GAC VTune Analysis (control cycle time is 475 T cycles)

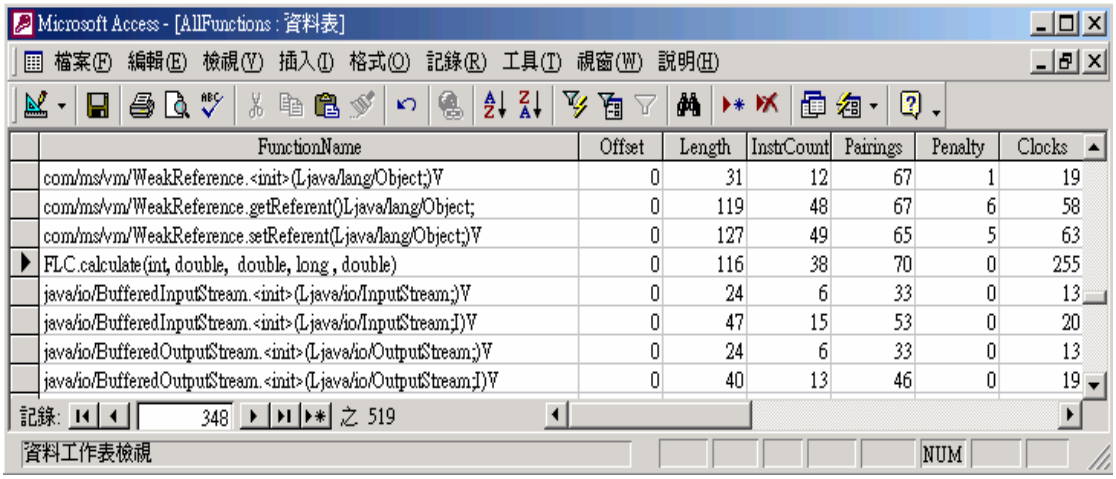


Figure 2.6.5.3 FLC VTune Analysis (control cycle time is 255 T cycles)

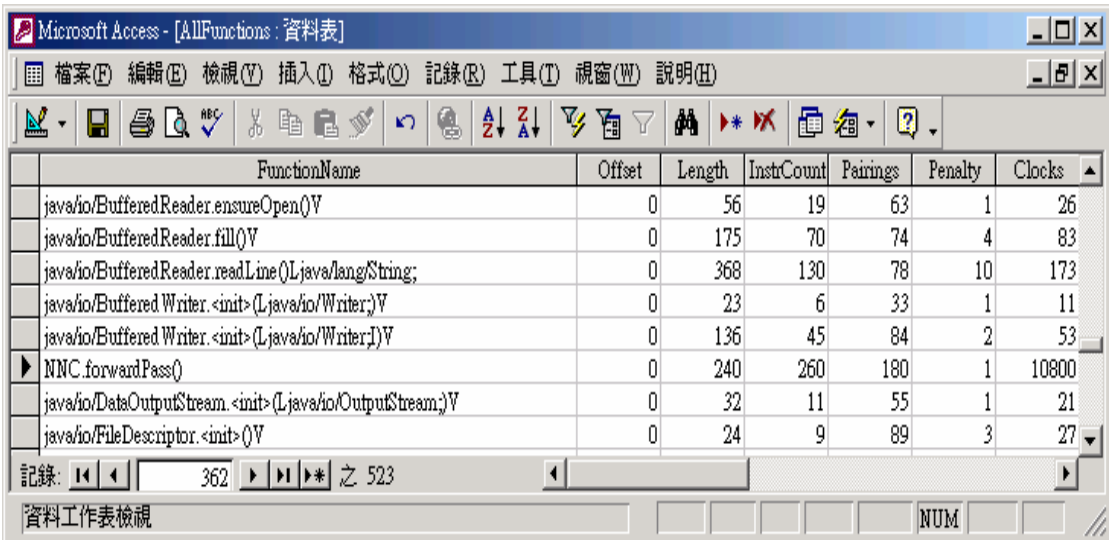


Figure 2.6.5.4 NNC VTune Analysis (control cycle time is 10800 T cycles)

Intel VTune [VTune2002] Timing Analyses for four buffer controllers with the Intel-Pentium III as the reference architecture						
Control models	Lines of Java code for controller implementation (Ln)	Average number of lines of code in Pentium III assembler	Clock/T cycles per assembly line (Pentium III 933MHz) (T)	*Average number of T cycles for* QOB_R *convergence* (NTC)	Measured average number of T cycles per control cycle (TCC)	Chosen architecture: Intel-Pentium III 933MHz (seconds)
PIDC	105	525	9	4725	205	2.1972E-07
GAC	111	555	9	4995	475	5.0911E-07
FLC	116	580	9	5220	255	2.7331E-07
NNC (*Input-Hidden-Output*: *10-20-1*)	240	1200	9	10800	10800	1.1576E-05

Table 2.6.5.1 Summary of the indicative control cycle times by the different controllers

2.7 CONNECTIVE SUMMARY

To recap, my MPhil research had achieved the following:

a) Four novel dynamic buffer overflow controllers for user-level applications were proposed; one algorithmic (i.e. the PIDC) and three intelligent ones (i.e. GAC, FLC and NNC).

b) The GAC was thoroughly tested and found to be unacceptable because it yields occasional buffer overflow.

c) The FLC was proposed and two designs, namely, FLC [4x4] and FLC [4x6] were tested. The results indicate that this direction is the right one because of the following: i) it eliminates buffer overflow completely, ii) its execution time comparable to the simpler PIDC's due the presence of the "*don't care*" state [Lin2002FLC], and iii) it always maintains the control output within $\pm\Delta$ about the chosen QOB_R reference, but its convergence to QOB_R can be oscillatory.

d) The success of using $\{0,\Delta\}^2$ as the operation principle and the desire to have a smoother QOB_R convergence led to the proposal of the NNC. The NNC differs form the GAC and the FLC because it does not include the PIDC as a component. The NNC, however, has a much longer control cycle time compared to the PIDC, GAC and the FLC and this is prone to deleterious effects. The argument is that by the time the remedy is computed the actual problem has already passed. Using the computed remedy to resolve a spurious problem may lead to

undesirable consequences or deleterious effects. The NNC prototype, which works by backpropagation with supervised training, has 10 input neurons, 20 neurons in the hidden layer, and one output neuron.

e) Timing analyses confirmed that the four novel dynamic buffer size tuning models are indeed suitable for time-critical applications over the Internet. The limit of application and accuracy is determined by the controller's mean control cycle time.

Meanwhile my MPhil research also left behind some important but unaddressed issues:

a) Does the Internet traffic impede the controllers' stability and accuracy? If so how can the impedance be alleviated or neutralized? In fact, the internet traffic can change without warning, for example, from LRD (long-range dependence) such as heavy-tailed and self-similar to SRD (short-range dependence) such as Poisson [Molnár1999]. Such changes may have a serious impact on the controllers' performance.
b) Is it possible to have an optimal (cost-effective) FLC design?
c) Is there a correlation between the accuracy and the number of neurons in the hidden layer of the NNC? In my PhD research finding such a correlation is called *sensitivity analysis*.
d) Is it possible to cut down the NNC control cycle time and lower the chance of deleterious effects?

CHAPTER 3 PROBLEM STATEMENT AND METHODOLOGY

3.0 INTRODUCTION

In this section the problem tackled in my PhD research will be explained. In order to achieve the research objectives in a qualitative manner and within the time constraints imposed on the duration of the project, the "*investigate & experiment & iterate* (IET)" methodology is adopted as the basis for evolution. This methodology helped my MPhil research a great deal and enabled me to produce useful and meaningful findings that led to several refereed journal and conference publications.

3.1 DEFINITIONS OF USEFUL TERMS

Client/server interaction - A client/server interaction has two levels: system and user. The system or router level includes all the activities within the TCP channel, and the user level includes the client and the server that interact over the TCP channel in the end-to-end manner.

Buffer - A finite memory space where objects queue up.

Network congestion - This happens when a router is inundated by a large volume of incoming packets and runs out of buffer space, leading to loss of packets and very slow or no response to the clients' requests

Queue – It is a series of requests waiting to be processed in a FIFO (first in first out) basis.

Adaptive buffer – It meets the "n <Buffer ≤ m" criterion with lower limit n ≥0 and upper limit m.

Adaptive/dynamic buffer size control – The buffer size is adjusted on the fly by the dynamic buffer size tuning function: $BufferSize_t = function(QOB_t, (dQ/dt)_t, M^3RT, ICM_t)$, where t indicates the time point. The parameters are defined as follows: QOB - ratio of queue length over buffer length, dQ/dt - rate of change of queue length, M^3RT - Micro Mean Message Response Time implementation, and ICM - integral control mechanism.

Traffic pattern – It is the traffic waveform/distribution, which may be SRD (short-range dependence) or LRD (long-range dependence).

Long-Range Dependence – A stationary process is long-range dependent if its autocorrelation function r(k) is nonsummable (i.e. $\sum_k r(k) = \infty$), applied only to infinite time series[Paxson1995].

Short-Range Dependence – A stationary process is short-range dependent if its autocorrelation function r(k) is summable (i.e. $\sum_k r(k) < \infty$).

Intelligent buffer controller – It uses soft computing techniques, for example, the FLC (Fuzzy Logic Controller) and the NNC (Neural Network Controller).

Roundtrip time (RTT): It is the delay/latency between the time that a client sends a request and gets the correct result from the server.

Packet loss – It happens in the transmission process (e.g. dropped by the receiver to prevent local buffer overflow as a congestion prevention measure).

3.2 PROBLEM DEFINITION

The scope of this research is dynamic buffer size tuning at the user level. The argument is that if the chance of overflow for the server's reception buffer in a TCP based client/server or C/S interaction path (Figure 1.01 and Figure 1.4.1) is eliminated, then the service roundtrip time or RTT can be shortened. In this sense the C/S path becomes more dependable and suitable for time-critical applications. My previous MPhil thesis had explored different possibilities of achieving reasonable user-level dynamic buffer size tuning, and as a result four novel dynamic buffer tuners were proposed, namely, the algorithmic PIDC and the intelligent/expert GAC, FLC and NNC [Lin2002]. These four models are unique because they operate with a

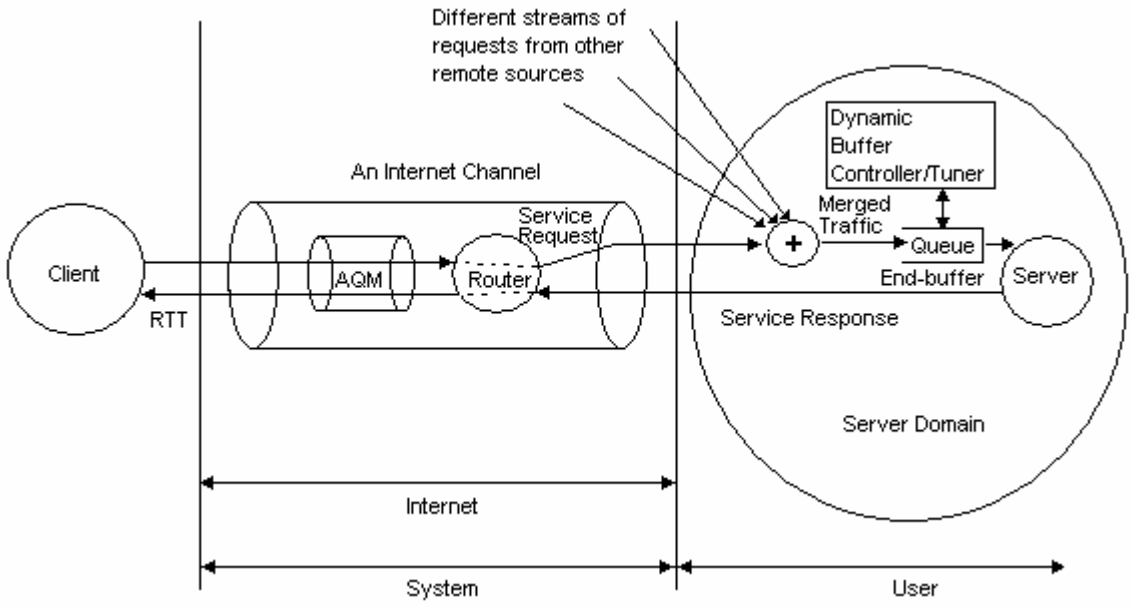
variable buffer length (VBL) as indicated in Figure 2.1.1. My MPhil research, however, left behind some important but unaddressed issues as follows:

a) Does the Internet traffic impede the controllers' stability and accuracy? If so how can the impedance be alleviated or neutralized? In fact, the internet traffic can change without warning, for example, from LRD (long-range dependence) such as heavy-tailed and self-similar to SRD (short-range dependence) such as Poisson [Molnár1999]. Such changes may have a serious impact on the controllers' performance.
b) Is it possible to have an optimal (cost effective) FLC design?
c) Is there a correlation between the accuracy and the number of neurons in the hidden layer of the NNC? In my PhD research finding such a correlation is called *sensitivity analysis*.
d) Is it possible to cut down the NNC control cycle time and lower the chance of deleterious effect?

Over a C/S path there are two levels of operations: system and user as shown by Figure 1.4.1, which is duplicated here to support a clearer explanation of the problem. The system level includes all the activities in the TCP channel, which inevitably has the collective error probability ρ (as explained in section 1.0 Introduction) due to the sheer size and heterogeneity of the Internet. There are existing mechanism that can prevent the network congestion, which results in router buffer overflow, loss of messages/segments, and widespread retransmission. The sender based mechanisms

tune the timeout windows to alleviate premature timeouts and/or the congestion window to reduce the sending rate and thus the amount of data across the network. Although the sender mechanisms have their contributions in cutting down the chance of network congestion, they are not powerful enough and have side effects. For example, the well-known AIMD (Additive Increase and Multiplicative Decrease) algorithm [Jacobson 1988] can impoverish bandwidth utilization in “long-fat-pipes”, which are high-bandwidth-high-latency networks [Wang2004]. This side effect is a relatively recent observation and since then different methods had been proposed to reduce it [Balakrishnan1997]. One of the counter measures is the AQM (Active Queue Management) approach proposed by the IETF’s RFC 2309. It allows the router to throttle the sender(s) once it has detected a strong likelihood of overflow in its reception buffer. The RED (Random Early Discard) algorithm is the candidate to do the job. The system-level congestion prevention activities cannot, however, prevent user-level reception buffer of a C/S path from overflowing. As shown in Figure 1.4.1, the client/server interaction at the user level is usually an asymmetric rendezvous, with the server simultaneously serving many different clients. At the periods of peak service demands the torrents of incoming request traffic merge to inundate the buffer to overflow easily. The cause is not only the high incoming traffic rate but also the pattern embedded in the merged traffic [Molnár1999]. If the server’s reception buffer on a C/S path overflows only after the system has dished out a large amount of resources to prevent network congestion and ensure the smoother passage for a message/segment/packet from the sender to the server, then the result can be disastrous. Therefore, it makes sense to propose dynamic buffer size tuners such as

the PIDC, GAC, FLC and NNC to eliminate the chance of user-level buffer overflow by ensuring the buffer length always covers the queue size. This needs the support of an efficient memory recycling system in the host where the server resides. Besides, dynamic buffer size tuning at the user level could also break down as the congestion problem is a persistent one. Conceptually the congestion prevention effort at the system level and the user-level dynamic buffer tuning operation together form a unified solution to stifle buffer overflow along a C/S path.



(Duplication of Figure 1.4.1 for clearer problem definition)

The Internet traffic pattern changes without warning, for example, from LRD (long-range dependence) such as heavy-tailed and self-similar to SRD (short-range dependence) such as Poisson. Since the traffic patterns and the sudden change from one pattern to another can have a serious impact on the queue dynamics and thus the dynamic buffer tuner performance, the buffer tuning mechanism should be able to

detect them and react within a reasonable time. Therefore, the real-time nature of the detection mechanism is important for it to be applied successfully.

3.3 PROBLEM STATEMENT

The aim of this PhD research is to address the following issues in-depth. The objectives include the following:

a) Study and define the impact of traffic on the stability and accuracy of the FLC and the NNC, and propose methods to counteract the negative impact effectively.
b) Explore and define the possible optimal range for the FLC design and implementation.
c) Define the correlation between the number of neurons in the NNC hidden layer and the control accuracy.
d) Propose a method(s) to optimize the NNC configuration to lower its control cycle time so that it becomes more suitable for time critical applications over the Internet.
e) Perform timing analyses of the improved/new FLC and NNC models to confirm that they are indeed suitable for time-critical applications over the Internet.

3.4 RESEARCH METHODOLOGY

The aim of any research is to address an issue or problem as thoroughly as possible. In the process it may involve the following: a) forming a conceptual framework, b) dissecting this preliminary framework into manageable pieces so that their functionalities and relationship can be investigated, c) developing the respective system supporting architecture so that the conceptual framework can be tested, verified and evaluated as a prototype, and d) improving the prototype continuously with new experimental results and observations. In the research process both backtracking and cross referencing are natural. To get meaningful research results within a given time frame, discipline is absolutely necessary in the course of action. This relies on choosing the correct research methodology, which is a totality of methods and tools that are appropriate for the problem domain. From the literature research activities can be classified in various ways in terms of their objectives and approaches. For example, the following types are summarized from the literature by [Nunamaker1991], a) basic and applied, b) scientific and engineering, c) evaluative and development, d) research and development, and e) “formulative” and “verificational”.

The system development approach involves theory building (development of new ideas and conceptual frameworks and models), experimentation (computer simulations to validate the underlying theory), and observation (case studies and formulation of hypotheses to be tested through experimentation). The work in this

PhD thesis clearly fits this approach, which supports system development and is therefore an information systems research methodology [Nunamaker1991]. The theory building part in the thesis consists of defining new types of buffer control strategies and formulating their mathematical structures. The experimentation involves both simulations and experimental studies on the Internet so that observations can be carried out for system validation, which is essential for proof of concept.

From another angle my PhD research is in the domain of computer science. According to [Philips1987] there are three basic types in this domain, namely, testing out, problem solving, and exploratory. From this perspective, this research is exploratory even though it ends up with a prototyping for rigorous testing and supporting future deeper research. The prototyping process concurs with the definition of system development by [Nunamaker1991]. Therefore, the concepts in [Nunamaker1991] and [Philips1987] complement each other. The workflow in my PhD research is top-down and includes literature search, problem statement definition, proposed solutions, and data collection and analysis. It is inappropriate, however, to apply the top down philosophy in a strict sense because the exploratory investigations at different stages may involve repetitive backtracking, re-orientation, and cross-referencing, to gain enough insight for going to the next step. For meeting the iterative or spiral behavior of the research activities the "*investigate & experiment & iterate* (IET)" methodology is adopted as the basis. The reason for adopting this basis

is that it helped me finish my previous MPhil research effectively and efficiently [Lin2002] and led to important findings and refereed publications.

Since the area of dynamic buffer overflow control over the Internet is relatively new, previous techniques and experience are limited in scope. It is inevitable that in my PhD research intermediary models would be proposed so that tests and experiments could be carried out to determine whether they are actually milestones. In this light the IET approach is natural for this project because experiments are continuously and repetitively needed to confirm the right direction for further actions. In the course of research it is only natural to have backtracking, concept refinement and modularization, and cross referencing. The basic IET methodology is conceptual, and this means that it can be realized in different ways. In this research the IET realization should gain from the experience of my previous MPhil project and be implemented as a roadmap (Figure 3.4.1). As the PhD research progresses the new experience gained would inspire inevitable changes to the IET hierarchy in the roadmap.

The first step in the IET approach is to divide the research problem is into sub-problems or tasks. The division is based on the knowledge gained from literature search and my previous experience. Each task is studied and executed carefully and the findings determine if a task should be further divided, eliminated, or combined with another extant one. As the research activities progress through a hierarchy of tasks and sub-tasks, backtracking and cross-referencing are sometimes necessary.

Backtracking is usually caused by insufficient research in one or several of the previous tasks higher in the hierarchy. Therefore, revisit(s) and more research are necessary with this/these previous task(s) to gain more insight so that the temporarily suspended task can continue and remains possible with proper re-orientation. Cross-referencing allows the present research stage to utilize the previous findings directly. In fact, tangible products may be produced by different tasks and sub-tasks, such as refereed journal and conference publications. This is clearly manifested by my previous MPhil research experience [Lin2002].

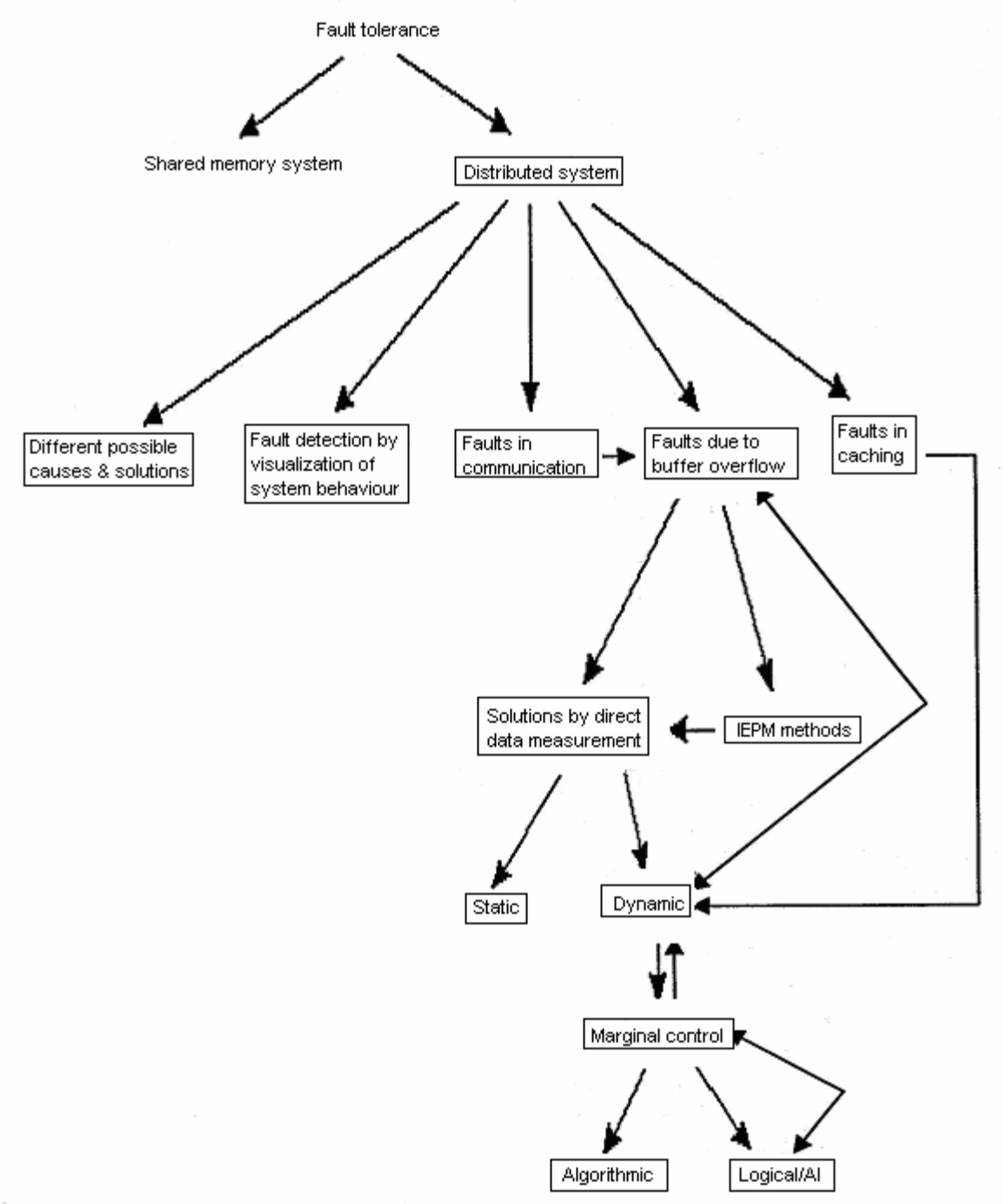


Figure 3.4.1 Realization of the IET methodology into a road map

CHAPTER 4 OVERVIEW OF SOLUTIONS

4.0 BACKGROUND

The area of my PhD research is directed at performance enhancement and fault tolerance in Internet applications. It is the continued, deeper investigation using my previous MPhil research experience as the basis. My MPhil research concentrated on how to use dynamic buffer size tuning to eliminate user-level buffer overflow at the receiver side. Figure 4.0.1 shows the end-to-end client/server interaction over a logical Internet TCP channel. This interaction in reality is *asymmetric rendezvous* because the server serves many different clients simultaneously (i.e. one-server-to-many-clients relationship). The request streams from different clients merge at the server's queue buffer. Every request has to wait there for its turn to be served. The queue length can grow very long during periods of peak demand, especially for a popular server. The overflow due to the merged traffic of different request streams that inundate the server buffer is called user-level overflow in the context of my PhD research. It differs from those along and inside the TCP channel. The dynamic buffer tuners proposed in my MPhil thesis include [Lin2002]: the algorithmic PIDC (i.e. *proportional* (P) plus *integral* (I) plus *derivative* (D) controller), and the intelligent trio, namely, GAC (Genetic *Algorithm Controller*), FLC (*Fuzzy Logic Controller*) and NNC (*Neural Network Controller*). This trio contributes to shortening the service roundtrip time (RTT) in the asymmetric rendezvous by eliminating the chance of user-level buffer overflow at the receiver side.

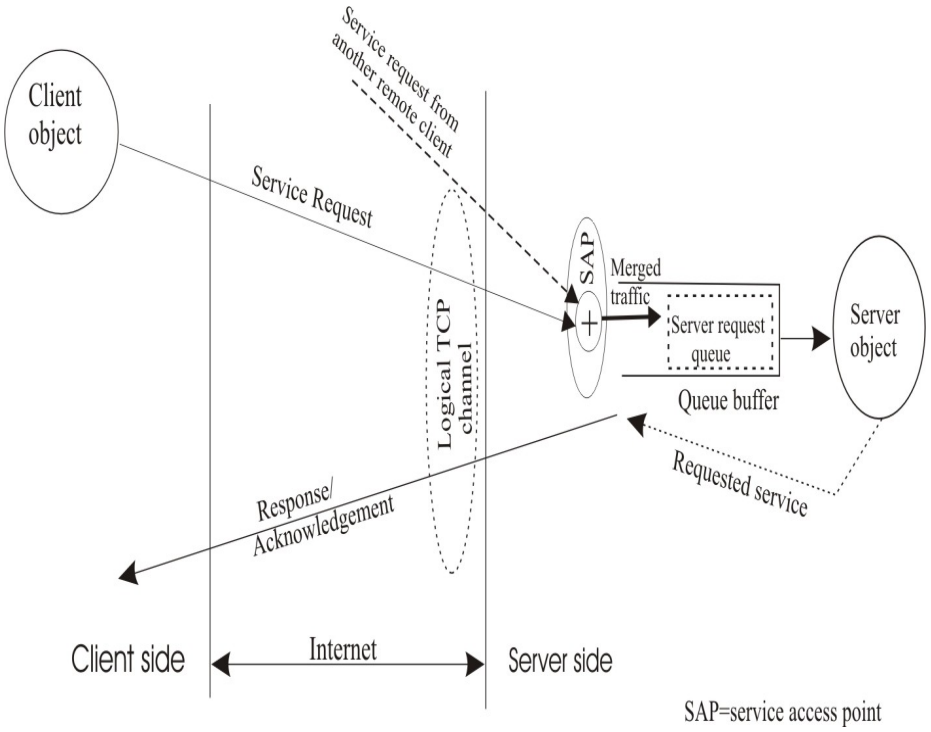


Figure 4.0.1 End-to-end client/server asymmetric rendezvous

Buffer overflow can occur at both the system/router level (all activities inside and including the logical TCP channel) and the user-level. Different strategies were proposed for reducing or preventing the chance of overflow at the system level by preventing network congestion. They are formally referred to as AQM (active queue management) algorithms by RFC2309 [Braden1998]. The more recent AQM algorithms use neural networks and their potential benefits had inspired the proposal of the NNC. Together with the system-level AQM mechanisms the user-level dynamic buffer size tuners proposed in my MPhil research, PIDC, GAC, FLC, and NNC, form a unified solution to stifle buffer overflow in an asymmetric rendezvous. If ρ encapsulates all the error probabilities that cause overflow in an asymmetric

rendezvous, the *average number of trials* (ANT) to get a successful transmission is $\sum_{j=1}^{K\to\infty} j[\rho^{j-1}(1-\rho)] \approx 1/(1-\rho)$. Therefore, eliminating the chance of user-level overflow reduces ρ and thus ANT. As a result it improves asymmetric rendezvous fault tolerance and shortens its RTT. The development history of the tuners in my MPhil research is summarized as follows:

a) PIDC – It was proposed to improve the efficacy of the first model, namely, "*P+D*" (i.e. proportional (P) plus derivative (D) controls). The "*P+D*" aims at eliminating user-level buffer overflow by dynamic buffer size tuning, which adaptively ensures that the buffer length always cover the queue length. It however produces overflow in real-life deployments because of the unrealistic expectation of using a set of static parameters to cover the whole spectrum of system dynamics. When integral (I) control is added to the "*P+D*" model the novel PIDC is formed. The PIDC always eliminates the chance of user-level buffer overflow despite its two shortcomings: a) it locks up unused buffer memory and this affects the overall system performance, and b) it does not have a safety margin and therefore the queue length can get dangerously close to the buffer length threatening overflow during peak demand periods. The desire to eliminate these two shortcomings prompted the investigation into the use of soft computing techniques.

b) GAC – It is basically the combination of "*PIDC plus* $\{0,\Delta\}^2$ *objective function plus genetic algorithm (GA).* The GA moderates the PIDC control process to make sure that it always stays within the Δ safety margin about the chosen reference represented symbolically by "0". The GAC eliminates the PIDC's shortcomings but produces rare user-level overflow. The reason is that the GA, similar to other evolutionary computing approaches, does not guarantee the *global-optimal solution* in the *solution hyperplane* [Mitchel1999].

c) FLC – It represents the desire and effort to eliminate any user-level overflow and preserve the GAC merits. The FLC is basically the following combination: "*PIDC plus fuzzy logic plus* $\{0,\Delta\}^2$ *objective function*". The FLC is more stable and faster than the GAC, and most important of all it does not produce any buffer overflow.

d) NNC – The success of some AQM algorithms at the experimental level inspired the NNC research, which is conceptually this combination: "*neural network plus* $\{0,\Delta\}^2$ *objective function*". Although the NNC provides smoother and more accurate control than the FLC, it has a much longer control cycle time. This makes the NNC less suitable for time-critical applications.

When the above dynamic buffer size tuners were verified, it was observed that traffic patterns can affect their performance and stability. In all the experiments the

FLC has remained the most efficient and stable dynamic buffer size tuner compared to other versions. For this reason the FLC is always the candidate for different testing purposes. The MPhil research, however, had left many unaddressed issues, which form the basis for this deeper PhD research of mine. The unaddressed issues are summarized in Table 4.0.1.

Tuner(s)	**Unaddressed issues in my MPhil research**
FLC	1) Is it possible to have an optimal design? 2) Is it possible to make it reconfigurable (especially with respect to traffic pattern changes)?
NNC	1) Is it possible to prune the NNC configuration on the fly so that its control cycle time can be consistently and adaptively reduced? 2) Is there a correlation between control accuracy and the number of hidden neurons in the NNC back-propagation architecture? (The procedure to provide the answer is called ***sensitivity analysis***.)
Traffic ill effects for PIDC, FLC and NNC	1) It is possible to calibrate the ill effects off-line so that the tuners can use these calibrations to ward off the impedance by fine-tuning its dynamic buffer tuning process adaptively? 2) If so, then how can the current Internet traffic pattern be deciphered on the fly (on-line) so that the off-line calibrations can be applied selectively?

Table 4.0.1 Unaddressed issues in my MPhil that forms the basis of my PhD research

Finding solutions for the unaddressed issues in Table 4.0.1 forms the backbone of my PhD research. It was difficult to rely on previous experience in solving some of the problems, in particular "*on-line*" traffic detection. Firstly, the off-line traffic analysis techniques are generally not well-established [Molnar1999] even though there are many relevant publications [Abry2000, Arvotham2001, Cao2001, Cottrel2001, Ryu1996, Crovella1997, Karagiannis2003, Leland1994, Resnick1997, Taqqu2003, Willinger2003]. Secondly, "*on-line*" traffic detection techniques were absent from the literature until the paper published by the COMP Team (or simply the Team) [ATNAC]. The team analyzed the available off-line or post-mortem statistical techniques and concluded that they are basically *lump analysis*. For example, Gaussianity test [Zhang2003] is used to determine the stationarity of a discrete stochastic process X. This is at best an estimate that can be reasonably accurate or a crude approximation because Gaussianity is continuous but the target process is discrete in contrast. A Gaussian distribution can be used to approximate a Poisson or binomial process only under certain conditions [Jain1992]. Since in most published cases the continuous and discrete ideas are lumped as one, the Team calls these cases *lump analysis*. The Team uses the Hurst parameter as the yardstick to determine if any aggregate X_l^m of block size m and lag l "H_{ss}" or not. The aggregate is H_{ss} for $0 < H < 1$, and the range $0 < H < 0.5$ indicates SRD (*short-range dependence*) traffic (e.g. Markovain) and $0.5 < H < 1$ for LRD (*long-range dependence*) traffic (e.g. heavy-tailed and self-similar). The limitation or criterion of application for the *real-time traffic pattern detector* (RTPD) [ATNAC] proposed by the COMP Team is "H_{ss} *and stationarity*" because self-similar traffic can be non-stationary (i.e. non-

linear). In non-linear situations the H value/effect does not scale linearly as a constant [Zhang2003]. In the remaining section on overview of the solutions proposed in my PhD research for the unaddressed issues in Table 4.0.1 are concisely described.

4.1 PROPOSED SOLUTIONS

4.1.1 FOR FLC

The empirical results indicate that an optimal design range exists for the FLC design. Figure 4.1.1.1 shows the optimal range. Any complex design not in this range yields no obvious advantage measured in terms of the amount of mean deviations (MD) from the given steady-state reference symbolically represented by "0" in the $\{0, \Delta\}^2$ objective function.

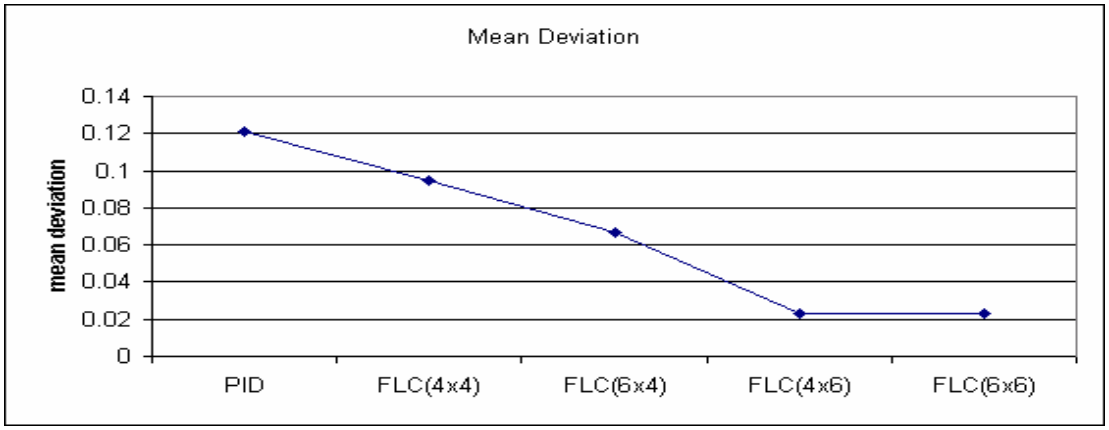


Figure 4.1.1.1 An optimal FLC design is possible (mean deviation stabilizes around 0.02) (excerpt of Figure 6.1.1.1)

It is found that it is possible to make the FLC adaptive or reconfigurable (i.e. A-FLC) [p12]. The approach is to squeeze the "*don't care*" state range threshold as shown by Figure 4.1.1.2. The amount of squeeze can be fixed/static or dynamic. The dynamic approach is suitable for neutralizing the ill effects by IAT traffic patterns on the tuner stability and efficacy on the fly.

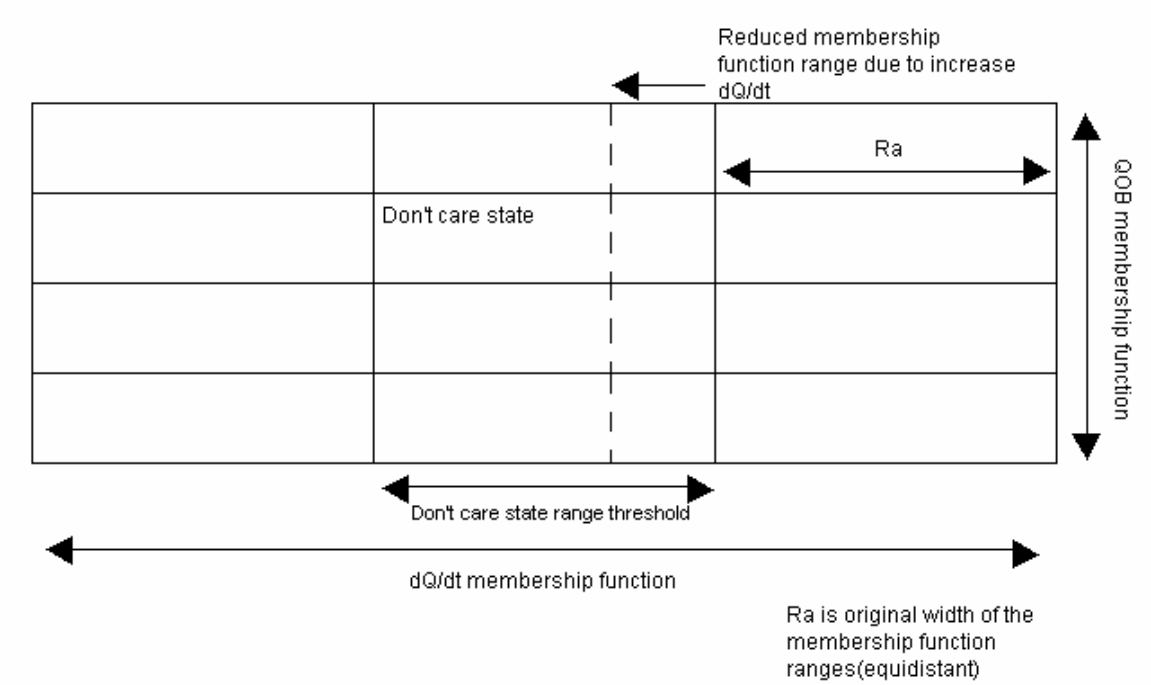


Figure 4.1.1.2 A-FLC adjustment of the *don't care* state range threshold on the fly

The calibration of the amount of squeeze versus traffic pattern (e.g. self-similar) was carried out for the FLC, as shown by Figure 4.1.1.3. Real-time application of the squeeze calibration, however, is possible only if the RTPD is included to detect the current traffic pattern on the fly. This led to the proposals of two traffic filters in my PhD research for enhancing the RTPD. The inclusion of these

two traffic filters into the RTPD framework produces the Enhanced RTPD or E-RTPD. The E-RTPD provides the basis of on-line traffic pattern detection/identification and neutralization of the traffic ill effects in the process of dynamic buffer size tuning.

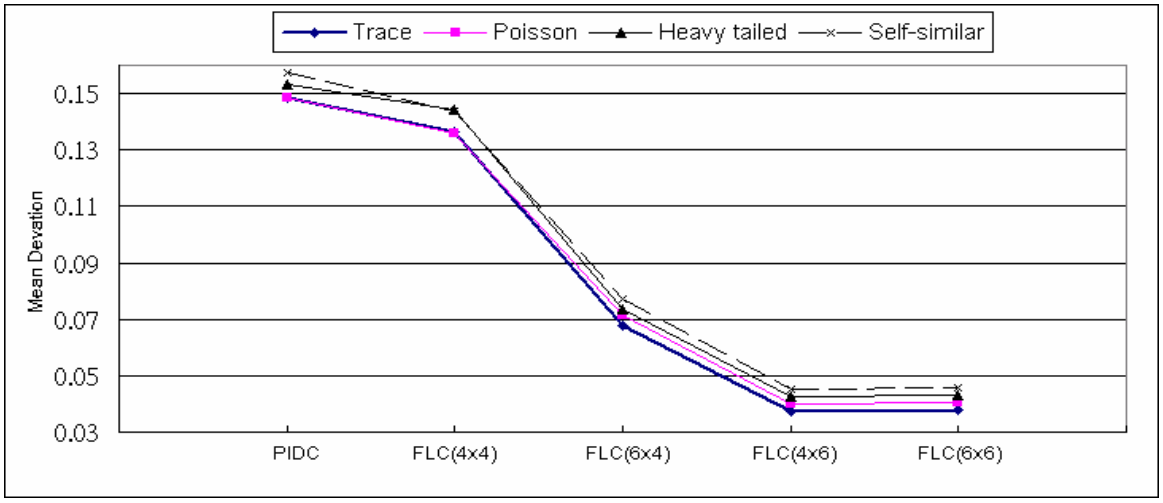


Figure 4.1.1.3 Mean Deviation Errors of different FLC designs versus traffic patterns (excerpt of Figure 6.3.1.1)

4.1.2 FOR NNC

The HBP (*Hessian Based Pruning*) approach was proposed to reduce the NNC execution time (i.e. control cycle time) on the fly. This on-line pruning/optimization technique always works with the same skeletal neural network. In operation the NNC has two modules: Chief and Learner. Figure 4.1.2.1 is the twin system of two NNC clones (*Chief* and *Learner*). The NNC operates in two distinctive phases, namely, *training/learning*, and *dynamic buffer tuning*. In action it is a *twin system* consisting of the "*Chief*" NNC module and the "*Learner*" NNC module as

shown in Figure 4.1.2.1. The *Chief* , which has already learnt previous patterns, carries out actual dynamic buffer tuning while the *Learner* undergoes training to acquire new knowledge to deal with new phenomena. Before training starts all the weights of the arcs in the *Learner's* neural network are randomized. As training progresses the error (difference) between the "*trainee*" output and the NNC desired/deserved output Δ decays gradually. After training the *Chief* and the *Learner* swap positions; the *Chief* becomes the learner.

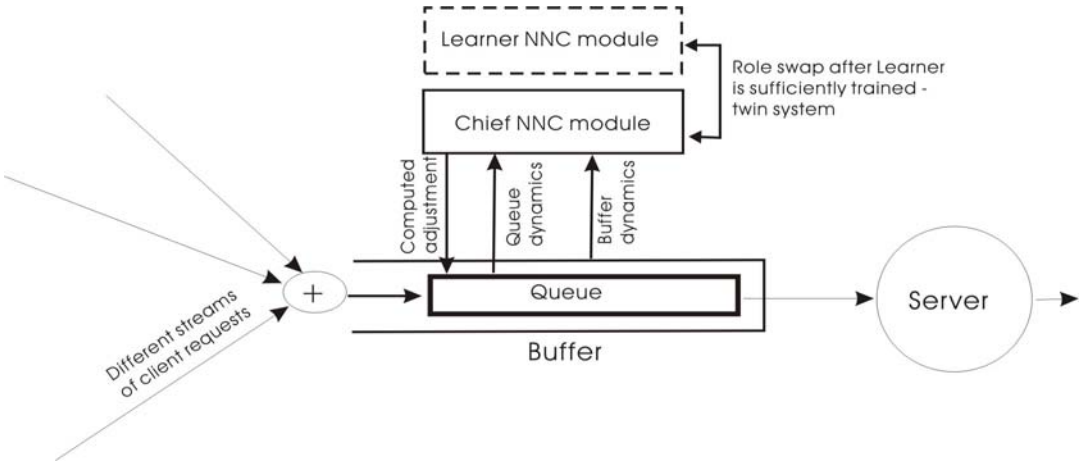


Figure 4.1.2.1 The NNC – a twin system of two NNC clones (excerpt of Figure 7.1.2)

Thorough analysis was carried out to determine if the number of hidden neurons would have an impact on the NNC performance. The preliminary empirical results shown in Figure 4.1.2.2 indicate that having 20 neurons in the NNC hidden layer is more or less the break point. Using more neurons does not produce better performance by yielding a lower MD. For the Poisson trace (a SRD pattern), the mean deviation error settles down for 15 hidden neurons in the hidden layer but for

other traffic patterns at least 20 neurons are needed. All the experimental results from this stage indicate that it is safer to use 20 neurons for the hidden layer for Internet applications because the traffic pattern, which includes all the patterns in Figure 4.1.1.2, can switch quickly without warning.

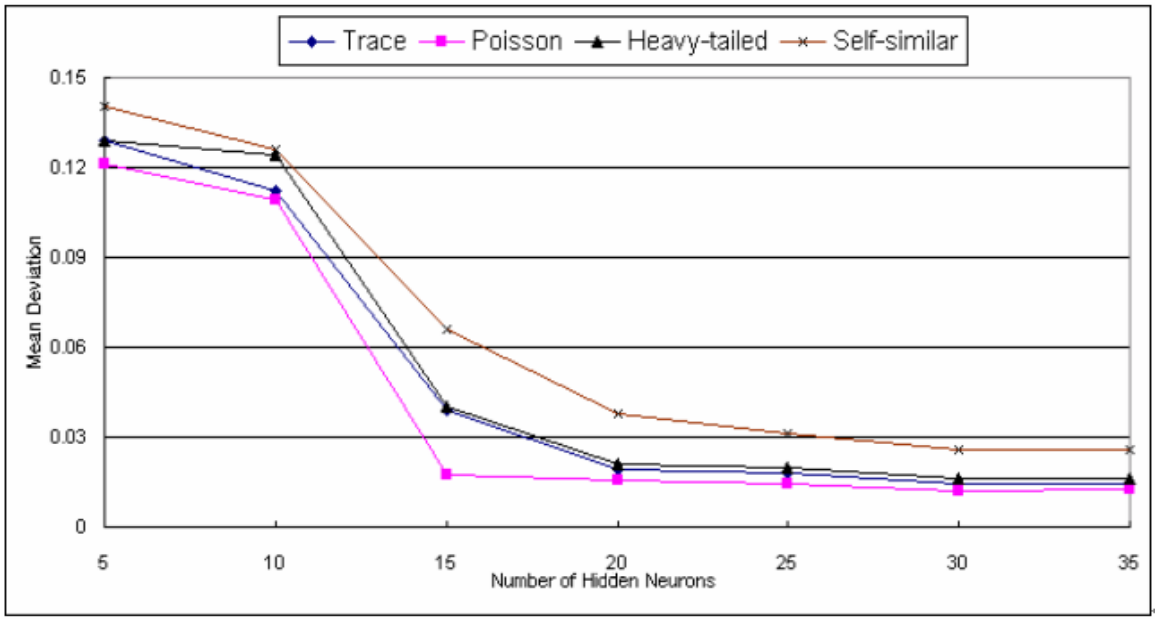


Figure 4.1.1.2 Mean deviation error for using different numbers of neurons in the NNC hidden layer versus different possible Internet traffic patterns (excerpt of Figure 7.1.1.6)

4.1.3 REAL-TIME TRAFFIC PATTERN ANALYSIS

I made use of the accumulated experience by the COMP Team in real-time traffic analysis. In return my PhD research contributed two traffic filters: *real-time modified QQ-plot* (or simply RT-QQ) filter/estimator and *self-similarity (S^2) filter*

for real time traffic pattern detection. The S^2 *filter* operation follows the CAB concept proposed by the Team. This concept helps find the starting point of a data section for meaningful RTPA evaluation. This point should satisfy the Gaussianity test, and only then the S^2 *filter* starts to find the necessary outcomes, including the H and D values for the successive aggregates X^m of a stochastic process X long the time axis. The block size m is a variable because the aggregate size for a pre-defined time interval depends on the average IAT of the aggregate; longer IAT means a smaller m. This "*timed aggregate*" approach avoids significant real-time sampling latency due to the unpredictable IAT. Figure 4.1.3.1 summarizes the CAB mechanism that the S^2 *filter* works with. The mechanism involves two separate real-time sub-operations: Gaussianity test, and traffic pattern detection. The Gaussinaity test continues throughout the CAB mechanism's service life. K1, K2 and K3 are blocks (timed aggregates of variable lengths) for three Gaussiainity tests. The second half of K1 is basically the first half of K2 to indicate that data in the current block/window is always half and half as the window is shifting forward along the time axis. For example, if Gaussianity is confirmed for K1 at $Ag1$, then the S^2 *filter* starts to collect the first timed aggregate X^m (between $Ag1$ and $Ag2$) so that the corresponding H value can be calculated. In the Figure 4.1.3.1 the S^2 *filter* tries to confirm self-similarity in X^m for the "*First aggregate*" and finds H by the P1 linear regression. The same process repeats if the K2 block is also Gaussian. If the K3 block were found to possess no Gaussianity, then the S^2 *filter* would stop operation because the data has become non-stationary. More details are presented in Chapter 5.

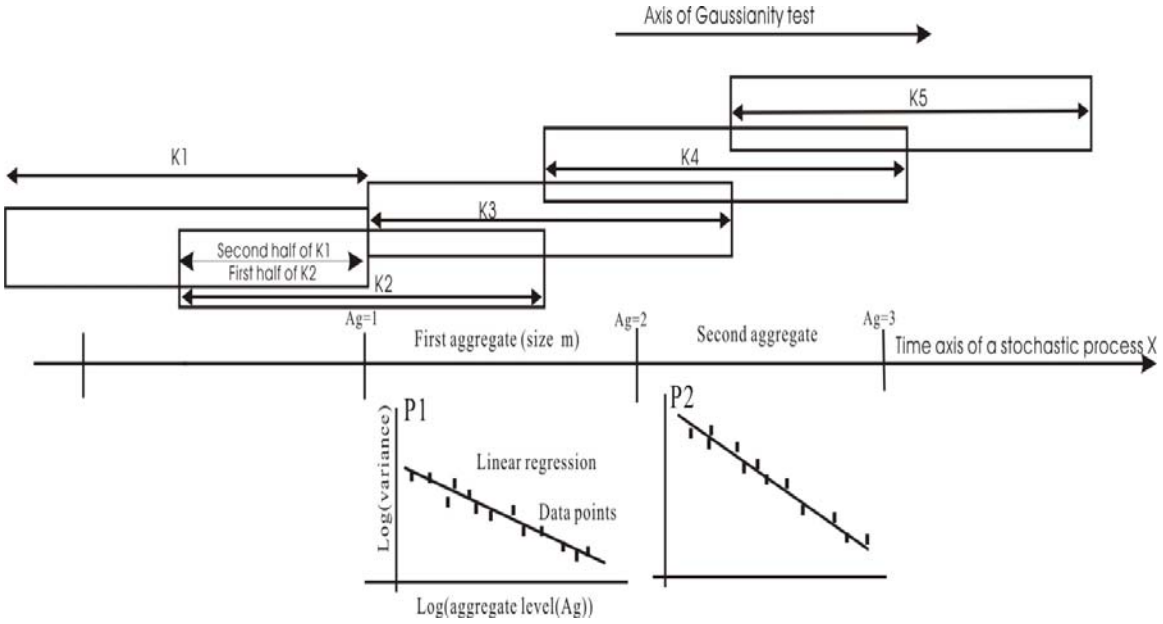


Figure 4.1.3.1 CAB mechanism has two real-time sub-operations (excerpt of Figure 5.2.3)

4.2 ORIGINALITY AND SIGNIFICANCE

This PhD research is a deeper exploration based on my previous MPhil findings [Lin2002] as the basis. In the MPhil project four original dynamic buffer size tuners for user-level applications were proposed: PIDC [Ip2001], GAC [Lin2001GAC], FLC [Lin2002FLC], and the NNC [Lin2001NNC]. In fact, these four tuners represent an evolutionary process. The PIDC, which is algorithmic, eliminates buffer overflow by proportional (P), derivative (D) and integral (I) controls despite the presence of performance shortcomings. The GAC uses genetic algorithms (GA) to eliminate these shortcomings. Unfortunately it produces occasional buffer overflow despite the fact that it has completely eliminated the PIDC shortcomings. Yet, the GA experience has confirmed that the effectiveness of the expert or soft computing

approach for dynamic buffer size tuning. The desire to preserve this effectiveness and prevent occasional buffer overflow at the same time led the proposal of the FLC, which uses fuzzy logic instead of GA. Meanwhile, the published positive experience of using neural networks in the AQM (active queue management) area [Braden1998] inspired the NNC proposal. The significant contribution by the four dynamic buffer size tuners is that they eliminate buffer overflow at the user level. As a result they shorten the client/server roundtrip time (RTT) over the Internet. These tuners are original because similar models have never been proposed before. They warrant deeper investigations for their positive impact on the performance of time-critical applications over a sizeable network such as the Internet. The findings from such deeper investigations should be original because they add new values to the original tuners.

This PhD research addresses those issues uncovered in my previous MPhil thesis, and they include the following:

a) To prevent occasional buffer overflows under GAC: The aim is to find a way to rectify the overflow problem. The preliminary conclusion is that the overflow is due to the very nature of any evolutionary techniques, which guarantee no global-optimal solution in the hyperplane [Mitchell1999]. Since this is a fundamental problem in evolutionary computing, it is outside the scope of the present research and no further pursuit was warranted.

b) To confirm that the FLC is a complete design approach: In the MPhil research only a few FLC configurations were proposed and tested. The aim is to ensure the following: i) these configurations need only short execution times, ii) they could indeed eliminate the PIDC shortcomings without causing buffer overflow and iii) the findings would pave a solid way for deeper investigation into the following:

i) Is the FLC indeed a generic design approach in the sense that any configurations would work correctly even with somewhat different performance?

ii) Is it possible to have optimal FLC design(s)?

The PhD findings confirm that the FLC is indeed a generic design approach and it is possible to have optimal FLC designs. This original contribution was not part of my MPhil findings.

c) To shorten the NNC execution time: The NNC proposed in my MPhil thesis has the longest execution time compared to PIDC, GAC and FLC and this can easily produce deleterious effects. The desire to shorten the NNC execution time led to the proposal of the Hessian based dynamic pruning technique, which successfully optimizes the neural network configuration of the skeletal NNC on the fly. This technique is original because no real-time dynamic neural network optimization by pruning as such has been proposed before. The success of using this technique to optimize the NNC continuously provides some insight into how real-time optimization of neural networks could be achieved.

d) To neutralize traffic ill effects on system performance in a dynamic manner: It was observed from the MPhil's experimental data that Internet traffic patterns can produce negative impact on a tuner's performance. Since the PIDC, GAC, FLC and NNC tuners operate in a real-time manner, a solution is needed to identify the traffic pattern at any time so that the traffic ill effects on tuner performance could be nullified. The deeper PhD investigation of this issue led to the following: i) it is possible to include real-time traffic detection capability into a tuner and ii) two novel real-time traffic pattern filters, namely the modified QQ-plot that identifies heavy-tailed traffic and the S^2 filter that detects self-similar traffic were proposed. The contribution from this area of investigation is original and significant because how real-time traffic pattern detection capability can be paired with time-critical applications for better system stability and performance is demonstrated for the first time.

The following table concisely differentiates the original and significant achievements by this PhD research from my previous MPhil thesis.

MPhil's original contribution	**PhD's original contribution**
*Four basic novel dynamic buffer size controllers/tuners were proposed, namely PIDC, GAC, FLC and NNC.	*The unaddressed issues for the four original tuners from the MPhil thesis form the problem statement of the PhD research.
*The GAC was found to be unacceptable because it yields occasional buffer overflow.	* Deeper investigation of the GAC confirms that the buffer overflow is due to the very nature of evolutionary computing. Since this is a fundamental issue in this discipline, no further work was pursued because the GAC is application of GA in nature.

* A few FLC designs were proposed (e.g. [4x4] and [4x6]) and tested with the aim to preserve the merits of the soft-computing approach as it was demonstrated by GAC and eliminate the buffer overflow at the same time.	* Different FLC designs were proposed and tested, and the empirical results confirm the following: i) the FLC is indeed a generic design approach, ii) an optimal design range exists, and iii) the FLC can be made to reconfigure on the fly for better performance (i.e. A-FLC and R2-FLC).
* The following had encouraged the NNC proposal: i) success of using the objective function for both GAC and FLC as the operation principle, ii) the desire to have a smoother convergence than the FLC, and iii) positive experience in using neural networks in the AQM area was published. * Two NNC designs, which both work as a twin parallel system: Chief (in control) and Learner (in training): a) recurrent NNC (i.e. NNC+CA), where CA is the feedback loop and b) basic NNC without feedback loop – oscillatory {The NNC+CA framework is the basis for the PhD investigation}. * The real-time nature of dynamic buffer size tuning requires short tuner execution time, and this led to the choice of backpropagation as the NNC configuration because of its simplicity. Preliminary empirical analysis indicated that configuration of 10 input neurons, 20 neurons in the hidden layer, and one output neuron could be cost-effective. The NNC still has the longest execution time compared to PIDC, GAC and FLC.	* It is desirable to shorten the NNC execution time for successful real-time applications. For this reason the original Hessian based technique that optimizes the NNC tuner by pruning its configuration on the fly was proposed. This technique is generic in nature and works correctly when incorporated into the NNC framework. The logical pruning process always starts with the same skeletal NN configuration. * Sensitivity analysis was carried out to find optimal NNC configuration(s), and the result confirmed that the [10, 20, 1] backpropagation configuration is indeed cost-effective. The analysis finds the correlation between the number of neurons in the NNC's hidden layer and the control/tuning accuracy.
* Traffic patterns can affect system performance as observed from the empirical data.	* Real-time traffic detection capability was considered and incorporated into the dynamic buffer size tuning process successfully. It is an original example of

	utilizing real-time detection to improve the performance of time-critical systems. In the process two real time traffic pattern detectors were proposed and verified: *real-time modified QQ-plot* (or simply RT-QQ) and *self-similarity (S^2) filter.* * The RTPD capability was incorporated successfully into the FLC and the NNC frameworks.

Table 4.2.1 Concise comparison of MPhil's and PhD's originality and contribution

To summarize, the original PhD contributions are the following findings:

a) The overflow under GAC is due to the very nature of genetic algorithms not to guarantee the global optimal solution in the solution hyperplane.

b) The FLC design approach is indeed generic.

c) Sensitivity analysis indicated that the [10, 20, 1] NNC configuration is indeed cost-effective.

d) The generic HBP technique can optimize the NNC execution time and makes it more suitable for time-critical application.

e) Real-time traffic detect capability is a powerful mechanism that neutralizes the ill effects by traffic patterns on system stability and performance.

f) The traffic filters, namely, RT-QQ plot and S^2 are effective for time-critical applications, independent of the composite nature of a traffic trace.

4.3 CONNECTIVE SUMMARY

This chapter has given a concise summary of all the solutions proposed in my PhD research with respect to the unaddressed issues from my MPhil as listed in Table 4.0.1. The details of these solutions and my PhD research contributions will be presented in details in the following chapters: a) Chapter 5 describes the real-time traffic detection contribution, b) Chapter 6 is the in-depth FLC research, c) Chapter 7 is the in-depth NNC research, and Chapter 8 is the location-aware test-bed with the FLC as the chosen dynamic buffer size tuner.

CHAPTER 5 REAL-TIME TRAFFIC DETECTION CONTRIBUTION

5.0 INTRODUCTION

The PIDC, GAC, FLC, and NNC dynamic tuner models proposed in my previous MPhil research [Lin2002] were verified with pre-collected Internet IAT traces (inter-arrival times among the requests from client to server). They provide the solid basis for my present deeper PhD research. The verification exercises of these tuners for my MPhil thesis, however, showed that they might produce various mean deviations (MD) from the given steady-state references. This inspired the investigation of the correlation between IAT traffic patterns and tuner stability in my present PhD study, using the FLC dynamic buffer size tuner as the test-bed. The setup for the experiments in my PhD investigations is shown in Figure (5.0.1). It has evolved over time to meet the changing experimental objectives. The dotted lines show the new additions to the basic setup shown in solid lines.

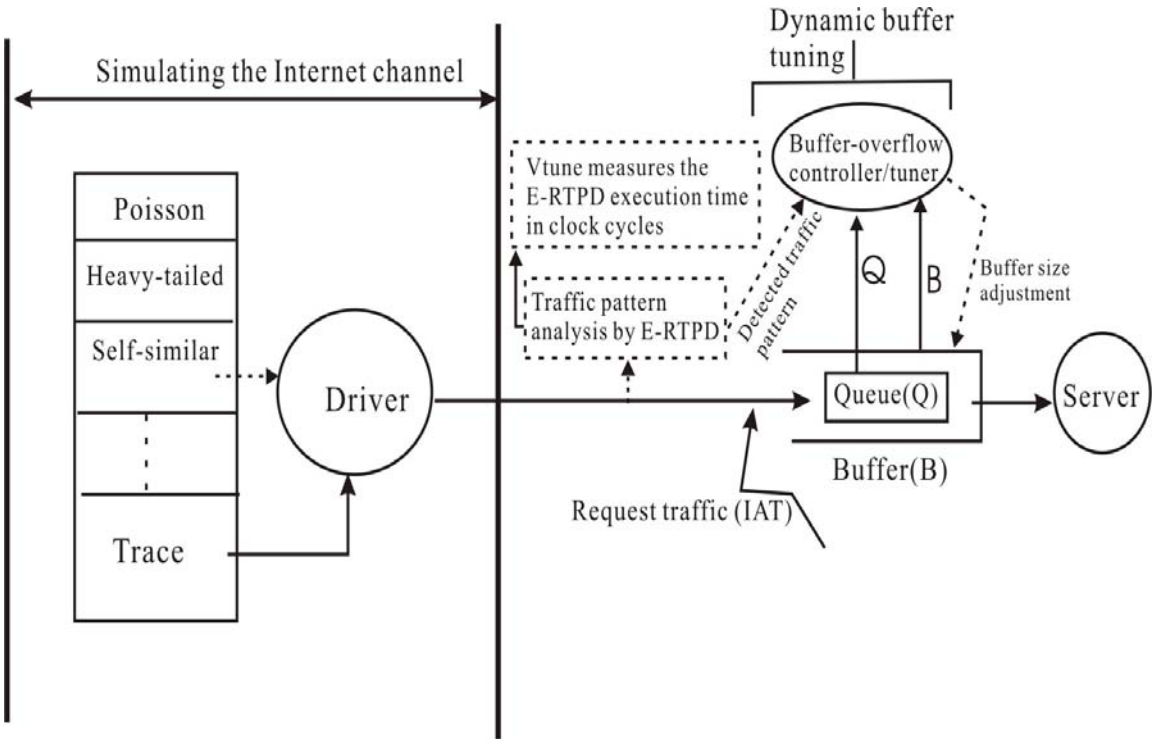


Figure 5.0.1 The setup for the subsequent tests

The setup for conducting experiments in my PhD research (Figure 5.0.1) is an Aglets mobile agent platform environment. The aim is to make the experimental results scalable because the platform is designed for the internet. The driver and the server in the setup are aglets (agile applets). The driver picks a waveform to simulate the desired IAT distribution/pattern for the requested traffic into the server's queue. The tuner (e.g. FLC) utilizes the buffer length (B) and the queue length (Q) to adaptively compute the buffer adjustment size for the dynamic tuning process. Figure 5.0.2 shows the different MD values produced by the FLC for different IAT traffic patterns. Besides detecting traffic patterns on the fly, the E-RTPD (Enhanced Real-Time Traffic Pattern Detector) also helps visualize the correlation between a traffic pattern and the corresponding MD value. Figure 5.0.2 is a result of the research work described in sections 6.2 and 6.3 of Chapter 6.

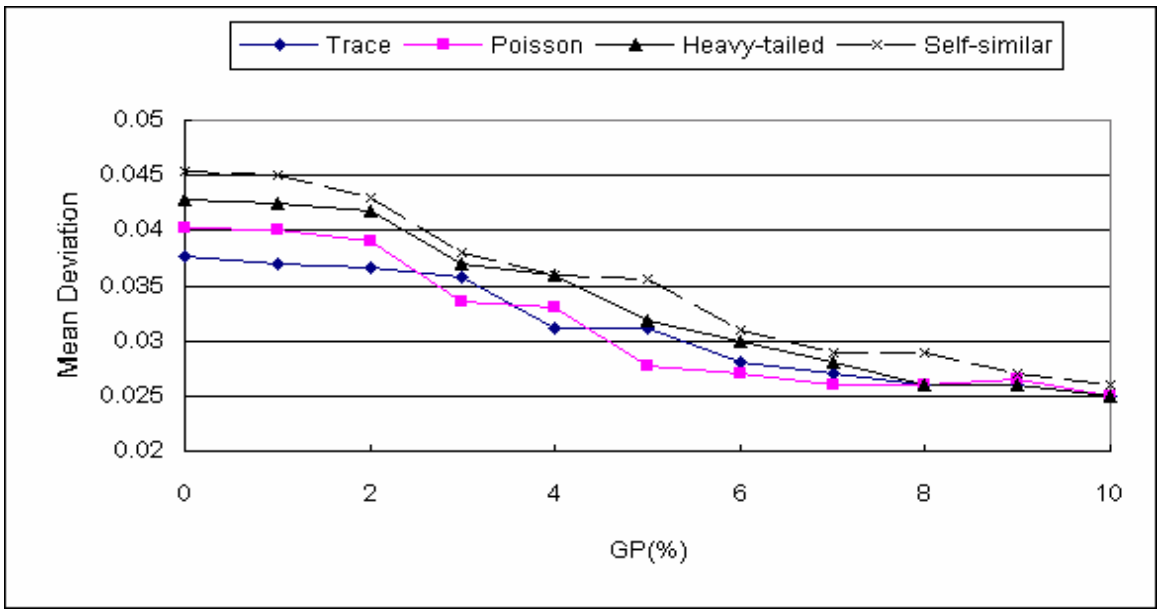


Figure 5.0.2 Different MD for specific traffic patterns by the FLC (Chapter 6)

Requirements for real-time traffic analysis differ from that for non-real-time or "post-mortem" purposes. Real-time analysis recognizes a specific pattern embedded in the data segment sampled on the fly. If an IAT collection of size m is made from a stochastic process X, then the data segment is the aggregate X^m. Over time X may yield many aggregates, which are uniquely identified by the aggregate level l [Taqqu2003], i.e. X_l^m. In my present research we call any entity that recognizes a specific pattern (e.g. self-similar) is a *traffic pattern filter*. For example, the statistical *modified QQ-plot* is a "post-mortem" filter to recognize heavy-tailed distributions [Molnar1999].

At the time of my PhD traffic investigation the research team (called the "COMP Team" hereafter) led by Dr. Allan Wong (my PhD supervisor) was deep into real-time traffic pattern detection and analysis already. The COMP Team proposed,

verified and published the novel *Real-Time Traffic Pattern Detector* (RTPD) [p11]. I was involved with the RTPD verification and that experience proved useful for traffic investigation. The RTPD design was gained from experience with the post-mortem, statistical Selfis tool [Karagiannis2003]. My participation and experience in the RTPD experiments has inspired my pursuit into proposing effective real-time traffic pattern filters. As a result I have achieved the following for my PhD thesis:

a) Converting the post-mortem *modified QQ-plot* for real-time applications.

b) Developing the novel *self-similarity* (S^2) *filter* because the original RTPD by the COMP Team does not detect self-similar traffic.

The RTPD uses the Hurst parameter/effect as the yard stick and calls a stochastic process H_{ss} if its H value is within the $0 < H < 1$ range. The range $0 < H < 0.5$ is for the *short-range dependence* (SRD) and $0.5 < H < 1$ indicates *long-range dependence* (LRD). The value $H = 0.5$ indicates "*white noise*" and is ignored. SRD includes Markovian traffic and LRD includes heavy-tailed[Resnick1997] and self-similar patterns[Leland1994, Crovella1997, Tsybakov1998]. The RTPD puts emphasis on stationary traffic. A stationary stochastic process has independent increments in its X_l^m aggregates [Willinger2003]. For example, the distribution of the arrivals between time t and $t+s$ depends solely on the interval s but not the starting point t. In the literature stationary processes are frequently associated with "Gaussinianity". A Gaussian, H_{ss}, stationary process is called the *fractional Brownian motion* and the independent increment is the *fractional Gaussian noise*. A H_{ss} process may be SRD or LRD, and a LRD process can be heavy-tailed and self-similar. The reverse may not

be true, for example, a self-similar process may not be stationary [Cao2001]. There is, however, a strong correspondence between self-similarity and stationarity.

The core of the original RTPD proposed by the COMP Team is the traditional R/S (*rescaled adjusted statistics*) approach for *non-real-time* applications. It is the statistical expression: $R/S = \frac{\max\{W_i : i = 1,2,....,k\} - \min\{W_i : i = 1,2,...,k\}}{\sqrt{\text{var}(X)}}$. The W_i parameter is defined by $W_i = \sum_{m=1}^{i}(X_m - \overline{X})$ for $i = 1,2,...k$, and $\overline{X}$ is the mean of $\overline{X} = 1/k \sum_{i=1}^{k} X_i$. Yet, the best value for *k* in this traditional approach has to be found by trial and error. This is the main drawback of the R/S approach because its speed and accuracy depend on *k*. The R/S ratio is the rescaled range of the discrete process X , $\{X_i : i = 1,2,...k\}$. The log-log plot of the $R/S = (k/2)^H$ feature yields the H value. The time to compute $\overline{X}$ is unpredictable because of *k*. The COMP Team resolved this unpredictability and converted the R/S into the *enhanced R/S* version (i.e. E-R/S) for real-time applications by incorporating the *Convergence Algorithm* (CA) [Wong2001]. This involves transferring and adapting CA, which is from the IEPM (*Internet End-to-End Performance Measurement*) domain [Cottrel1999], for effective application in real-time traffic analysis.

The CA operation is based on the *Central Limit Theorem*, and its accuracy is therefore independent of the traffic waveform. It is summarized by the equations: (5.1) and (5.2).

$$M_i = \frac{M_{i-1} + \sum_{j=1}^{j=F-1} m_j^i}{F} \text{..... (5.1)}; M_0 = m_{j=0}^{i=1} \text{..... .(5.2)}; i \geq 1$$

The estimated mean M_i in the i^{th} prediction cycle is based on the fixed F (*flush limit*) number of data samples. The cycle time is the interval for collecting the F samples physically. It was previously confirmed that M_i has the fastest convergence for F=14 [Wong2001]. Other parameters include: a) M_{i-1} is the feedback of the last predicted mean to the current M_i prediction cycle, b) m_j^i is the j^{th} data item sampled in the current i^{th} M_i cycle, $j = 1,2,3,....,(F-1)$, and c) M_0 is the first data sample when the MCA had first started running. M_i replaces $\overline{X}$ to yield $W_i = \sum_{m=1}^{i} (X_m - M_i)$ for the E-R/S, which is more suitable for real-time applications because the number of data items (e.g. IAT) needed to calculate W_i is fixed (predictable), namely $F = 14$.

To summarize, my PhD contributions to real-time traffic analysis are threefold:

a) Development of two novel real-time traffic pattern filters: *RR-QQ* (*real-time modified QQ-plot*) and *self-similarity* (S^2)

b) Conversion of the RTPD to its enhanced version (i.e. Enhanced RTPD or E-RTPD) by including *RR-QQ* and S^2 filters.

c) Addition of these two novel filters to enable the FLC to reconfigure (i.e. the Reconfigurable FLC (R-FLC) in section 6.2 of Chapter 6) by using the results detected by the E-RTPD on the fly. The reconfiguration adjusts the FLC's derivative (D) control to neutralize the ill effects arising from changing traffic patterns. As a result more accurate dynamic buffer size tuning can be carried out and maintained.

5.1 TRAFFIC ANALYSIS IN GENERAL

Three goals for traffic analysis can be identified from the literature: a) gauging the end-to-end channel traffic to interpret the channel behavior, b) trace-based, post-mortem or off-line traffic analysis (OTA) to understand the network behavior in the period where the trace was collected, and c) real-time traffic pattern analysis (RTPA) so that the result can be used immediately by a running application to self-tune or reconfigure to maintain high performance. Trace-based, off-line analyses pertain only to the traces concerned because the empirical results cannot be construed as the general network behavior. For example, the different traces may exhibit similar behavior because when the traces were collected the same network parameters happened to be coincidentally dominant.

5.1.1 GAUGING END-TO-END BEHAVIOR

It is always desirable to gauge the end-to-end client/server path (EE-path) (Figure 5.1.1.1) or a TCP channel for more reliable and efficient communication purposes. This approach is the basis of the IEPM (Internet End-to-End Performance Measurement) school of thought [Cottrel1999]. The off-line tools that manipulate pre-collected traces include the Skitter, PingER, M^2RT [Wong2001M^{2}RT] and SURVEYOR [Cottrel2001]. The M^3RT (*Micro Mean Message Response Time*) tool is the only known IEPM model capable of on-line applications. It can predict the

mean of an IAT aggregate X^m accurately and quickly on the fly. The difference between channel traffic and EE-path traffic is subtle and therefore they are used interchangeably. Precisely, channel traffic means those that have reached the exit of a logical channel. A server on the user-level is normally an *asymmetric rendezvous* (i.e. one-server-to-many-client relationship). In Figure 5.1.1.2 different service request streams from different channels merge at the server's *service access point* (SAP) before entering its queue of service requests. Therefore, a channel traffic pattern could be very different from the composite "merged traffic" or "EE-path" behavior.

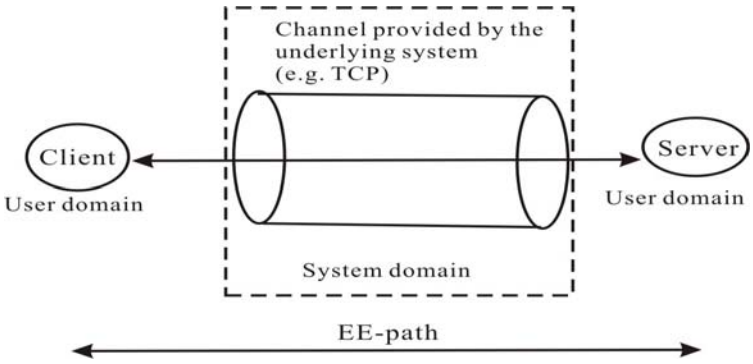


Figure 5.1.1.1 The EE-path

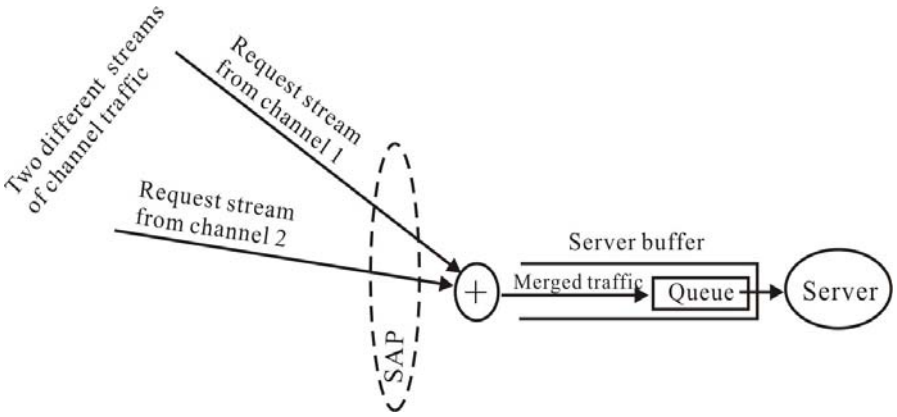


Figure 5.1.1.2 Merged traffic at the user-level

5.1.2 OFF-LINE (POST-MORTEM) TRAFFIC ANALYSIS

The aim is to understand the network/channel behavior with respect to the trace being examined. The *off-line traffic analyzing* (OTA) techniques are not well-established at this moment [Molnar1999, Taqqu2003]. From the COMP Team's point of view, the results from using these techniques should be accepted only from the trace perspective. They cannot be generalized to represent the underlying network. The OTA techniques are basically statistical [Karagiannis2003] and aim at determining the following properties:

a) *Stationarity*: A stochastic process X is stationary if its aggregates X^m of block size m have independent increments. Conceptually stationarity is an expression of the Gaussian property (i.e. "*Gaussianity*"). It is generally accepted that the Gaussian (normal), Poisson, Erlang, and binomial distributions belong to the *exponential family*, which is memoryless [Mitrani1987] and is therefore stationary. Consequently any bell curve that fits an Erlang variant of a specified shape parameter is exponential. This provides the basis for the "*kurtosis/skewness*" test that can verify Gaussianity [Jain1991]. The kurtosis value determines if a bell curve is *peaked* (for positive values) or *flat* (for negative values). The skewness value decides if the bell curve skews to the right (for positive values) or to the left (for negative values). For example, skewing to the right means the right tail in the distribution is heavier than the left. The pivotal point is that Gaussianity of a normal distribution is perfect for $kurtosis = 3$ and $skewness = 0$, which are the "*standards or references*" for

comparison. It is reasonable to decide if a bell curve is Gaussian by comparing with these "standards" in a test for Gaussianity. Statistically estimated kurtosis and skewness values from a trace are rarely perfect. Reasonably predefined kurtosis and skewness limits, however, help determine if a bell curve of the exponential property does exist. For example, if the following are computed: $kurtosis = 1.5$ and $skewness = 0.5$, statistically the bell curve is somewhere between a Weibull distribution ($gamma = 1.5$, $skewness \approx 4.5$ and $skewness \approx 1$) and a normal distribution. Therefore it may be regarded as part of the exponential family to possess stationarity. The COMP Team regards the Gaussianity test as a crude but workable way to look for the sign of existence of an exponential bell curve. Choosing the appropriate kurtosis and skewness limits, however, depends on empirical experience and is therefore an art rather than a science. Guessing the nature of the Gaussianity test by kurtosis and skewness becomes obvious if the properties of relevant distributions are examined. A normal distribution is inherently continuous but Poisson and binomial processes (such as packet traffic flow over the Internet) are discrete. *The normal, binomial and Poisson processes can possess approximately the same behavior only under certain constraints* (to be explained later). Therefore, there is ample room for making wrong guesses in using only the kurtosis/skewness test for Gaussianity.

b) *Hurst (H) effect*: H effect/value measurement originated from hydrology (water flow), and only much later was adopted by researchers for traffic analysis [Molnar1999]. Statistical methods to estimate H include the R/S (*rescaled adjusted*

statistics) method and the Periodogram [Molnar1999, Karagiannis2003]. The H value of a H_{ss} stochastic process is divided into three sections: $0 < H < 0.5$ for SRD (short-range dependence; e.g. Markovian inter-arrival times (IAT) traffic), $H = 0.5$ for "*white noise*" and $0.5 < H < 1$ for LRD (long-range dependence). The relatively more complex LRD has two basic components: heavy-tailed and self-similar. Self-similar patterns often result from heavy-tailed traffic but the latter is <u>*not*</u> a necessary condition for self-similarity [Ryu1996]. For example, the self-similar FSNDPP (*Fractal-Shot-Noise-Driven Poisson Process*) has no heavy-tailed property. Fractal and self-similar are synonymous except that the fractal dimension (D) is non-integer (i.e. real number). Objects are self-similar or fractal if they can be derived from others by scaling, rotation, and translation. The different existing definitions for the fractal dimension are non-converging. The *Cantor Set*, however, provides a reasonable conceptual basis. If an object is geometrically, recursively split into similar pieces, then at the K^{th} iteration step the total measure of the object is the "*product of the number of similar pieces and* O^D *"*. The parameter *O* is the splitting *resolution* or *reduction*. For example, the *Cantor Set* considers drawing a line segment of interval [0,1] as the first step (i.e. $K = 0$). This line is then manipulated by the subsequent steps: a) divide the line into three equal portions (i.e. *resolution* is $1/3$) and remove the middle portion (i.e. $K = 1$), b) remove the middle portions from the remaining two (i.e. $K = 2$), and c) repeat the last step *ad infinitum*. The K^{th} iteration produces 2^K similar line segments of length $s = (1/3)^K$. The *Cantor Set's* self-similarity dimension is defined by the formula $D_s = 2^K * (1/3)^K$ or alternatively

$D_s = [{(K \log(2))}/{(K \log(3))}] \approx 0.63$. In fact, the extant FD3 tool [Sarraille] can determine if an object or image is fractal and measures its D value.

c) *Linearity*: Self-similar traffic can be linear and non-linear. Linear fractal traffic scales with a specific H value, but for the non-linear cases H becomes a variable. The following two methods can test and confirm linearity effectively: a) the "*wavelet partitioning function* (WPF) [Abry2000]" approach and b) the "*CAB-based D/H plot*" proposed by the COMP Team (explained later).

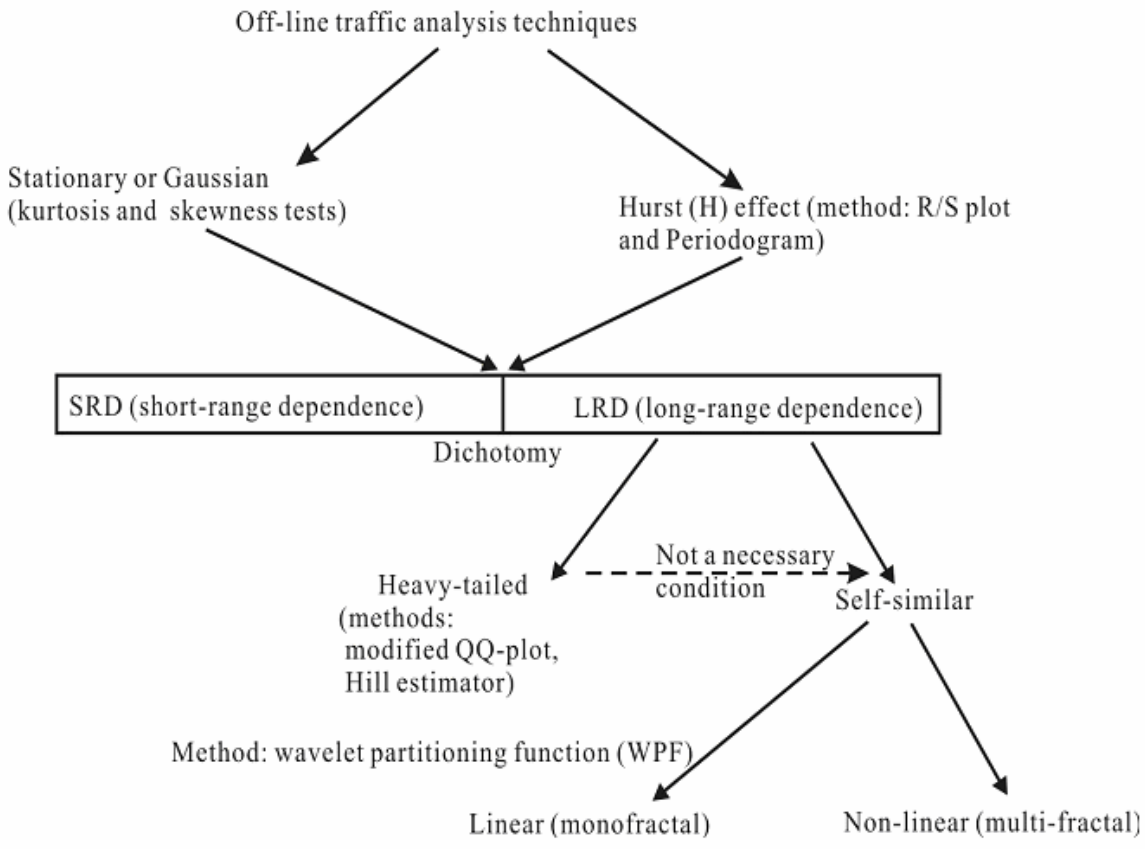


Figure 5.1.2.1 The hierarchy of OTA methods

All the OTA discussions above are summarized in Figure 5.1.2.1. As a demonstration this hierarchy is walked through in order to find the self-similarity dimension D for an IAT aggregate X^m. The steps involved are as follows: a) determine the Guassianity of X^m by computing its kurtosis and skewness values and comparing them with the chosen limits, b) if Gaussianity is positive then compute H by the R/S or Periodogram methods, c) for $0.5 < H < 1$ determine D by using the FD3 tool [Sarraille], and d) use the WPF approach to confirm that D is correct due to the existence of linearity.

5.1.3 REAL-TIME TRAFFIC PATTERN ANALYSIS (RTPA)

Post-mortem traffic analysis is useful to understand what happened in a network, but only in the trace perspective. It is, however, impractical to engineer a system and expect it to work correctly in a time-variant environment, based on partial past performance data. The *real-time traffic pattern detector* (RTPD) published by the COMP Team is an example of the RTPA approach [Lin2004a]. If the RTPD is incorporated as a component in a time-critical application, the latter can use the detected traffic pattern to self-tune for more stable performance in a dynamic and adaptive fashion.

5.2 THE COMP TEAM

Real-time traffic pattern detection proposed by the COMP Team is a novel concept. Before that the known methods are basically post-mortem. The relevant experience accumulated by COMP Team that is useful for my PhD study includes the following, namely:

a) Lump analysis

b) Essence of time

c) Traffic independence

d) Micro implementation

e) D/H correlation

a) *Lump analysis*: The COMP Team considers the OTA techniques as lump analysis. The reason behind this is that the raw trace used in an OTA exercise can be composite. It has no demarcation where one traffic pattern begins and ends before the next. Therefore, the overall result indicates the composite effect of the different traffic types interleaved together. This is misleading in terms of system behavior. For real-time applications the response to stimulation, however, is immediate and clear. For example, a dynamic buffer tuner must respond adaptively to the continuous IAT traffic pattern changes, which are interleaved along the time axis. To trace where a traffic pattern begins and ends means scrutinizing the true characteristic of a X^m aggregate by having the size m as a variable. To have a meaningful scrutiny the m value should represent a sufficient number of samples to make the traffic

characteristic of X^m stand out. The RTPD approach is time based and this makes m a variable. Being time based means we have to denote it as $X^m_{t=T}$, where the suffix T indicates a pre-defined interval. The $X^m_{t=T}$ aggregates are examined one after another until the analytical process has ended. This kind of successive aggregate inspections makes the RTPD approach differ from the OTA lump analysis

b) *Essence of time*: For the success of any real-time application, time is of the essence. OTA techniques normally work with immediately available data in the trace. For example, if an OTA method needs an average of 200 time units to compute the result from 1000 samples in the IAT trace (i.e. $X^{m=1000}$), then the computation/execution time is *intrinsic*. It is intrinsic because it does not include the actual sampling latency for the 1000 samples. If the average IAT for the 1000 samples is 1 second, then the actual time needed to compute the result is 1000*1+200 or 1200 time units on-line. The data items have to be sampled one by one before the computation. For immediately available data in a pre-collected trace, however, there is no such sampling latency. A long sampling latency/delay can lead to deleterious effect because by the time the traffic characteristic is identified it has become history and would have changed and is, therefore, useless for on-line application. Therefore time essence requires the RTPD mechanism to produce a result quickly so that it is can be used immediately by a real-time application to self-tune and rectify itself.

c) *Traffic independence*: The essential quality for any tool to analyze traffic and identify its characteristic(s) correctly is traffic independence. This was repeatedly

demonstrated by the previous IEPM (Internet End-to-End Performance Measurement) applications [Cottrel1999]. Any tools that are based on the Central Limit Theorem (CLT) [Aloisio1980] are inherently traffic independent [Wong2001M^{2}RT].

d) *Micro implementation*: Any successful tool for RTPA purposes should be simple so that it executes quickly to produce the result needed for real-time applications [Ip2002]. It should run independently so that: a) it can be invoked for service anytime and anywhere, and b) it does not burden/delay the execution time of its service user. For example, if the tool executes much faster than its service requestor running in parallel, then when the requestor needs the result it is immediately available (i.e. no substantial waiting).

e) D/H correlation: The fractal dimension D is proportional to the H value and the resolution (as in the *Cantor Set*). Figure 5.2.1 correlates the D and H values computed by seven experiments. The self-similar traffic traces were artificially generated for the experiments by using Kramer's tool [Kramer]. The D measurements were conducted with the FD3 tool [Sarraille], and the H values were estimated by using the Selfis tool [Karagiannis2003]. If we assumed that the traces for the seven experiments were aggregates (i.e. $X^{m}_{l=1,2,..7}$) of the same stochastic process X, then X is nonlinear. The D/H correlation shows the non-linearity of X because H changes as a variable. In fact, in Figure 5.2.1 the D value is proportional to H and the splitting *resolution* (e.g. the Cantor set's resolution of $1/3$).

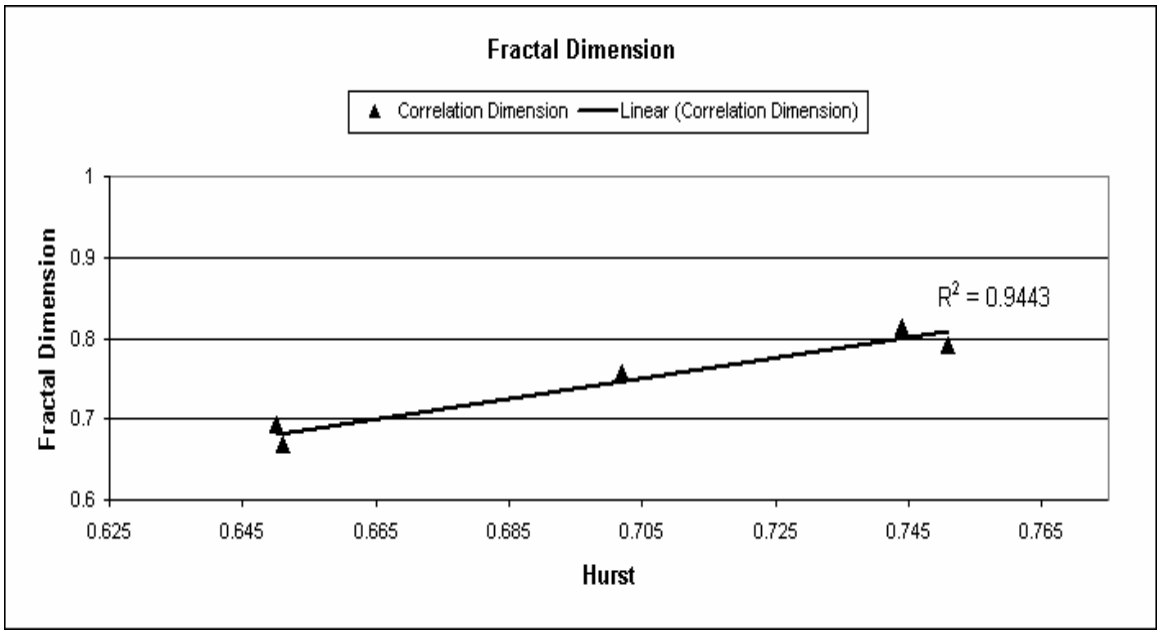


Figure 5.2.1 D/H correlation with respect to Table 5.3.2.1.1

Some of the COMP Team's conclusions such as the following directly pertain to my PhD research of real-time traffic analysis and filter design, namely:

a) M^3RT adoption

b) Conceptual discrepancy

c) CAB (continuous aggregate based)

d) CAB-based D/H plot

a) M^3RT *adoption*: M^3RT is the unique *micro* implementation of the *Convergence Algorithm* (CA) [Wong2001]. It is CLT based and needs only 250 clock cycles to execute and predict on the fly the mean of a waveform [Ip2002]. It is very useful for calculating the mean values needed by a real-time traffic filter quickly and accurately.

It has converted the traditional R/S method for measuring H to the *enhanced* R/S version (i.e. E-R/S), which is a component in the RTPD [Lin2004a].

b) *Conceptual discrepancy*: One reason for *off-line traffic analysis* or OTA techniques being not well-established is the conceptual discrepancy between continuous and discrete processes. Some OTA tools just lump the two concepts together in a high-level manner, and consequently these tools could hardly produce qualitative results. For example, many publications try to explain the association between the Hurst parameter and Gaussianity in the light of a continuous stochastic process such as hydrology (i.e. water flow) [Hurst1965]. This is probably fine for both phenomena originated from the continuous domain. Yet, when researchers directly transfer/apply their Gaussian explanations to the discrete domain such as packet traffic in the Internet, problems emerge. The transfer may be logical, but it is only "approximately or marginally correct" in the discrete domain. In fact, the problem of direct transfer as such is well known. In the area of process control [Courriou2004], digital (discrete) and analog (continuous) implementations of the same controller model produce different results. Discrete control (e.g. digital motor) works with different equations and falls into the domain of *Z Transform*, but continuous controllers work with *Laplace Transform*. Although the analog controller delivers the expected controlled system behavior, its supposedly equivalent digital version can shift the 3 db down point to create system instability. This implies that discrete control needs compensation to yield the same behavior as its analog counterpart. The COMP Team grasped this conceptual discrepancy by reviewing the

relationship among different discrete and continuous distributions. The discrepancy is summarized in Figure 5.2.2. Conceptually the Bernoulli Trials experiment (e.g. throwing a coin until a head appears) produces a memoryless binomial distribution. If σ is the probability of having a head, then the probability P_j of producing a head at the j^{th} trial is $P_j = \sigma(1-\sigma)^{j-1}$. Consider the F(j) distribution $F(j) = \sum_{j=1}^{K} P_j = \sum_{j=1}^{K} \sigma(1-\sigma)^{j-1} = 1-(1-\sigma)^K$.

The F(j) distribution obtained by summing P_j above is power, geometric and binomial. This binomial distribution can be approximated by the Poisson distribution for rare events. An event is rare if it has less than 10% chance (i.e. $\sigma = 0.1$) to occur in a sample of size $n > 50$ [Mitrani1987], $n\sigma \le 5$ is the criterion for rarity. The approximation deteriorates as σ increases and n decreases. The normal distribution, which is in the continuous domain, can approximate both the discrete, memoryless binomial and Poisson distributions. The approximation is good for the binomial distribution for $n\sigma > 25$ and for the Poisson distribution (i.e. $e^{-\lambda}e^{\lambda} = 1$) for $\lambda > 9$ (or $\lambda = n\sigma$). The conceptual discrepancy, which lies in the constraint differences shown in Figure 5.2.2, can sometimes make the Gaussianity test based on using kurtosis and skewness values unreliable. The constraints are, namely, $C1 \Rightarrow n\sigma \le 5$, $C2 \Rightarrow n\sigma > 25$, and $C3 \Rightarrow n\sigma > 9$ [Jain1991]. The Gaussianity test based on kurtosis and skewness is, however, empirically a workable but crude approach to verify the memoryless (i.e. exponential) property of X^m as a necessary condition for stationarity. The success, however, depends on choosing correct kurtosis and

skewness limits for comparison with other distributions in the exponential family, for example, the Weibull and normal distributions.

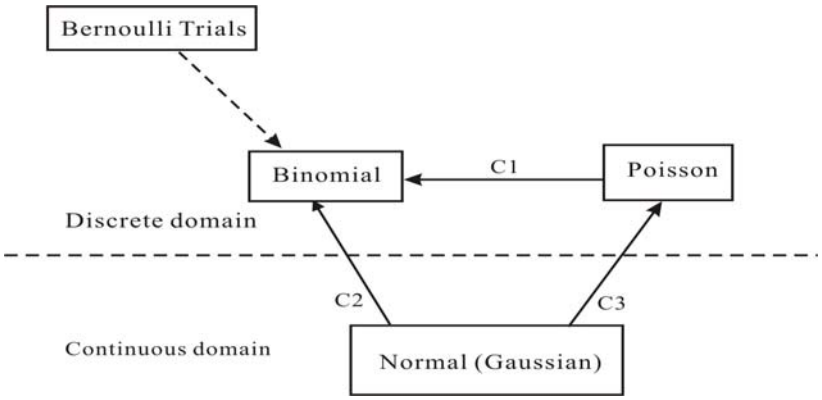


Figure 5.2.2 Relationship among some common distributions

c) CAB (*continuous aggregate based*) approach: This is for finding the beginning of a data section for meaningful RTPA evaluation. This starting point should satisfy the Gaussianity test. Only then, could the RTPA mechanism (e.g. the RTPD) start to find the necessary outcomes, for example, the H and D values for the successive aggregates X^m of a stochastic process X along the time axis. The block size m is a variable because the aggregate size for a pre-defined time interval depends on the average IAT of the aggregate; a longer IAT means a smaller m . This "*timed aggregate*" approach avoids significant real-time sampling latency due to unpredictable IAT. Figure 5.2.3 summarizes the CAB mechanism, which has two separate real-time sub-operations: the Gaussianity test, and RTPD. The Gaussinaity test continues throughout the CAB mechanism's service life. K1, K2 and K3 are blocks (timed aggregates of variable lengths) for Gaussiainity tests. The second half of K1 is basically the first half of K2 to indicate that data in the current block/window

is always half and half as the window is shifting forward. For example, if Gaussianity is confirmed for K1 at $Ag1$, then the RTPD mechanism starts to collect the first timed aggregate X^m (between $Ag1$ and $Ag2$) so that the corresponding H value can be calculated. In the Figure 5.2.3 example, the novel self-similar (S^2) *filter*, which is my PhD contribution, tries to confirm self-similarity in X^m for the "*First aggregate*" and finds H by the P1 linear regression (explained in detail later). This process repeats if the K2 block is also Gaussian. If the K3 block was found to possess no Gaussianity, then the S^2 *filter* stops operation because the data has become non-stationary.

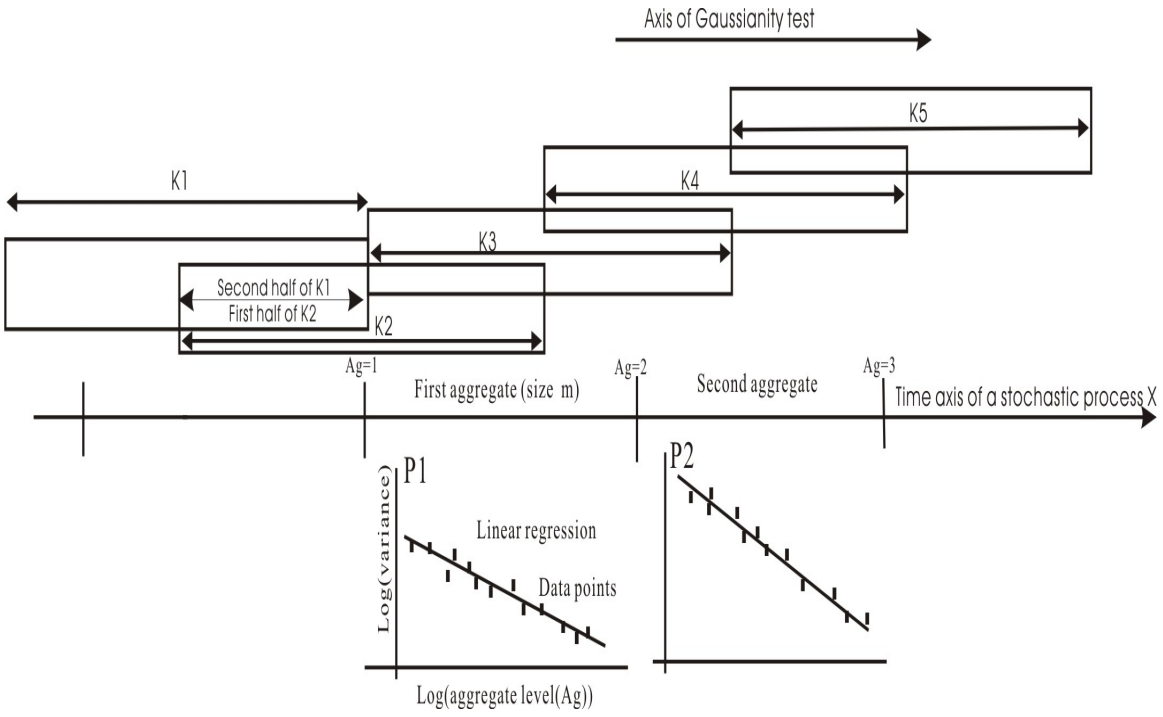


Figure 5.2.3 The CAB mechanism has two real-time sub-operations

d) *CAB-based D/H plot*: This is an extension of the S^2 *filter* investigation and theoretically the correlation between D and H can be found for every aggregate in a

real-time manner. If the H values of all the aggregates along the time axis of stochastic process X are approximately the same, then X is monofractal; otherwise it is multifractal. At this moment the *CAB-based D/H plotter* makes use of the extant FD3 tool [Sarraille] to compute D for every timed X^m aggregate. The plotter is being refined and the focus is on how to construct a real-time mechanism that is functionally similar to FD3 but needs less time to execute.

5.3 THE RTPD CONTRIBUTION

5.3.1 REAL-TIME MODIFIED QQ-PLOT FILTER

A distribution *F* is LRD and heavy-tailed [Resnick1997] if

$1 - F(x) = x^{-\alpha} L(x)$ holds.

L is slowly varying, if

$Lim_{x\to\infty} L(tx) \big/ L(x) = 1$ for $t > 0$.

The simplest case of heavy-tailed distribution is the Pareto in the form of $F(x) = 1 - x^{-\alpha}$. The preliminary experimental results with different heavy-tailed traces show that the E-R/S always recognizes their LRD character. The rationale of the *modified QQ-plot*[Kratz] consists of the following: a) pick *k* upper order statistics from the samples $\{X_1, X_2, ..., X_n\}$, namely, $X_1^* \geq X_2^* \geq ... \geq X_k^* = u$ and discard the rest, b) plot $\{(\log(\frac{X_j^*}{u}), -\log(\frac{j}{k+1}))$, $1 \leq j \leq k\}$, and c) best-fit the data points to

estimate α. Physically the $X_1^* \geq X_2^* \geq ... \geq X_k^* = u$ set consists of the following: a) X_1^* represents the event that has the highest frequency of occurrence in the set, b) the set is arbitrarily chosen from a much larger set of ranked events by their frequencies of occurrences, and c) u is the value of the lowest ranked event in the set, namely, X_k^*. The *coefficient of determination* R^2 characterizes the regression (fitting) quality, the higher the better. The *modified QQ-plot* is one of the many tools that can identify the heavy-tailed character. In my PhD research I have converted this popular post-mortem statistical technique for real-time applications, namely, the RT-QQ. The conversion in the form of a Java object is actually a traffic filter to be invoked by the filtration process, which is part of my RTPA (real-time traffic pattern analysis) contribution.

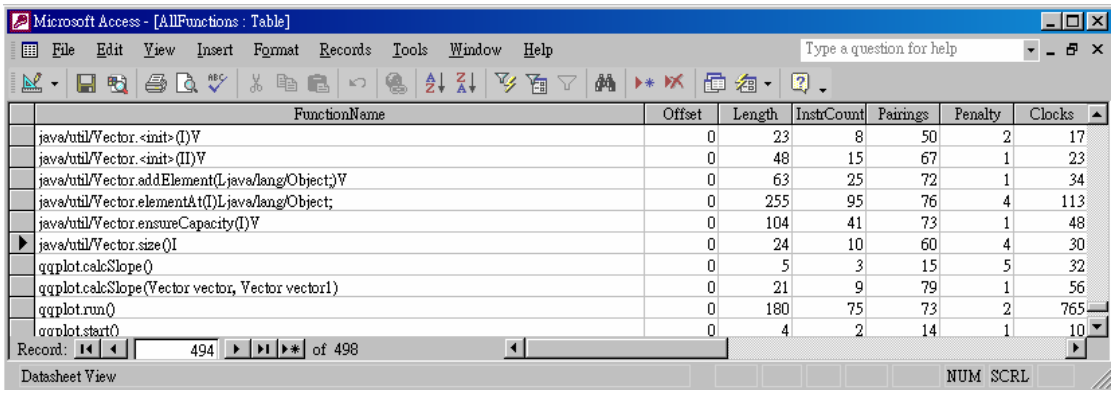


Figure 5.3.1.1 Timing Analysis of the QQ Estimator (765 clock cycles) by the Intel VTune Timing Analyzer

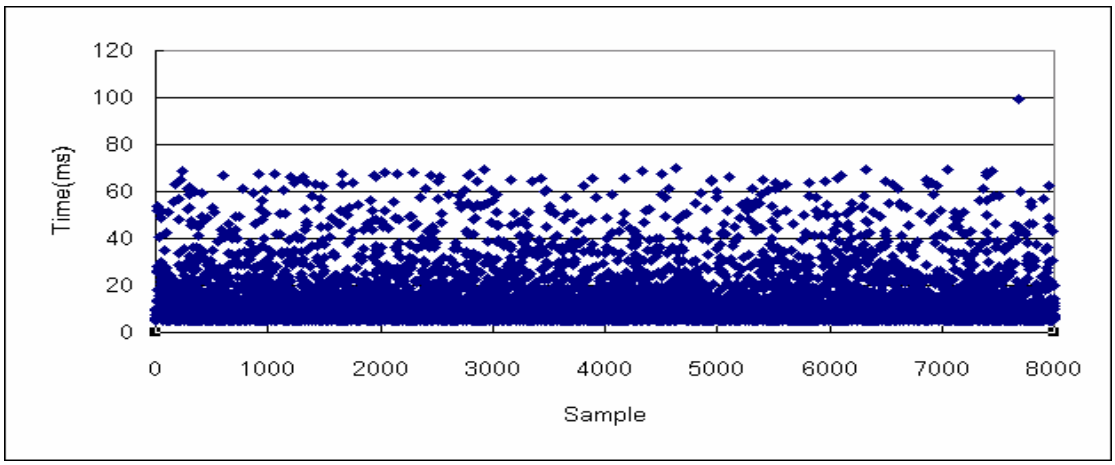


Figure 5.3.1.2 A heavy-tailed traffic trace

Timing analysis of the of the real-time modified QQ-plot Java-based or RT-QQ filter by the VTune [VTune2002] show that it needs an average of 750 clock cycles to execute. If the filter is running on a platform that operates at 100Mhz, the physical time is $PT = {750}/{(100*10^6)} \approx 7.5$ micro seconds only. This is the operational limit of the filter because what it identifies is meaningless if the IAT of the waveform is shorter than 7.5 micro seconds. The physical limit, indicates that the RT-QQ filter can cater for a wide spectrum of time-critical applications. Figure 5.3.1.1 shows the VTune analytical result of 765 clock cycles that the filter needed to identify the heavy-tailed IAT traffic pattern shown in Figure 5.3.1.2. Like other traffic filters the *RT-QQ* runs independently as a logical entity to provide service anytime and anywhere even though it is structurally an E-RTPD component. The filter indicates that the traffic pattern in Figure 5.3.1.2 as heavy-tailed because strong likelihood is confirmed by the high *coefficient of determination* for the $R^2 = 0.9231$ regression. The meaning of strong likelihood is user-defined. The criterion is that the computed R^2 value should be greater than the chosen threshold Th_{R^2}; $R^2 > Th_{R^2}$. In

light of the heavy-tailed property defined by the $1 - F(x) = x^{-\alpha} L(x)$ expression for a LRD trace, α is equal to 0.5989 in this case. Although the Java RT-QQ filter prototype recognizes "*heavy-tailedness*" by the quality of the linear regression, namely, R^2 in on-line applications, it also provides the function to produce plots like the post-mortem approach. The plot in Figure 5.3.1.3 is produced by the RT-QQ filter for demonstration purposes.

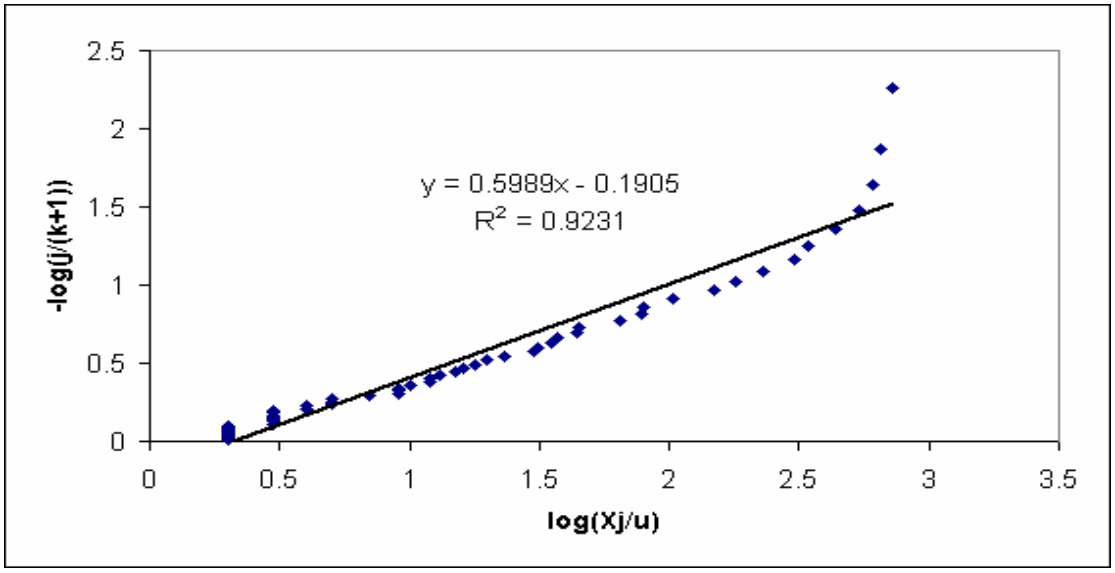


Figure 5.3.1.3 Modified QQ-plot filter identifies heavy-tailed character for the trace in Figure 5.3.1.2

5.3.2 SELF-SIMILARITY (S^2) FILTER

LRD traffic has two basic components: heavy-tailed and self-similar. The proposed *self-similarity* (S^2) *filter* differentiates heavy-tailed IAT patterns from self-similar ones. Self-similarity in many fractal point processes results from heavy-tailed

distributions, for example, FRP (*Fractal Renewal Process*) inter-arrival times. The heavy-tailed property, however, is not a necessary condition for self-similarity because at least the FSNDPP (*Fractal-Shot-Noise-Driven Poisson Process*) does not have the heavy-tailed property. The S^2 *filter* basis is the "*asymptotically second-order self-similarity*" concept, or simply called *statistical* 2^{nd} *OSS* or *S2OSS*, which is associated with a sufficiently large *aggregate level* or *lag* l in a discrete stochastic process X. For an aggregate $X^m = \{X_l^m : l \geq 1\}$ of size m in X, *S2OSS* for $m \to \infty$ means that the associated *autocorrelation function* (ACF), namely $r^m(l)$ (for X^m) is proportional to $l^{-(2-2H)}$. *S2OSS* is LRD for its ACF is non-summable, as indicated by $r^m(l) = \sum_{l-1}^{\infty} r^m = \infty$ .The condition of " $r^m(l) \propto l^{-(2-2H)}$ for $m \to \infty$ " is mathematically equivalent to the *slowly decaying variance property*. That is, the variance of the mean of sample size m decays more slowly than m . This phenomenon is represented by the expression: $Var(X^m) \propto m^{-\beta}$. For a *S2OSS* process X and $0.5 < H < 1$ the value of $\beta = 2 - 2H$ should apply. Equations (5.3.2.1) and (5.3.2.2) summarize the *S2OSS* property and they hold for the weaker condition in equation (5.3.2.3). The *slowly decaying variance* property is clear if a log-log plot is produced for equation (5.3.2.1). As shown by equation (5.3.2.4), log($Var(X)$) is a constant, log($Var(X^m)$) versus log(m) yields a straight line with slope $-\beta$. The H value can then be calculated by the $H = 1 - (\beta/2)$ formula. The S^2 *filter* finds β for X^m on the fly. The $Var(X^m)$ calculation uses the mean value $E(X^m)$ estimated

by the M^3RT process. $E(X^m)$ is $m^{-1}\sum_{n=(l-1)m+1}^{lm} X_n$ conceptually, and the key for the S^2 *filter* operation is to choose a sufficiently large m, which is the multiples (i.e. C) of $F = 14$ to virtually satisfy $m \to \infty$; $m = C * F$ for estimating β. The detected result is available at the Ag time point. In Figure 5.3.2.1 for example, the β result for aggregate 2 is available at the point of $Ag = 2$.

$$Var(X^m) = \frac{1}{m^{(2-2H)}} Var(X) \text{.....(5.3.2.1)}$$

$$r^m(l) = r(k) \text{....(5.3.2.2)} \quad \lim_{m\to\infty} r^m(l) = r(k) \text{....(5.3.2.3)}$$

$$\log(Var(X^m)) = \log(Var(X)) - \beta \log(m) \text{....(5.3.2.4)}$$

The process in the S^2 *filter* to calculate β is the "*continuous aggregate based* (CAB)" concept, which is proposed by the COMP Team. The CAB evaluates if an aggregate is stationary by checking its Gaussian property or "Gaussianity" [Arvotham2001] by the *kurtosis* and *skewness* metrics. A symmetrical normal distribution has perfect Gaussianity indicated by $kurtosis = 3$ and $skewness = 0$. Statistically measured *kurtosis* and *skewness* values are rarely perfect, and reasonable limits can be used to indicate the presence of a bell curve, which belongs to the exponential family that is capable of independent stationary increments. The S^2 *filter* uses the CAB concept and finds β by linear regression, and the quality of which can be judged by the *coefficient of determination* or R^2 between 0 and 1 [Jain1991]. A higher R^2 implies better quality for the linear regression. By the predefined threshold Th_{R^2} (e.g. 0.85 or 85%) the S^2 *filter* can reject a hypothesis of self-

similarity in X^m for $R^2 < Th_{R^2}$. The CAB operation in Figure 5.3.2.1 works with the aggregates $X^m_{Ag=l}$ in a discrete stochastic process X along the time axis. Assuming: a) P1, P2, and P3 are the log-log plots for three successive aggregates based on equation (5.3.2.4), b) these plots yield different β values: β_1 for P1 with $R^2 = 0.82$, β_2 for P2 with $R^2 = 0.98$, and β_3 for P3 with $R^2 = 0.95$, c) $Ag = l$ is the aggregate level, and d) $Th_{R^2} = 0.9$, then both P2 and P3 confirm self-similar traffic but not P1 (for $R^2 < Th_{R^2}$). If P2 and P3 yield very different β values, their H values by $H = 1 - (\beta/2)$ indicate different dimensions or D. The D value may change over time due to various factors, for example, ON/OFF situations in the network [Willinger2003]. A changing D or H is a sign of non-linearity in the stochastic process being examined.

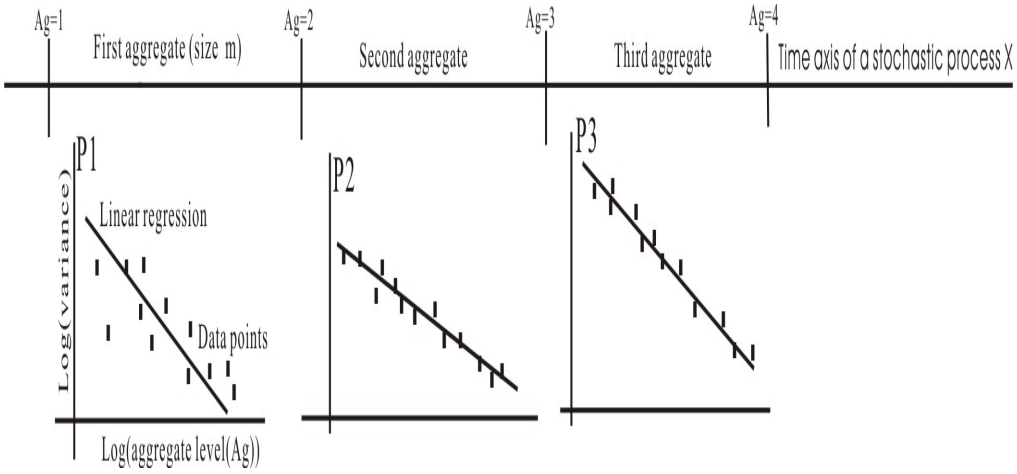


Figure 5.3.2.1 The "*aggregate based* (AB)" approach

Skewness is represented by $\frac{\sum_{i=1}^{N}(x_{i-}\bar{x})^3}{(m-1)sd^3}$, where $\bar{x}$ and *sd* are the measured mean and standard deviation respectively for the aggregate of m samples. It measures the symmetry of a bell-shaped aggregate distribution. A positive value indicates that the bell curve skews right and the right tail is heavier than the left one. *Kurtosis* is represented by $\frac{\sum_{i=1}^{N}(x_i-\bar{x})^4}{(m-1)sd^4}$, and its value decides whether the bell curve is *peaked* (for positive value) or *flat* (or negative value) compared to the normal distribution with *kurtosis=3* and *skewness = 0.*

5.3.2.1 EXPERIMENTAL RESULTS

The S^2 *filter* was verified by simulations based on the CAB approach. The experiments were conducted on the stable Aglets mobile agent platform, which is designed for Internet applications. The Aglets makes the experimental results scalable for the open Internet. The setup for the experiments is shown in Figure 5.3.2.1.1, in which the driver and server are both aglets (agile applets).

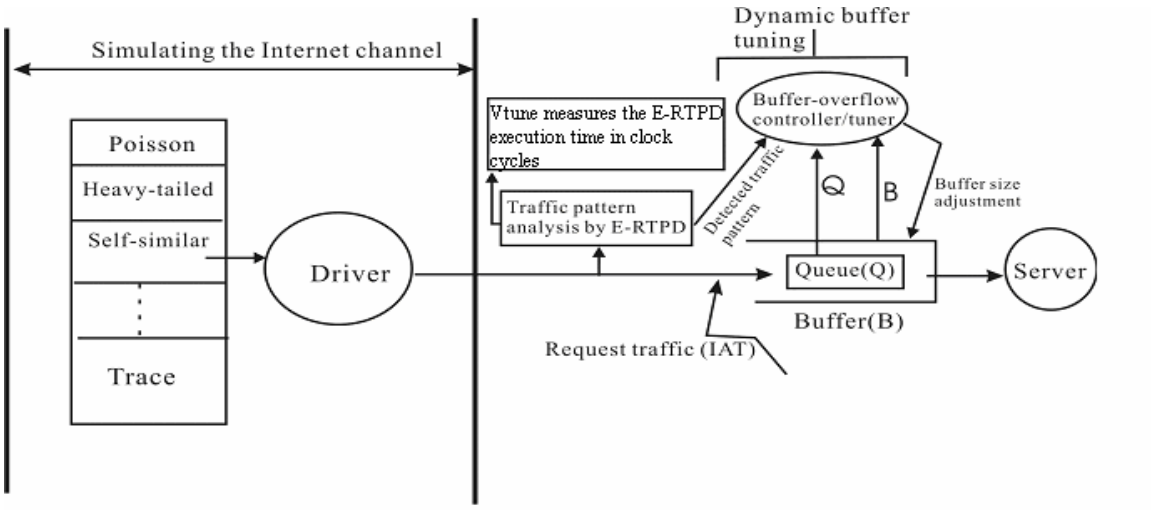


Figure 5.3.2.1.1 Setup for the S^2 *filter* experiments

The driver picks a known waveform or a pre-collected IAT trace that may embed different traffic patterns over time. The pick simulates the IAT among the requests that enter the server queue. The FLC dynamic buffer size tuner is the test-bed for the S^2 *filter*. It adjusts the buffer size on the fly by leveraging the current queue length, buffer length, and detected traffic pattern. The traffic pattern(s) that drives the IAT is also recorded by the E-RTPD that has included the S^2 *filter*. This helps matching the FLC control behavior with the specific traffic pattern. The VTune measures the E-RTPD's average execution time so that its contribution to time-critical applications on the Internet can be evaluated. Experiments with different IAT traffic patterns were carried out. The results conclude that the S^2 *filter* indeed detects self-similar traffic and helps the FLC deliver more accurate dynamic buffer size tuning. The experimental results presented here include: self-similarity detections, traffic and FLC accuracy, and D/H correlation.

Table 5.3.2.1.1 summarizes seven of the many different simulations conducted. The self-similar traces, which simulate the inter-arrival times (IAT) for the request into the server's buffer being controlled by the FLC (Figure 5.3.2.1.1), are generated by using Kramer's tool [Kramer].

β	$H = (1 - \beta/2)$	R^2 (coefficient of determination)	loading ψ	kurtosis	skewness
0.6583	0.671	0.956 (95.6%)	0.1 (10%)	0.597045	1.180861
0.6809	0.660	0.975 (97.5%)	0.2	-0.56218	0.798282
0.6425	0.679	0.977 (97.7%)	0.3	0.40215	1.277175
0.6473	0.677	0.972 (97.2%)	0.4	-0.53386	0.861215
0.4685	0.766	0.959 (95.9%)	0.5	-0.58417	0.892037
0.3762	0.812	0.885 (88.5%) (less than Th_{R^2})	0.6 (rejected)	-1.01033	0.446756
0.1978	0.901	0.605 (60.5%)	0.7 (rejected)	-1.16043	0.388599

Table 5.3.2.1.1. S^2 **filter log(variance) versus log (aggregate level) to find** β

The useful information from the Table 5.3.2.1.1 summary is listed as follows:

a) The S^2 *filter* always detects and recognizes self-similarity in the IAT traffic as long as the network loading or utilization ψ is 50% (i.e. 0.5 simulated by the same tool) or less.

b) ψ is proportional to the self-similarity dimension (explained later with Figure 5.3.2.1.7). For $\psi > 0.4$ the traffic self-similarity scales differently as indicated in Figures 5.3.2.1.3 and 5.3.2.1.4. Our analysis indicates that this is possibly the beginning of non-linear scaling or a sign of possible multifractal traffic. Both Figures 5.3.2.1.3 and 5.3.2.1.4 work with $Th_{R^2} = 0.9$.

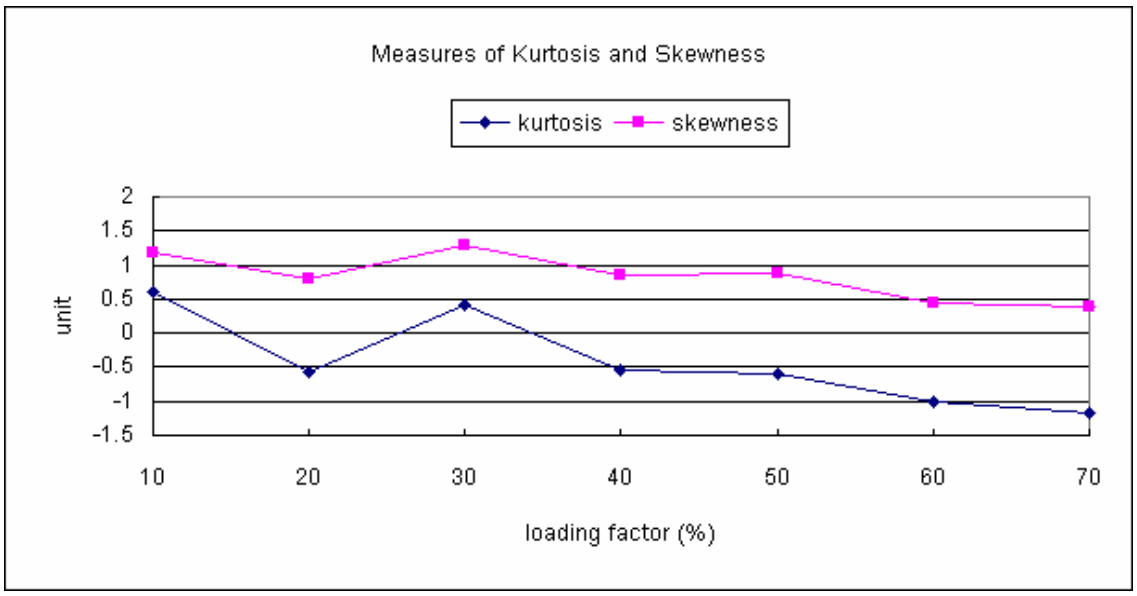


Figure 5.3.2.1.2 Kurtosis and skewness measurements for the 7 cases in Table 5.3.2.1.1

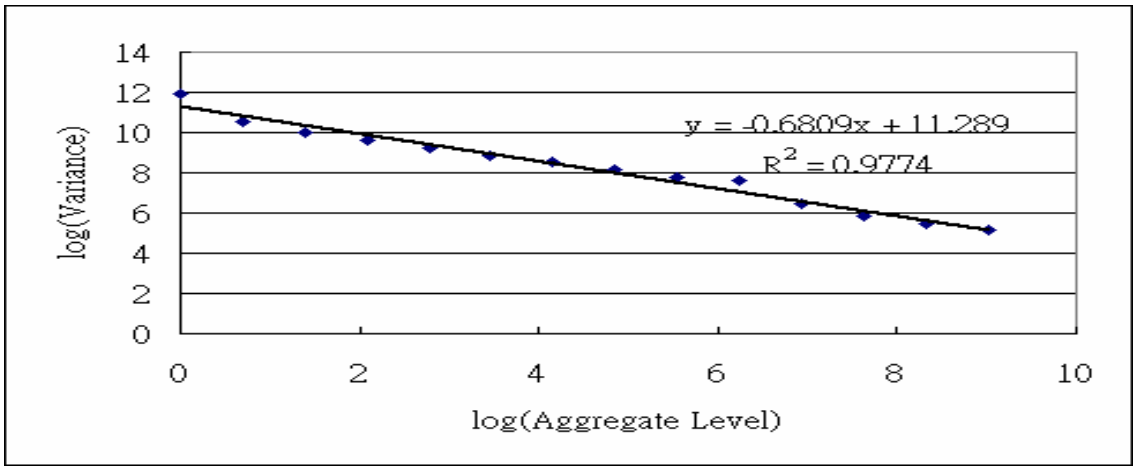


Figure 5.3.2.1.3 S^2 **filter yields slope = -0.6809(β= 0.6809),** R^2**= 97.74% for** $\psi = 0.2$

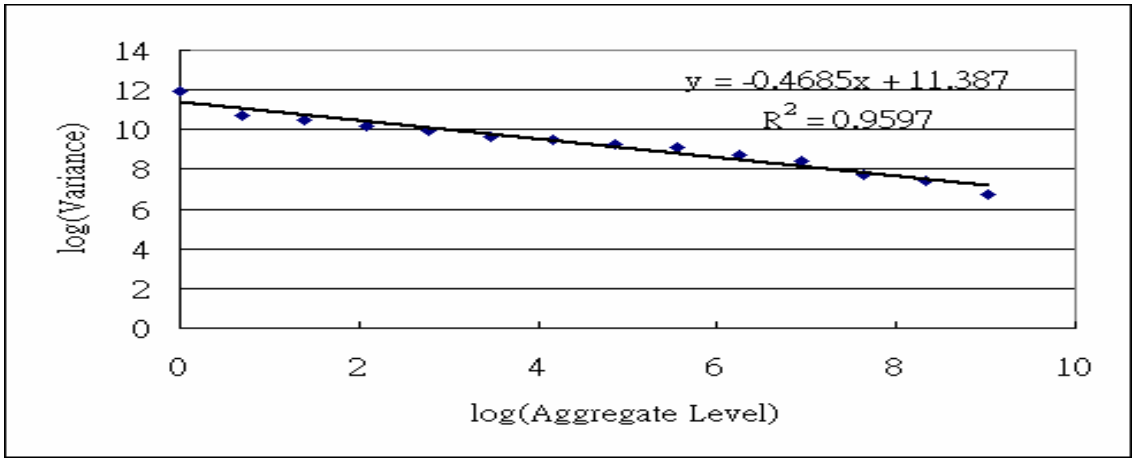


Figure 5.3.2.1.4 S^2 **filter yields slope = -0.4685(β= 0.4685), R^2= 95.97% for** $\psi = 0.5$

c) The scaling exponent H (Hurst effect) changes with ψ, which is inversely proportional to the IAT length that is the "*reduction/resolution*" in light of traffic. For $\psi \leq 0.4$ the scaling is basically the same (i.e. a *monofractal* sign). The β value in every case (row) in Table 5.3.2.1.1 is the average of several aggregates for the same stochastic process X.

d) The kurtosis and skewness are different for the different self-similar traces. Nevertheless they always indicate the presence of a bell curve.

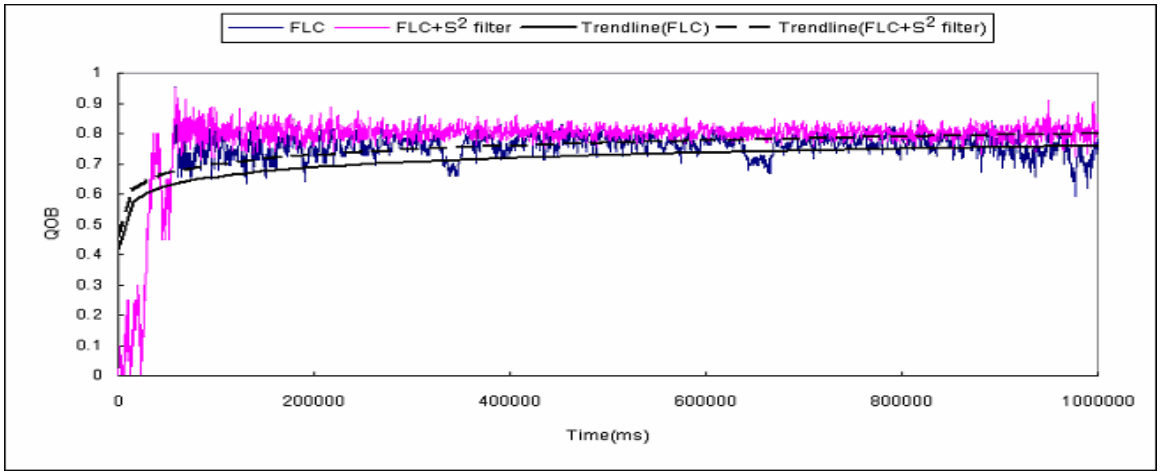


Figure 5.3.2.1.5 Faster convergence of the FLC+ S^2 filter than the FLC working alone

The kurtosis and skewness values for each case (row) in Table 5.3.2.1.1 are plotted for comparison (Figure 5.3.2.1.2). These values are obviously affected by the loading. When the loading is high (e.g. 60% and 70%) the bell curve tends to skew less but still to the right. Meanwhile the bell curve tends to get flatter. Comparatively the skewness of the bell curves for the seven simulation cases in Table 5.3.2.1.1 are less than a Weibull ($gamma = 1.5$) distribution, which is relatively more peaked (*kurtosis=4.5*).

The trend-lines in Figure 5.3.2.1.5 for the IAT traffic trace in Figure 5.3.2.1.3 shows that the "$FLC + S^2$ *filter*" combination converges much faster to the given steady state than the FLC working alone. In fact, this combination is one of the working modes in the Adaptive/Reconfigurable FLC [p12], the details of which will be discussed in section 6.2. With help from the S^2 *filter* the FLC main body adjusts the GP value for the derivative (D) control element on the fly and according to the

self-similarity property currently detected. As a result it produces less MD than the FLC working alone (Figure 5.3.2.1.6).

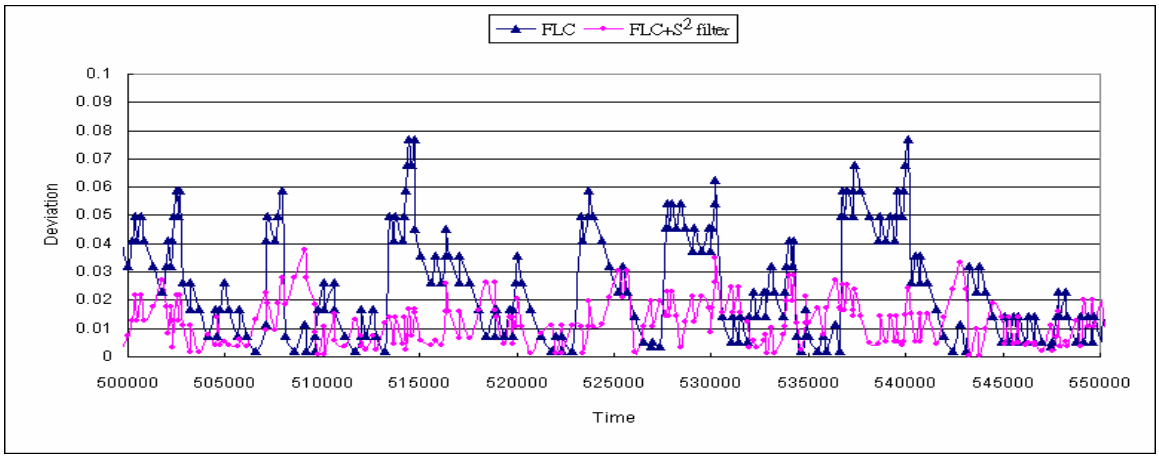


Figure 5.3.2.1.6 Less MD deviation by FLC+ S^2 than the FLC alone

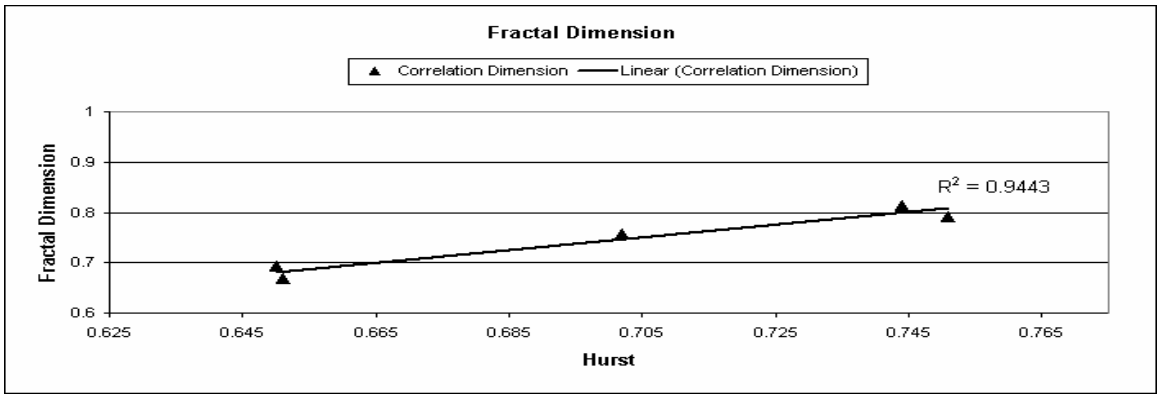
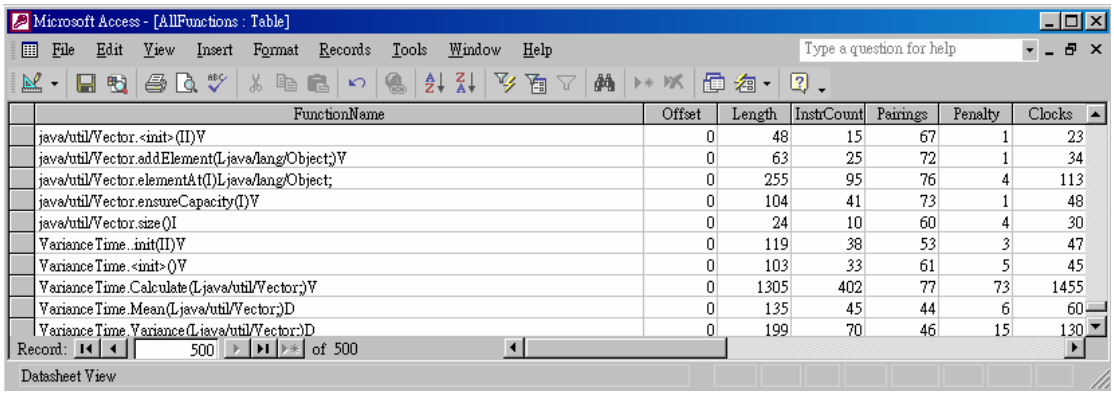


Figure 5.3.2.1.7 D/H correlation for Table 5.3.2.1.1

In the experiments the *FD3* tool [Sarraille], which confirms if an image (e.g. a time series generated by the Kramer's tool) is really fractal and measures its dimension D, was used. The purpose is to evaluate the D/H correlations [Peitgen2004]. This correlation for Table 5.3.2.1.1 is plotted and shown in Figure

5.3.2.1.7. It shows that if D changes suddenly, H also rescales accordingly to indicate possible traffic nonlinearity. In contrast, if H scales linearly, it is a sign of *monofractal* traffic. The *intrinsic* average S^2 *filter* execution time as observed from all the experiments is 1455 clock cycles as measured by the *Intel's VTune Performance Analyzer*. It is intrinsic because it works with immediately available data (without any actual IAT delay) in a trace. For a platform of 100 mega hertz the corresponding physical time is $1455/(100*10^6)$ or 14.55 micro seconds. In real-life applications the S^2 *filter* has to collect enough IAT samples on the fly before computing β. This sampling latency can be significant, and therefore the success of S^2 *filter* application depends on choosing size m for the X^m aggregate correctly. For example, if the average IAT is one second, $m = 1000$ means 1000 seconds. On the contrary for the same size m and mean IAT of 1 ms, the physical time is only one second. Therefore, the m value for the S^2 *filter* Java prototype is a variable rather than a chosen constant, and the user/tester should fix the time span T instead of collecting the fixed m samples on the fly. That is, the number of samples (i.e. m) in an aggregate within T depends on the IAT; shorter IAT delays yield a larger m. Then, the S^2 *filter* works adaptively with the m value decided by the IAT for the "*timed aggregate*" based on the chosen *T*.



5.3.2.1.8 The S^2 filter execution time (1455 clock cycles) by Intel's VTune

5.4 CONNECTIVE SUMMARY

It was observed in my previous MPhil research that changes in traffic patterns can affect the performance of the different dynamic buffer size tuners for user-level applications, namely, PIDC, GAC, FLC and NNC. Therefore, there is a need to neutralize the ill effects of changes in traffic patterns on the tuners' performance. As a result I made use of the COMP Team's accumulated experience in real-time traffic pattern detection and analysis. In return I contributed two real-time traffic filters to enhance the extant RTPD (real-time traffic pattern detector) proposed by the team. Now the RTPD is renamed *Enhanced* RTPD or E-RTPD to include my PhD contributions: the real-time modified QQ-plot and self-similarity (S^2) traffic filters. In order to confirm that these two traffic filters indeed work correctly over the Internet, which follows the power law and has widely varying traffic patterns over time, the FLC dynamic buffer size tuner is chosen for the tests. The choice is natural because there is a need to make the FLC adaptive and reconfigurable [p12]. The preliminary experience with the E-RTPD is positive and encouraging. In different

experiments the FLC made use of the E-RTPD to reconfigure itself (mainly the derivative control) to neutralize the ill effects arising from traffic characteristics successfully. The CAB concept is followed in my experiments that verify the self-similarity (S^2) filter. It involves the following stages: a) sample and examine the data in a discrete stochastic process X, b) use the sampled data to confirm the appearance of the Gaussian property, which means stationarity, c) start the S^2 filter to confirm self-similarity for the aggregates of X_l^m, where l and m identify the lag and the block size of the aggregate respectively, and d) examine the H values for all the X_l^m aggregates on the time axis to differentiate the monofractal property from the multifractal one. Monofractal property means that the H value remains virtually constant. In fact, the Gaussianity test continues and in parallel with the S^2 filter operation. If Gaussianity has disappeared, then the S^2 filter stops operation because its pivot is "*asymptotically second-order self-similarity*", which is stationary and LRD. Gaussianity confirmation at the present stage is achieved by computing the kurtosis and skewness values and comparing them to the chosen limits. The argument is that these limits help indicate the existence of a bell curve (maybe skewed), which resembles a known distribution, for example the Weibull with a known gamma value. The issue of how to choose the proper limits for real-time applications is a non-trivial one and relatively unexplored. One of the major future work items, therefore, is to deepen the investigation of how to choose the limits effectively.

CHAPTER 6 IN-DEPTH FLC RESEARCH

6.0 INTRODUCTION

The basic version of the FLC (Fuzzy Logic Controller) dynamic buffer tuning model was proposed [Lin2002]. It is basic in the sense that only two designs, namely FLC[4x4] and FLC [4x6] were investigated. The FLC is conceptually the combination: "PIDC plus fuzzy logic plus the $\{0,\Delta\}^2$ objective function". The fuzzy logic divides the PIDC control domain into a set of smaller fuzzy control regions and supports each region with either a "don't care" state or a predefined the fuzzy rule. The fuzzy rule moderates the PIDC dynamics to ensure that the controlled output would not deviate outside the $\pm\Delta$ safety band about the chosen reference. This reference point, which is symbolically represented by "0" in $\{0,\Delta\}^2$, is actually a chosen QOB (***q***ueue length ***o***ver ***b***uffer length) ratio known as the QOB_R. For the FLC [4x4] and FLC [4x6] Java prototypes different QOB_R values were tested and it was confirmed that these prototypes not only eliminated the PIDC shortcomings when they worked alone but also produced no overflow at all. The following issues were not addressed in my MPhil thesis [Lin2002]: a) the possibility of having an optimal FLC design, b) the due techniques to smoothen the FLC convergence process towards QOB_R, and c) the impact of Internet traffic patterns on the FLC's accuracy.

This section presents the results of the deeper research work on these issues in my PhD project as follows:

a) Firstly it is empirically found that an optimal FLC design range does exist [p14].
b) Secondly, the FLC can be made more adaptive by manipulating the *"don't-care range-threshold"* in a dynamic manner. This led to the proposal of the new A-FLC (Adaptive/Reconfigurable FLC [p12]) concept.
c) Thirdly, the adaptive capability of the A-FLC can be further improved with respect to different Internet traffic patterns. The investigation in this aspect created the R^2-FLC (Real-time Reconfigurable FLC).

Experiments verifying the main results have been carried out and the results obtained are presented here.

6.1 OPTIMAL FLC DESIGN

The FLC expert dynamic buffer tuner is conceptually the "*fuzzy logic plus PIDC plus the* $\{0,\Delta\}^2$ *objective function*" combination. The fuzzy logic refines and moderates the PIDC control process so that it adaptively maintains the given Δ safety/tolerance margin of the $\{0,\Delta\}^2$ objective function. By itself the PIDC control does not work by the $\{0,\Delta\}^2$ principle and therefore has no safety margin. This means there is potential overflow and buffer space wastage. The algorithmic PIDC operation (i.e. *"P+I+D" Controller*; *P* for *proportional* control, *I* for *integral* control, and *D* for *derivative* control) is shown in Figure 6.1.1.

The PIDC parameters are: ICM (*integral control mechanism*) for integral or I control, L_{now} for current buffer length, $L_{minimum}$ for the minimum buffer size estimated

from the past performance, QOB_i as the "*queue length (Q) over buffer length*" ratio in the i^{th} PIDC control cycle for proportional control, and dQ/dt as the current rate of change in Q for derivative control. ICM is defined in terms of the current RIC value. RIC uses the current mean queue length $Queue_{CA\_estimated}$ predicted by the M^3RT (*Micro Mean Message Response Time*) mechanism, as well as the mathematical average of the queue length, $Queue_{Reference}$. P_{CON} is the damping factor that smoothens the convergence towards the estimated mean M_i (now realized as $Queue_{CA\_estimate}$) for the time window of interest. The width of the i^{th} window is defined by the total time required to collect the (*F-1*) m_j^i number of samples, for $j = 1,2,..(F-1)$. F is the flush limit chosen for the M^3RT operation. M_0 is the first sample recorded after the M^3RT has started running, and M_{i-1} is the feedback of the last predicted result into the current prediction cycle. B_a is a prescribed constant or "*seed*" for the particular ICM implementation. The M^3RT mechanism is the *micro* implementation of the *Convergence Algorithm* (CA) [Wong1999B], which is derived form the *Central Limit Theorem* and predicts the mean of any waveform quickly and accurately. Being *micro* the tool runs as a logical object, which can be invoked for a prediction via message passing anytime and anywhere. In contrast a *macro* tool must be installed at the nodes that represent the two ends of a logical channel before measurement can start. The M^2RT (*Mean Message Response Time*) IEPM tool [Wong2001] is a *macro* example and the predecessor to *micro* M^3RT implementation.

If *{(dQ/dt > prescribed_positive_threshold) OR*

[(dQ/dt is_ positive) AND

(QOB_i > prescribed__positive_threshold)]}

Then $L_{now} = L_{now} + ICM;\ L_{now} \geq L_{minimum}$

Else If *{(dQ/dt < prescribed_negative_threshold) OR*

[(dQ/dt is_ negative) AND

(QOB_i < prescribed_negative_threshold)]} ***Then***

$L_{now} = L_{now-} - ICM;\ L_{now} \geq L_{minimum}$

Figure 6.1.1 The basic PID controller (PIDC) algorithm

The fuzzy logic in the FLC divides the PIDC control domain into a set of smaller fuzzy control regions for more refined operation. Each fuzzy region is then supported by a either a fuzzy rule or a "*don't care*" state. The fuzzy rules maintain the given Δ safety margin about the reference point of $\{0, \Delta\}^2$, symbolically represented by "0". For the FLC prototypes the reference point is the chosen QOB (*queue length over buffer length*) ratio or QOB_R. For QOB_R equal to 0.8 (i.e. $\Delta = 0.2$) the FLC operates in the QOB range from 0.6 to 1. The extant FLC model maintains Δ by tuning only the ICM value on the fly by the QOB and dQ/dt parameters. When the FLC control enters an inert "*don't care*" state, it requires no action. The inertness of the "*don't care*" states offsets the FLC computational complexity due to the fuzzy logic presence. As a result the FLC execution time is comparable to the much simpler

PIDC. Figure 6.1.2 is the matrix of fuzzy regions for the FLC[6x6] design. The "*dot*" defines the QOB_R value of 0.8 and X marks a "*don't care*" state.

QOB Reference		dQ/dt					
		NL	NM	NS	PS	PM	PL
0.7	ML	-	-	-	-	-	-
	SL	-	-	-	-	-	-
0.8	L	-	-	X	X	+	+
	G	-	-	X	X	+	+
0.9	SG	+	+	+	+	+	+
	MG	+	+	+	+	+	+

Figure 6.1.2 An FLC design/configuration example, FLC[6x6]

The FLC linguistic variables are:

a) *Current QOB ratio (or QOB_i)*: ML for *M*uch *L*ess than QOB_R, L for *L*ess than QOB_R, G for *G*reater than QOB_R, and MG for *M*uch *G*reater than QOB_R.

b) *Current* dQ/dt : NL for *N*egative and *L*arger than the threshold, NM for *N*egative but *M*edium to the threshold, NS for *N*egative and *S*maller than the threshold, PS for *P*ositive and *S*maller than the threshold, PM for *P*ositive and *M*edium to the threshold, and PL for *P*ositive and *L*arger than the threshold.

The FLC control decision in the i^{th} control cycle depends on the current QOB_i and dQ/dt. It may be *Addition (buffer elongation) or "+", Subtraction (buffer shrinkage) or "- "* or *don't care* "X". Different fuzzy rules can be formulated as required by different FLC designs. Some examples of the fuzzy rules for Figure 6.1.2 are as follows:

Rule 1: If (QOB_i is L) AND (dQ/dt is NM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM

Rule 2: If (QOB_i is L) AND (dQ/dt is NS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$

Rule 3: If (QOB_i is L) AND (dQ/dt is PS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$

Rule 4: If (QOB_i is L) AND (dQ/dt is PM) Then Action is "+"(Addition) AND $L_{new} = L_{old}$ + ICM

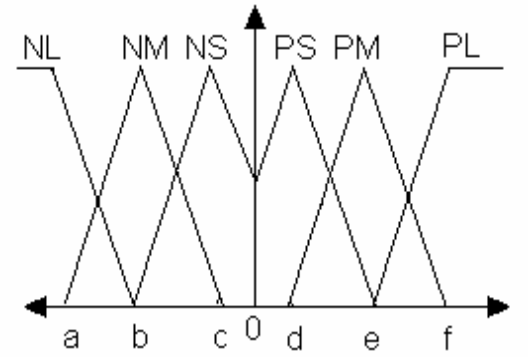


Figure 6.1.3 Membership function for dQ/dt

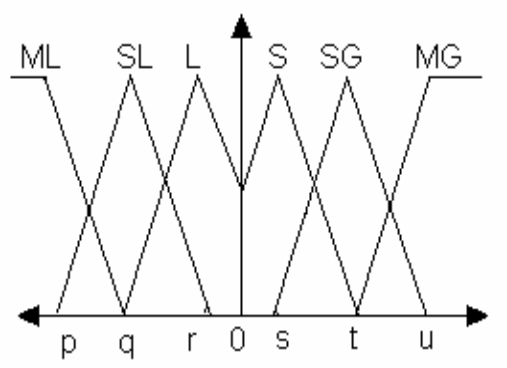


Figure 6.1.4 Membership function for QOB

The two control parameters for the FLC[6x6] design are QOB and dQ/dt. The dQ/dt *membership function* is in gradient, and the y-axis of Figure 6.1.3 is the degree of membership measurement. The x-axis is the gradient difference between two successive dQ/dt measurements. For this design the values from *a* to *f* are: *a*=0.003, *b*=0.002, *c*=0.001, *d*=0.001, *e*=0.002 and *f*=0.003. Figure 6.1.4 shows the QOB membership function for the same design, and the x-axis is the QOB ratio that changes in a dynamic manner. The values for the *p, q, r, s, t, u* are respectively: 0.65, 0.7, 0.75, 0.85, 0.9 and 0.95. The current dQ/dt and QOB values decides which fuzzy region that the FLC should operate. For example, if the degree of the dQ/dt membership function is between *b* and *c* (i.e. $y = 0.5$) and that for QOB is between q and r, four fuzzy regions are candidates: [SL,NM], [SL,NS], [L,NM] and

[L,NS]. With respect to Figure 6.1.2, the possible operations are: -,-,- and X, and the majority rule selects the minus (-) operation.

6.1.1 OPTIMAL FLC DESIGN IS POSSIBLE

There are basically two approaches to find out if an optimal FLC design is possible. The first approach is to represent the FLC in its mathematic form and use mathematical manipulation such as a theorem proving to confirm the possibility. The second approach is empirical and this means the following: a) carry out experiments for different FLC designs, and b) find the correlation between the FLC structural complexity and the chosen performance index such as the mean QOB deviation from the given QOB_R reference. The second approach is chosen for addressing the issue because the different designs have to be tested against different Internet traffic patterns [p11].

6.1.1.1 EXPERIMENTAL RESULTS

The accumulated FLC experimental data indicates that it is possible to have an optimal FLC design. Figure 6.1.1.1, which is plotted with the mean deviations from QOB_R by the different FLC designs tested in different experiments, reveals approximately where the optimal region is. This region may vary with respect to different traffic patterns. In this plot the mean deviation from the QOB_R reference stabilizes around 0.02 (or 2%) after the FLC[4x6] design. More experiments confirm

that more complex FLC designs do not yield less deviation after this point. The possibility of having optimal FLC designs makes it worthwhile to explore the correlation among the following in the future work: the mean deviation from QOB, the FLC design complexity (i.e. the matrix size), and the metrics being leveraged (e.g. traffic pattern). If this correlation could be formally established, then proper intelligence could be incorporated to let the FLC timely auto-tune its configuration (i.e. the matrix size and the matrix entries) for cost and effectiveness. A simpler FLC design is always desirable because it yields shorter execution and better RTT timeliness to enhance the chance of success for time-critical applications.

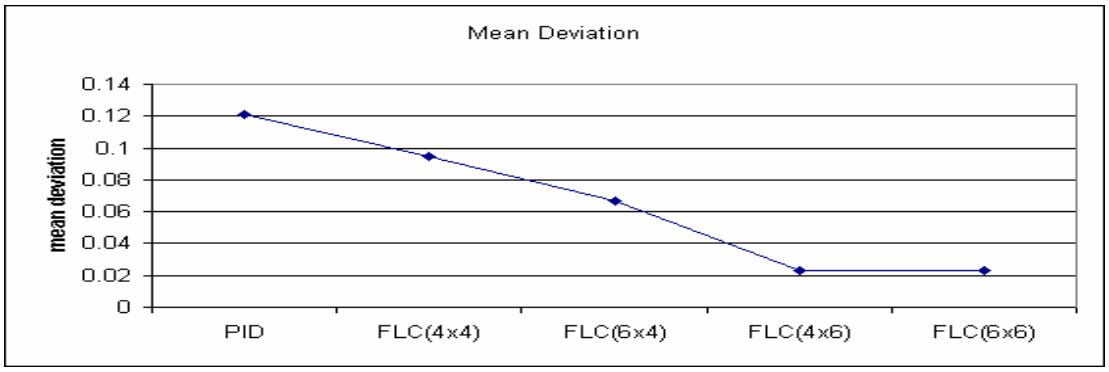


Figure 6.1.1.1 An optimal FLC design is possible (mean deviation stabilizes around 0.02)

6.2 THE ADAPTIVE/RECONFIGURABLE FUZZY LOGIC CONTROLLER (A-FLC)

The A-FLC (adaptive/reconfigurable FLC) model uses a static adjustment size to tune the "*don't-care region's range-threshold*" by using a static adjustment size. The A-FLC is basically the combination: "*don't-care region's range-threshold auto-*

tuning capability + FLC". The PIDC component in the FLC uses two static thresholds to achieve its control purpose, namely, *Th1* and *Th2*. The PIDC working alone conceptually should have four control regions, defined by different $\pm$ *Th1* and $\pm$ *Th2* combinations. The FLC fuzzy logic divides these thresholds into the finer membership functions, with *range-thresholds* among them (e.g. the range-threshold in Figure 6.1.4 between $a = 0.003$ and $b = 0.002$ is 0.001). In the original FLC the range-thresholds are static and decide which region the dynamic buffer tuner should operate at the time. The static nature of the range-thresholds introduces an intrinsic delay for the corrective action by a fuzzy control. For example, if the dQ/dt value increases but less than the range-threshold of the current "*don't care*" region, there will be no control action. By the time any control action is triggered (range-threshold exceeded) there would be significant overshoot or undershoot already. The overshoot/undershoot accumulations make the FLC control process oscillate. The A-FLC differs by preventing any significant overshoot/undershoot proactively in a timely manner. The prevention is achieved by adapting the range-threshold of the current "*don't care*" fuzzy region on-line. At anytime, if the increase (or decease) of dQ/dt is more than (or less than) the given "*gradient threshold* (GT)", the range-threshold is adjusted by the "*given percentage* (GP)" in a timely manner. Even though the GT and GP are fixed values, the experimental results shows that the A-FLC has much better performance than the original FLC by yielding less mean deviation from the QOB_R reference. Figure 6.2.1 shows how the "*don't care*" range-threshold of a growing dQ/dt value is "*squeezed*" to enlarge the range-threshold on

the right side (now larger than *Ra*). This enlargement, in effect, urges the A-FLC to take immediate action instead of waiting passively for the "dQ/dt *predefined range-threshold*" condition to hold. The dynamic "*squeezing*" action based the GT and GP parameters quicken the A-FLC response. In the R^2*-FLC* model GP is <u>*adjusted again*</u> in a dynamic manner with respect to the traffic pattern identified by the RTPD capability.

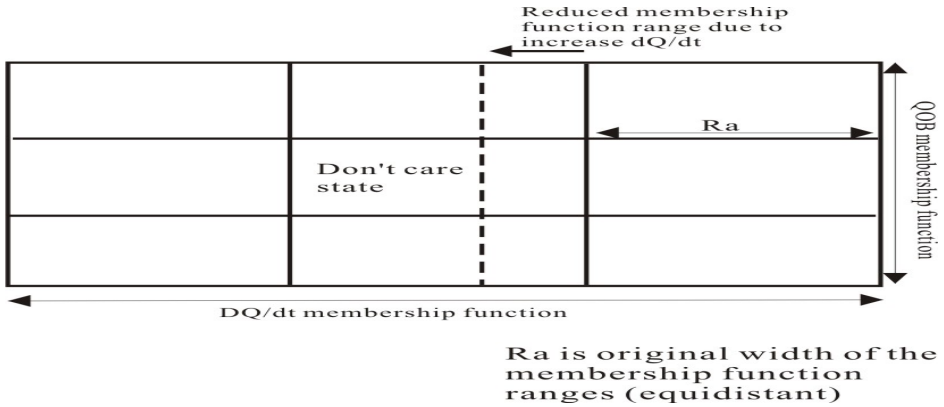


Figure 6.2.1 A-FLC adjustment of the range threshold of the *don't care* state on the fly

6.2.1 EXPERIMENTAL RESULTS

Different experiments were conducted over the Internet to verify the A-FLC prototype implemented in Java. The preliminary results indicate that it is indeed more efficient and less oscillatory than its FLC predecessor. Different QOB_R and Δ values were tried in the experiments, and the results presented here are based on: $\Delta = 0.2$ (or 20%), and $QOB_R = 0.8$ (or 80%). The QOB_R value is the reference point chosen for the $\{0, \Delta\}^2$ objective function. Similar to its predecessor the A-FLC never failed to upkeep the Δ safety/tolerance margin in the experiments. Figure 6.2.1.1

shows both the deviations and the MD of the A-FLC control that auto-tuned the range threshold over the entire control process. The MD is measured by the M^3RT component in the A-FLC. If the A-FLC operation stops the last MD value together with the given safety margins: *Th1* (for QOB or P control) and *Th2* (for dQ/dt or D control) become a new point, namely, *[MD,(Th1,Th2)]* in the *MD-vs-Thresholds* graph. In this way the graph records the past experience for the future determination of sounder *Th1* and *Th2* initialisations for the A-FLC to run again. Figure 6.2.1.2 shows the A-FLC deviation situations with the same set of data as for Figure 6.2.1.1 but the capability to auto-tune the range threshold for "dynamic threshold" operations is absent. The absence makes the deviations more prominent. Figure 6.2.1.3 shows Figure 6.2.1.1 and Figure 6.2.1.2 in a comparative manner to make the difference in the deviations conspicuous. Figure 6.2.1.4, 6.2.1.5 and 6.2.1.6 are plots for another experiment. The result from this set concurs with the observations for the previous one. In fact, all the experiments indicate that auto-tuning of the range thresholds for the "*don't care*" fuzzy regions are important for yielding smoother, more responsive buffer overflow control. For the experimental results presented in this paper, the GT and GP values are respectively 0.003 radians and 5%.

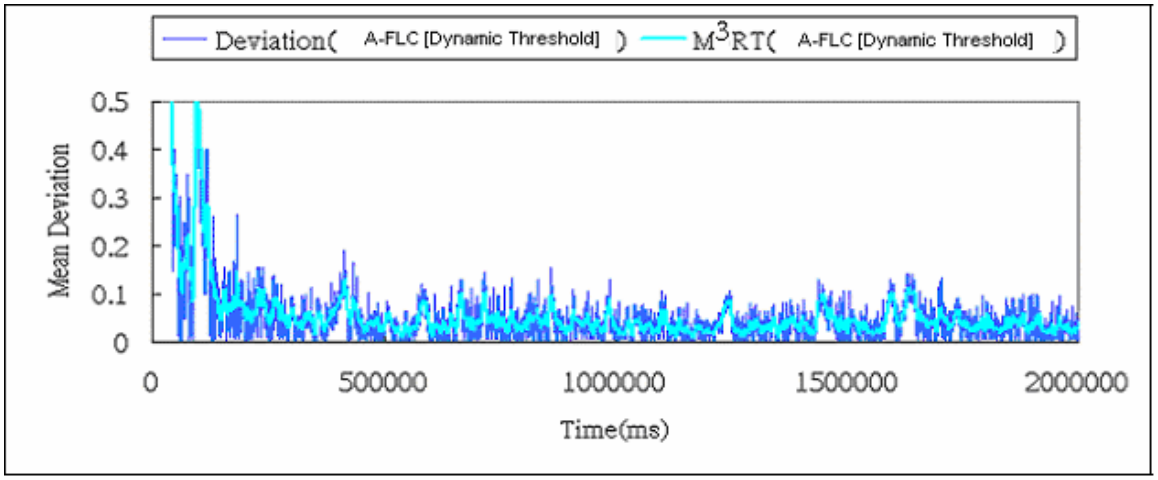


Figure 6.2.1.1 MD value by the M^3RT over time for A-FLC with "dynamic threshold"

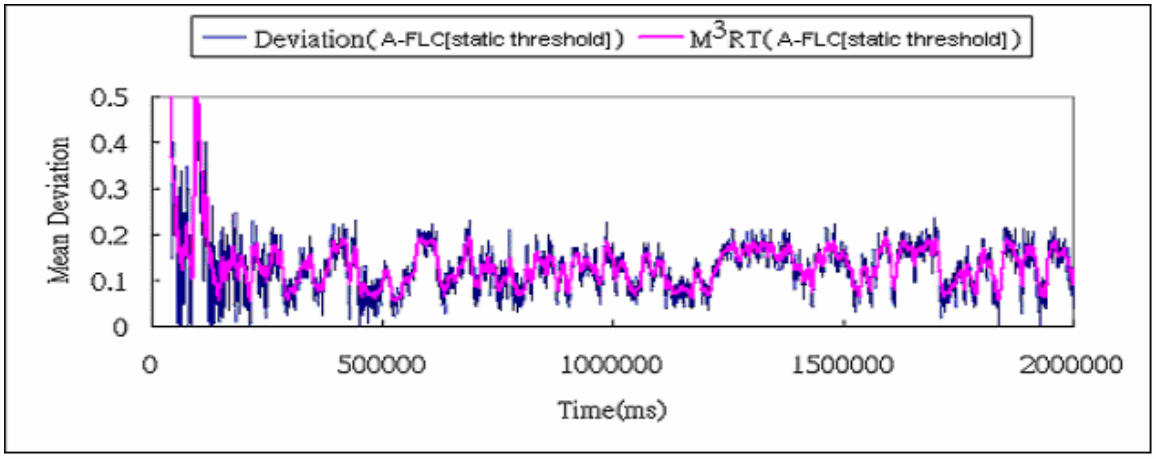


Figure 6.2.1.2 MD value by the M^3RT over time for A-FLC "static threshold"

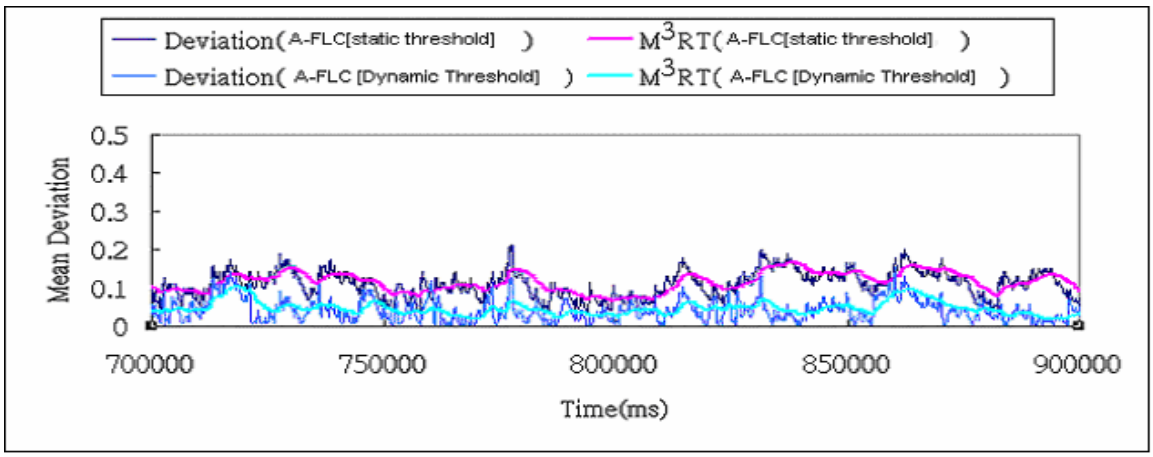


Figure 6.2.1.3 Comparing the A-FLC[static threshold] and the A-FLC[dynamic threshold]

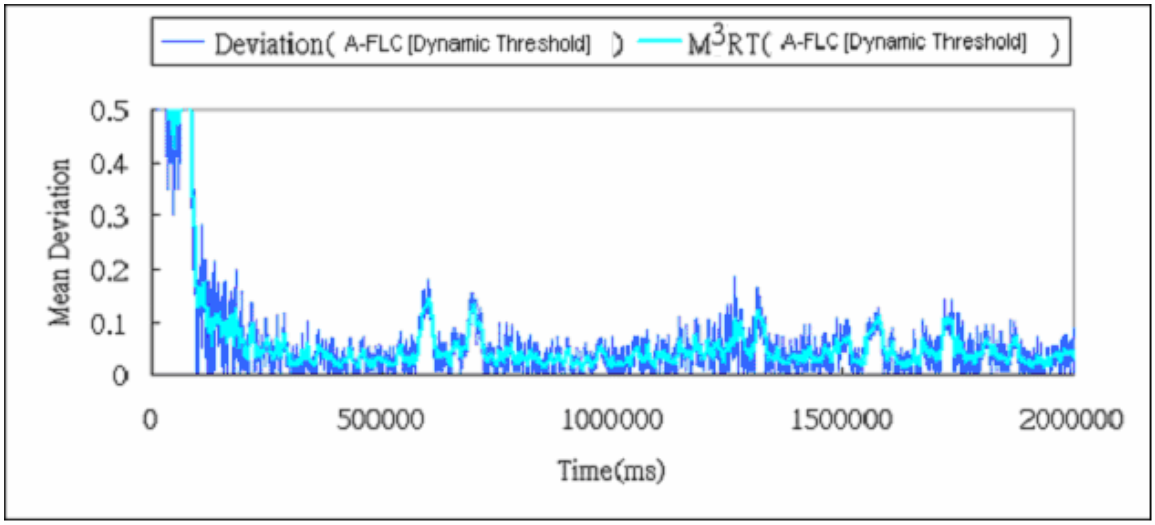


Figure 6.2.1.4 MD value by the M^3RT over time for A-FLC dynamic threshold

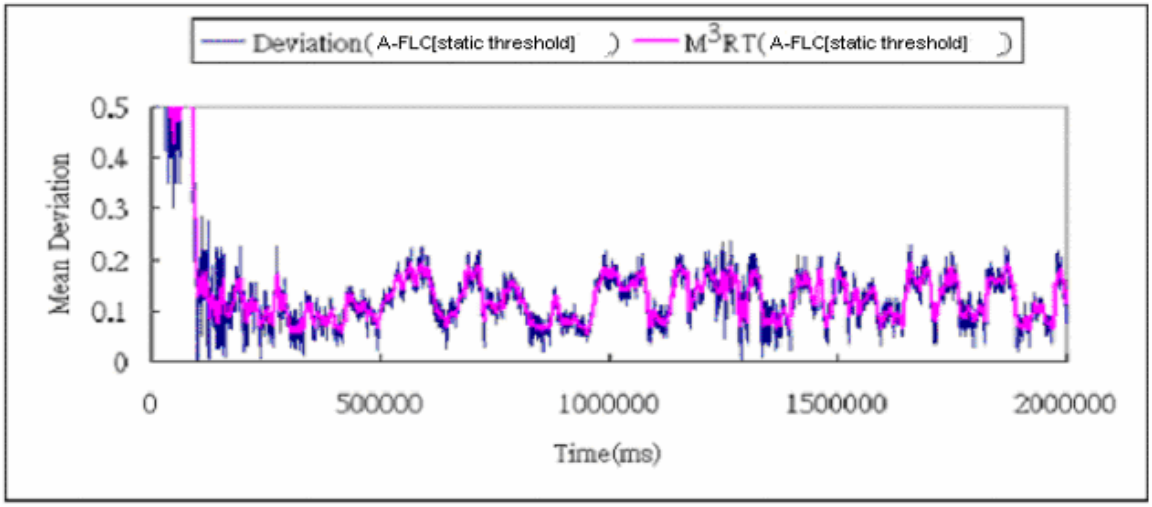


Figure 6.2.1.5 MD value by the M^3RT over time for A-FLC static threshold

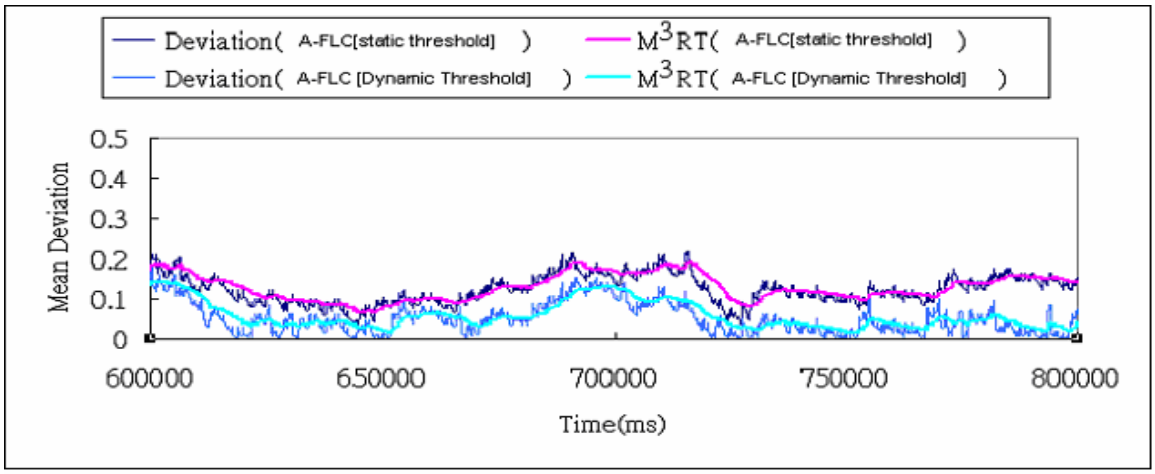


Figure 6.2.1.6 Comparing A-FLC[static threshold] and the A-FLC[dynamic threshold]

6.3 THE REAL-TIME RECONFIGURABLE FUZZY LOGIC CONTROLLER

The novel R^2-*FLC* model has two main components, namely, the A-FLC (*adaptive fuzzy logic controller*) and the RTPD (*real-time traffic pattern detection*). The A-FLC reconfigures itself on-line with respect to the traffic pattern currently

identified by the RTPD capability. It achieves this by using the current RTPD result to tune the RAC capability, which tunes GP in a dynamic manner. The RTPD, as explained in Chapter 5, is statistical by nature. Therefore, its traffic detection and identification accuracy is independent of the traffic changes in the Internet, which follows the power law. Over time the Internet traffic pattern switches frequently, for example, from SRD to LRD or multifractal [Leland1994, Paxson1995, Crovella1997].

To recap, the RTPD carries out the following on the fly: a) differentiates LRD from SRD by measuring the *Hurst* (H) effect/value, and b) identifies the traffic pattern (e.g. heavy-tailed) through a filtration process. The H value indicates LRD behaviour for $0.5 \le H \le 1$ and SRD for $0 < H < 0.5$. In the filtration process the appropriate filter is invoked to identify the specific traffic pattern (e.g. the *modified QQ-plot* filter identifies "*heavy-tailedness*"). Many methods/algorithms in literature can be adopted for measuring the H value or to differentiate LRD from SRD. The examples include the *R/S* (*rescaled adjusted statistics*), *periodogram* and *whittle* estimators. After experimenting with different estimators from literature (e.g. [Molnár1999]) the R/S is chosen as the backbone for the RTPD capability. This choice is natural because the R/S estimator requires only simple calculations and is therefore naturally suitable for time-critical applications. In contrast any complex calculations would lead to serious time delay and higher chance of deleterious effects. The traditional R/S (*rescaled adjusted statistics*) estimator (Molnár [Molnár1999] and others) is used primarily for analyzing pre-collected traces: $R/S = \frac{\max\{W_i : i = 1,2,....,k\} - \min\{W_i : i = 1,2,...,k\}}{\sqrt{\text{var}(X)}}$. In the R/S expression W_i and

$\overline{X}$ are represented by the following: $W_i = \sum_{m=1}^{i}(X_m - \overline{X})$ for $i = 1,2,...k$ and $\overline{X} = 1/k \sum_{i=1}^{k} X_i$. The best value for *k*, however, should be found by trial and error. This is the drawback of the traditional R/S estimator because the R/S accuracy and speed depend on *k*. The R/S ratio is the *rescaled range* of the stochastic process over a time interval *k*, where X is the discrete time for $\{X_i : i = 1,2,...k\}$. The most useful R/S feature is the relationship for a large *k*: $R/S = (k/2)^H$. The H (Hurst) effect/value is the slope of the *log-log* plot: *log(R/S) versus log(k)*. The filtration process is invoked after the traffic differentiation stage. For example if the LRD traffic type is recognized, the *modified QQ-plot* and *De Haan's moment* filters can be then invoked from the library to confirm the "*heavy-tailedness*" of the recognized LRD waveform by consensus. Similarly the "$\delta = \mu$" *filter*, where δ and μ are the standard deviation and mean of the waveform, confirms a Poisson process. For the *R^2-FLC* the M^3RT micro IEPM technique is adopted to support quicker and more accurate W_i computations. That is, $W_i = \sum_{m=1}^{i}(X_m - M_i)$ is used instead of the traditional $W_i = \sum_{m=1}^{i}(X_m - \overline{X})$ approach. To summarize the RTPD essence consists of:

a) It runs as a traditional R/S computation if the M^3RT support is inhibited.

b) The M^3RT converts the traditional R/S into the *enhanced R/S* (E-R/S) estimator.

c) The M^3RT estimates the mean of the waveform (i.e. M_i) with F=14 samples.

d) The E-R/S computes with $W_i = \sum_{m=1}^{i} (X_m - M_i)$ but not the $W_i = \sum_{m=1}^{i} (X_m - \overline{X})$.

The *R²-FLC* uses the RTPD result to reconfigure itself on-line by auto-tuning its RAC capability. In the process the RAC selects a more appropriate GP value for its operation. Both M^3RT and RTPD capabilities are realized as logical object entities in the *R²-FLC* prototype. They run in parallel with the *R²-FLC* main body, namely, the A-FLC. Timing analysis with the *Intel's VTune Performance Analyzer* [VTune2002] shows the following (explained in more details later): a) the RTPD with M^3RT support needs an average of 890 clock cycles to execute and 950 without, and b) the *R²-FLC* module, which runs in parallel with the RTPD, needs 350 clock cycles to execute on average, and this means the *de facto R²-FLC* execution time depends on the RTPD.

The *Th1* and *Th2* thresholds for the PIDC component in the *R²-FLC* model are not assigned by the user but generated by the system automatically from past performance. The important salient feature for the *R²-FLC* model is the *MD-vs-Thresholds* graph, which accumulates all the previous *R²-FLC* experience. Whenever *R²-FLC* has stopped running it contributes a new point, which is defined by the last three values used: MD (mean deviation), *Th1* and *Th2*. If the *R²-FLC* control is started again, from the [MD, (*Th1*, *Th2*)] points a good estimate for the best starting *Th1* and *Th2* values will be automatically determined. The best choice is the pair that yields the minimum MD value on the *MD-vs-Thresholds* graph. The *R²-FLC* needs the M^3RT object in the MD computation.

In order to demonstrate how traffic patterns affect *R²-FLC* performance, some of the results obtained with the FLC[6x6] design are presented here (Figure 6.3.1). The

significant observation is that different traffic patterns need different GP values to yield the same MD value from the QOB_R reference. For example, for MD=0.026 the R^2-*FLC* needs GP=0.07 (or 7%) for Poisson, GP=0.08 (or 8%) for heavy-tailed (Pareto), and $GP > 0.1$ (or 10%) for self-similar traffic. This means that the GP value should be chosen appropriately in a real-time fashion for attaining better dynamic buffer tuning cost effectiveness, with respect to different traffic patterns. Detecting/identifying the exact traffic pattern and selecting the corresponding correct GP value is *real-time reconfiguration* in the R^2-*FLC* context.

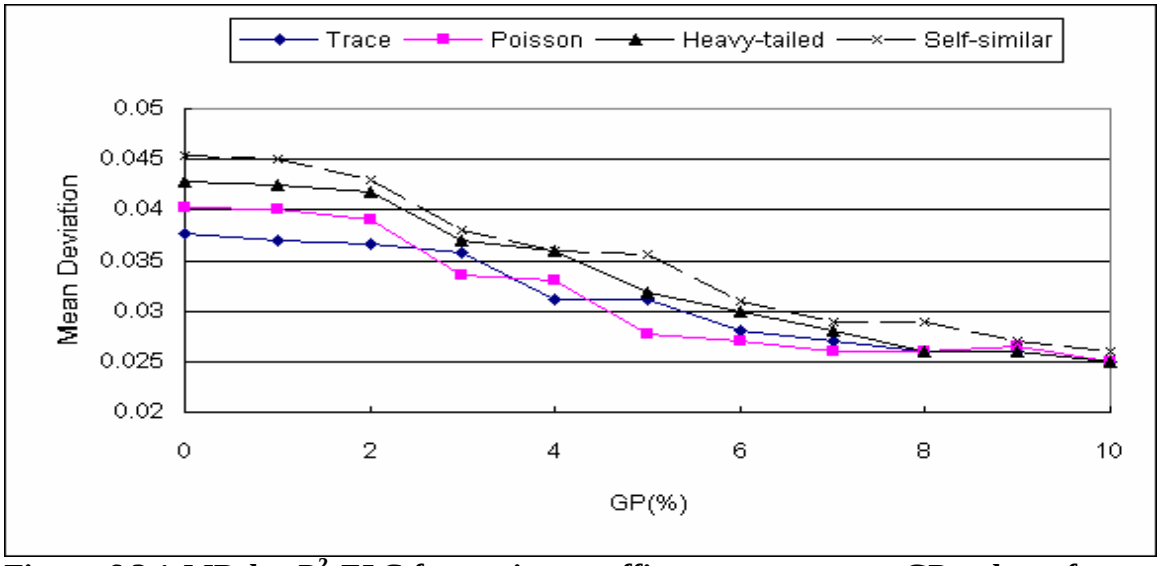


Figure 6.3.1 MD by R^2-*FLC* for various traffic patterns versus GP values, for FLC[6x6]

6.3.1 EXPERIMENTAL RESULTS

Experimental results shows that for the same FLC design bases (e.g. FLC[6x6]) the R^2-*FLC* performance is upward compatible to the *A-FLC* tuner. In the

experiments different FLC design bases and traffic patterns were involved. For example, Figure 6.3.1.1 plots the mean deviations from the QOB_R reference produced by the *R^2-FLC* with different FLC design bases versus different traffic patterns. Each mean deviation (MD) value in the plot is for a FLC design basis versus the specific traffic pattern. It is the average of the results from ten separate experiments with different TCP traces. For comparison purposes the mean deviation for by the PIDC tuner is also shown. For the results shown in the plot, the *R^2-FLC* works with the initializations: GT=0.003 radians and GP=0.05 (i.e. 5%). The striking similarities between the *R^2-FLC* and *A-FLC* are as follows: a) they show the same trend of mean deviations with the same FLC basic design matrices, b) they both yield the same optimal FLC design range (Figure 6.3.1.1), as well as no more mean deviation reduction than the more complex FLC[4x6] design, and c) the R^2-FLC model produces less mean deviation on average than the *A-FLC*, with or without M^3RT support (Figure 6.3.1.2).

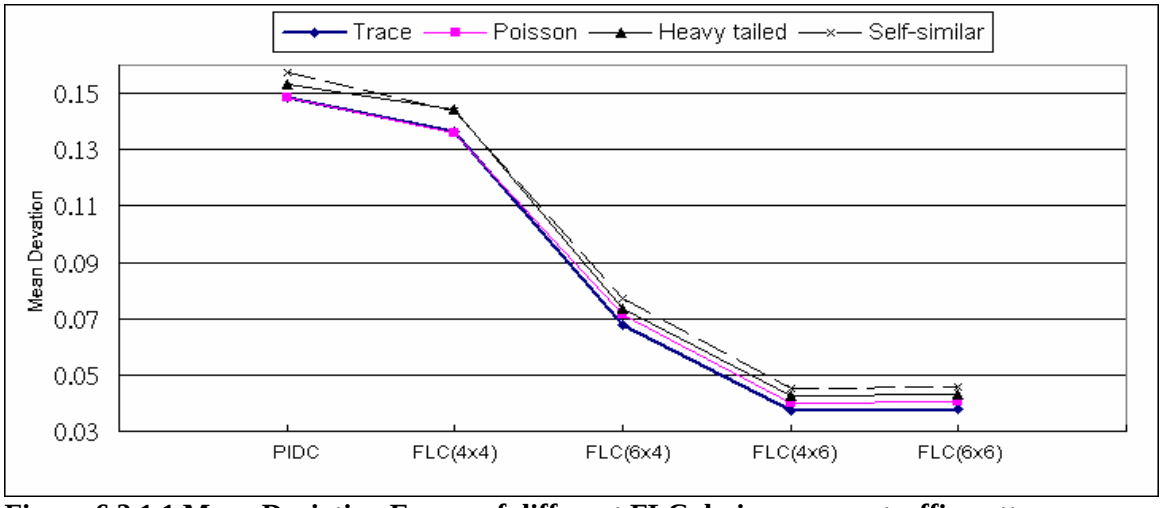


Figure 6.3.1.1 Mean Deviation Errors of different FLC designs versus traffic patterns

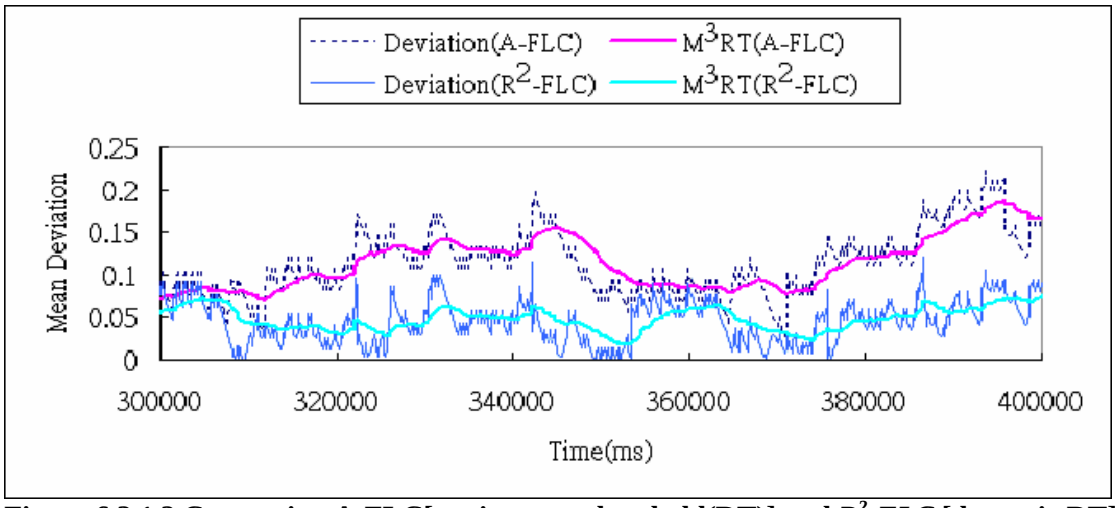


Figure 6.3.1.2 Comparing A-FLC[static range threshold(RT)] and *R^2-FLC* [dynamic RT]

The execution times of the Java *R^2-FLC* prototype were measured against different GP values. The measurements in terms of the number of neutral clock cycles were carried out with the *Intel's VTune Performance Analyzer* [VTune2002]. For all the experiments the prototype needs less than 400 clock cycles to execute its control pass/cycle. For example, Figures 6.3.1.3 and 6.3.1.4 show the prototype execution times for the Poisson and heavy-tailed traces. The given/static GP values for these two cases are 5% and 7% respectively, for the given mean deviation of 0.027. The *VTune* measurements show that the *R^2-FLC* needs only 280 clock cycles for the Poisson trace but 340 clock cycles for the heavy-tailed one. If the RTPD component detects that the traffic pattern has changed from heavy-tailed to Poisson, the *R^2-FLC* tuner should self-configure immediately to deal with the situation. This means squeezing the GT by 5% instead of 7% and the action improves the dynamic buffer tuning cycle time by (340 - 280)/340 or 17.65 %. As a result this lessens the chance of having deleterious effects for the buffer size tuning process.

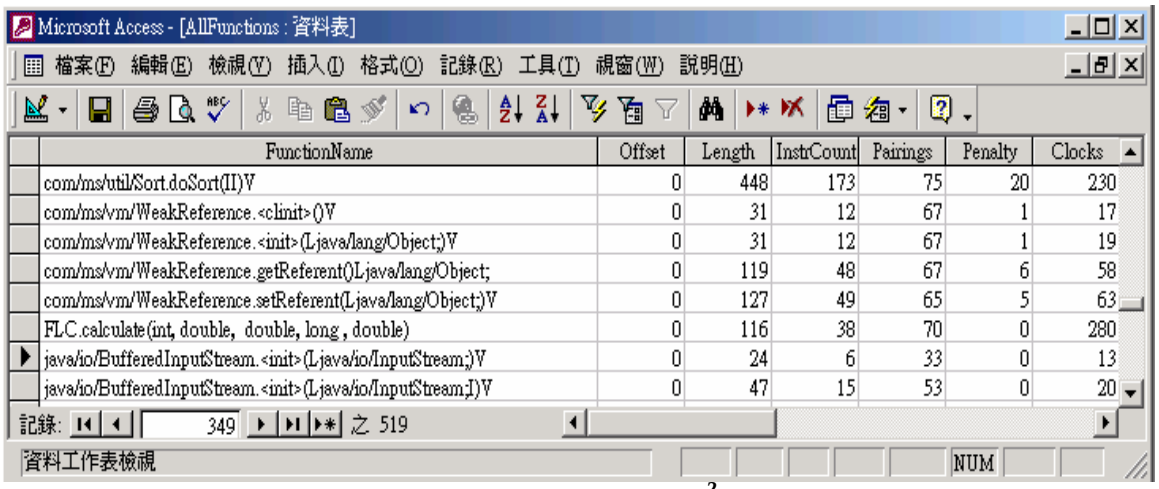


Figure 6.3.1.3 For GP=5% and MD=0.027, the *R^2-FLC* execution time is 280 clock cycles for the Poisson distribution

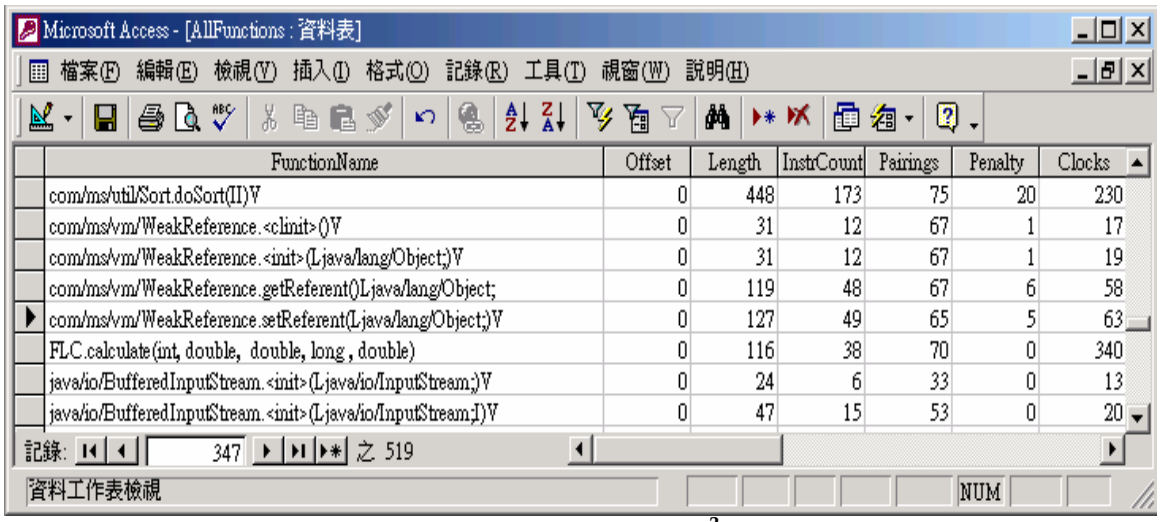


Figure 6.3.1.4 For GP=7% and MD=0.027, the *R^2-FLC* execution time is 340 clock cycles for the heavy-tailed distribution

The preliminary experimental results indicate that the *R^2-FLC*, with or without the support of the RTPD capability, is consistently more accurate than the A-FLC and the FLC predecessors by yielding less MD values. This is demonstrated by Figures 6.3.1.5, 6.3.1.6 and 6.3.1.7, in which for GP=5% (or 0.05) the novel *R^2-FLC* tuner

with the FLC[6x6] design basis consistently produces less mean deviations than the basic FLC[6x6] and the more adaptive A-FLC[6x6] version. The three different traffic traces used in the experiments were: Poisson, heavy tailed, and self-similar. The different experiments confirm that the *R^2-FLC* tuner consistently has better performance than the FLC and the A-FLC predecessors.

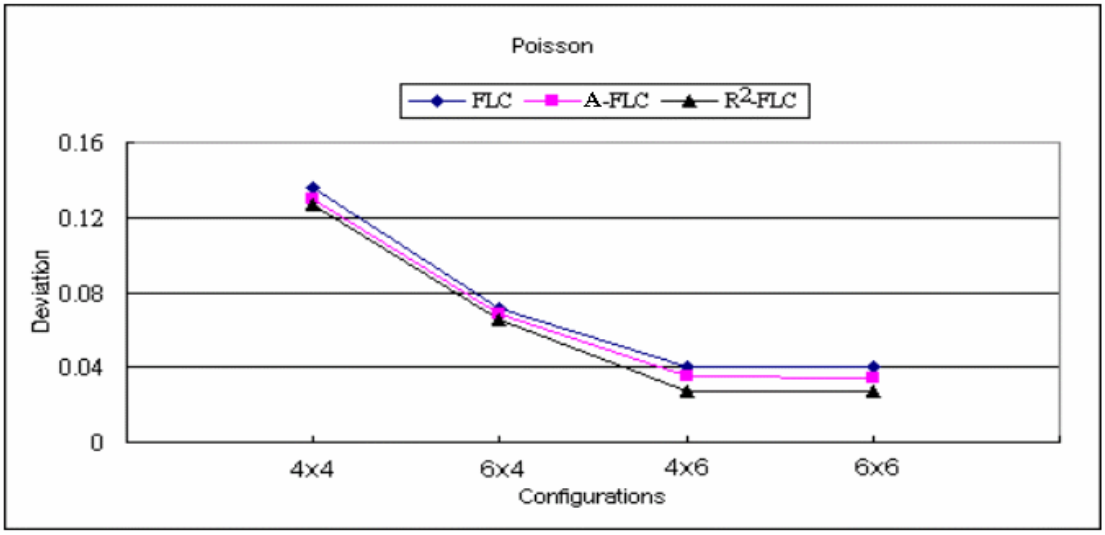


Figure 6.3.1.5 Better *R^2-FLC* [6x6] performance than FLC[6x6] and A-FLC[6x6] (alternatively known as R-FLC[6x6]) for the Poisson trace , GP=0.05

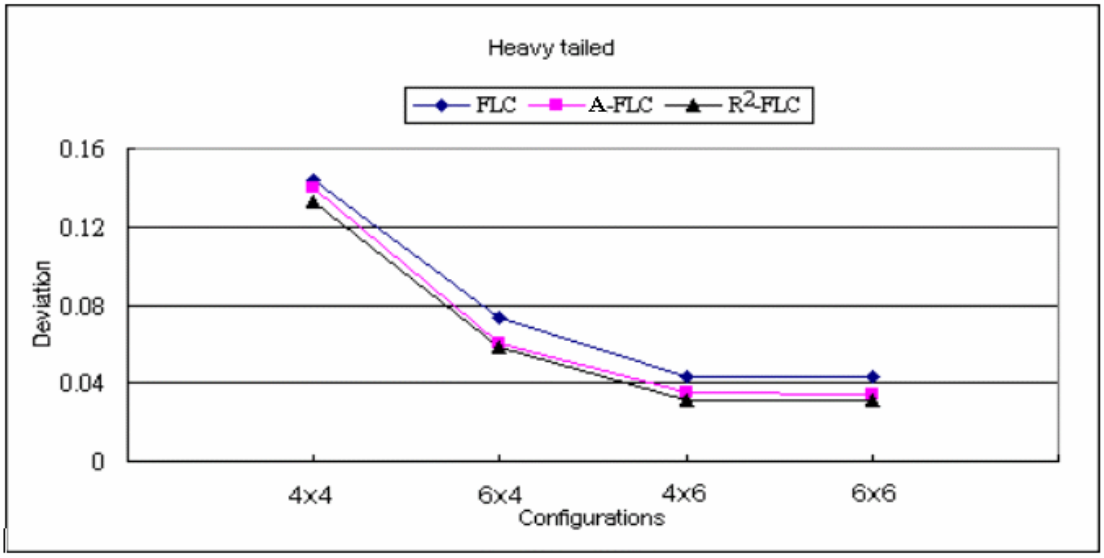


Figure 6.3.1.6 Better *R^2-FLC* [6x6] performance than FLC[6x6] and A-FLC[6x6] for the heavy-tailed trace, GP=0.05

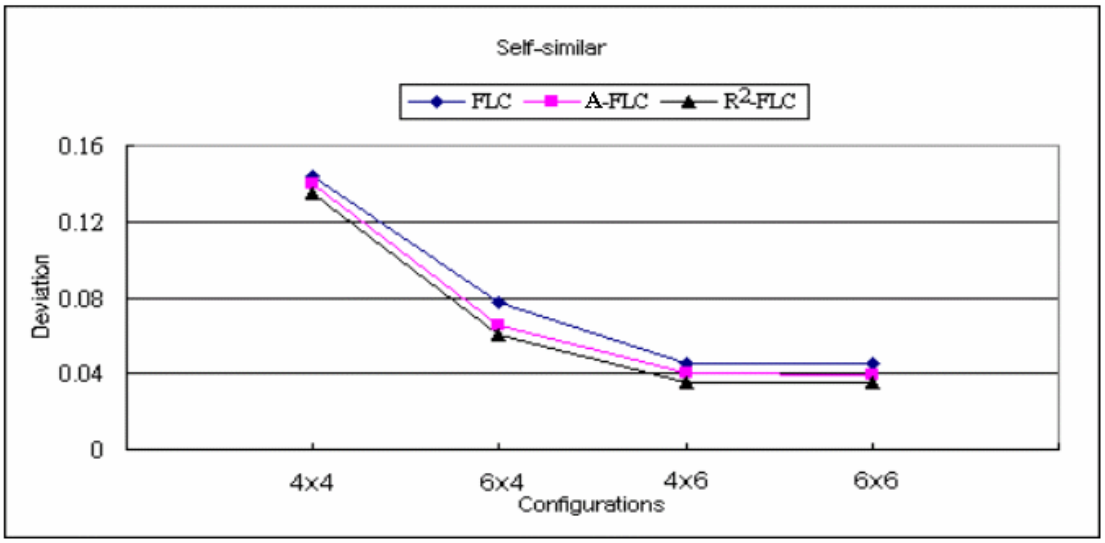
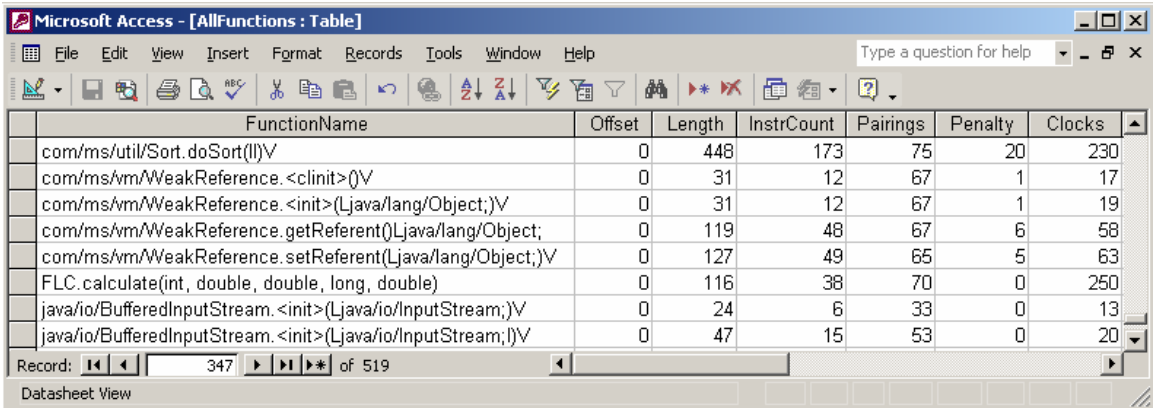


Figure 6.3.1.7 Better *R^2-FLC* [6x6] performance than A-FLC[6x6] for the self-similar trace, GP=0.05

6.4 TIMING ANALYSIS OF THE THREE FUZZY LOGIC CONTROLLERS

The timing analyses of three fuzzy logic controllers (FLC, A-FLC and R^2-FLC) were carried out with the Intel's VTune Performance Analyzer [VTune2002]. Different traffic distributions, which include known waveforms (e.g. Poisson and heavy-tailed) as well as Internet traffic traces, were used in the experiments. Some of the experimental results were selected for demonstration in this section.

6.4.1 FLC



FunctionName	Offset	Length	InstrCount	Pairings	Penalty	Clocks
com/ms/util/Sort.doSort(II)V	0	448	173	75	20	230
com/ms/vm/WeakReference.<clinit>()V	0	31	12	67	1	17
com/ms/vm/WeakReference.<init>(Ljava/lang/Object;)V	0	31	12	67	1	19
com/ms/vm/WeakReference.getReferent()Ljava/lang/Object;	0	119	48	67	6	58
com/ms/vm/WeakReference.setReferent(Ljava/lang/Object;)V	0	127	49	65	5	63
FLC.calculate(int, double, double, long, double)	0	116	38	70	0	250
java/io/BufferedInputStream.<init>(Ljava/io/InputStream;)V	0	24	6	33	0	13
java/io/BufferedInputStream.<init>(Ljava/io/InputStream;I)V	0	47	15	53	0	20

Figure 6.4.1.1 FLC execution time is 250 clock cycles for the Poisson distribution

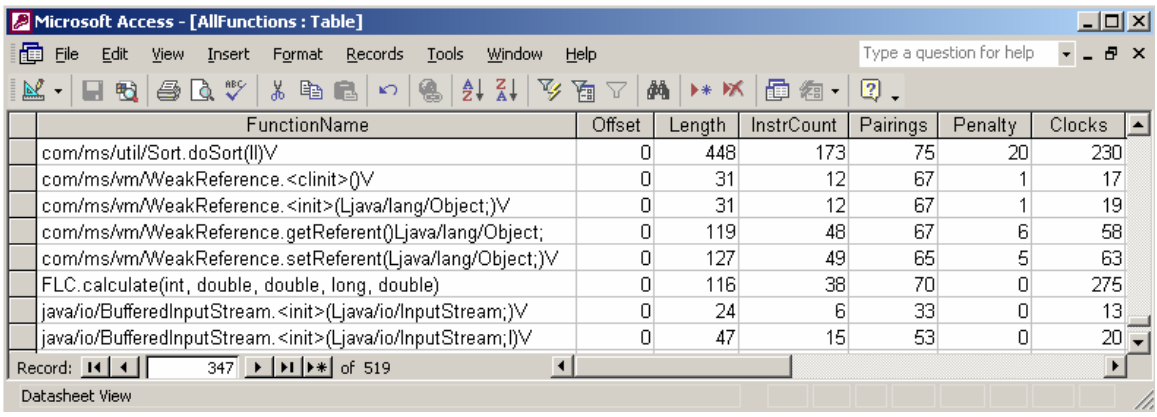


Figure 6.4.1.2 FLC execution time is 275 clock cycles for the heavy-tailed distribution

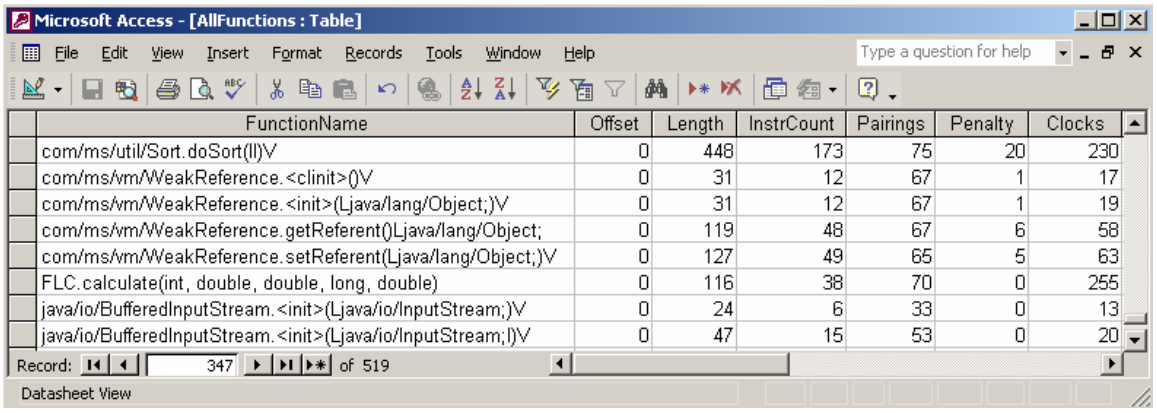


Figure 6.4.1.3 FLC execution time is 255 clock cycles for the trace [Trace]

6.4.2 A-FLC

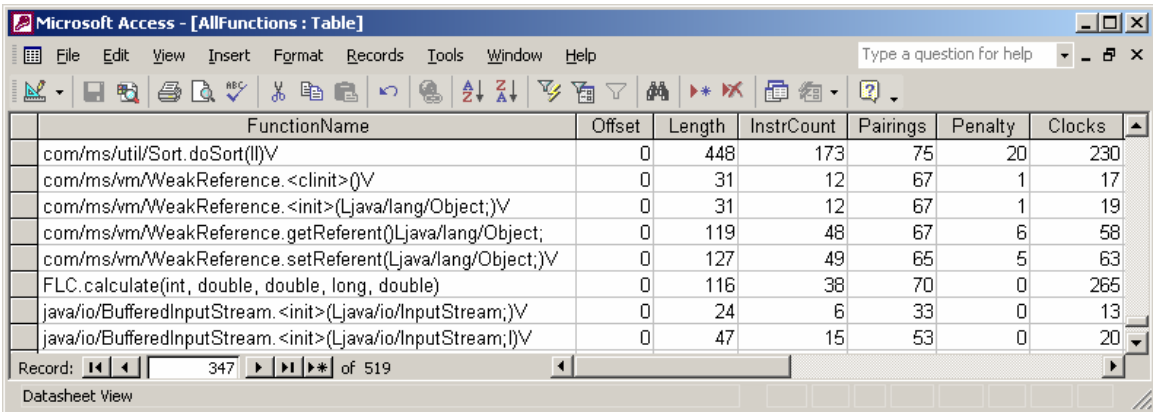


Figure 6.4.2.1 A-FLC execution time is 265 clock cycles for the Poisson distribution

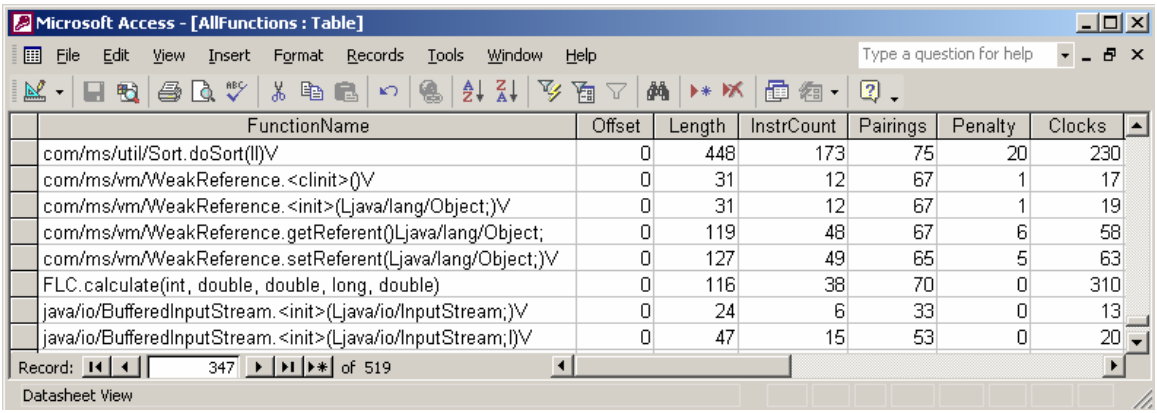


Figure 6.4.2.2 A-FLC execution time is 310 clock cycles for the heavy-tailed distribution

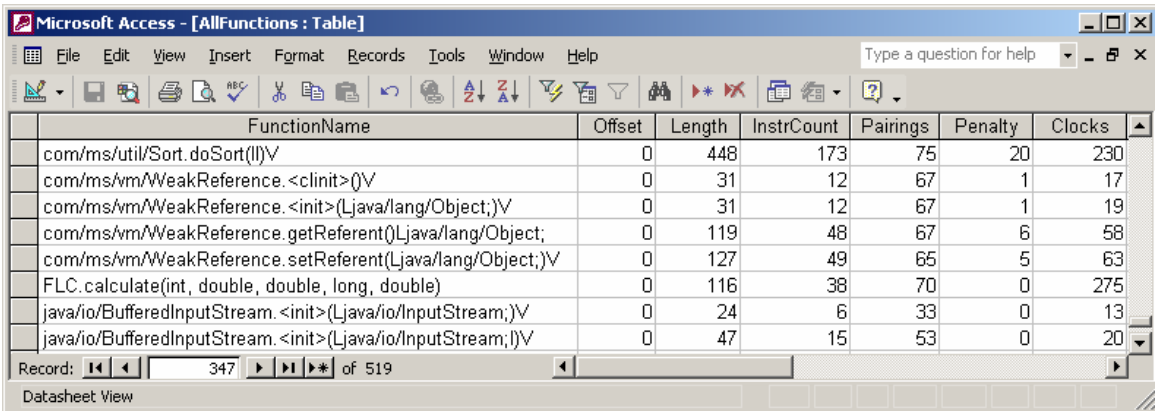


Figure 6.4.2.3 A-FLC execution time is 275 clock cycles for the trace [Trace]

6.4.3 R^2-FLC

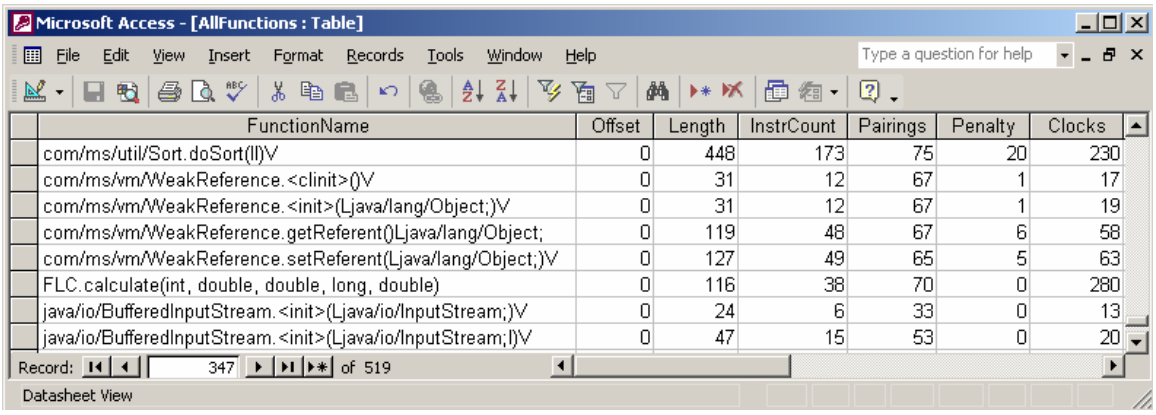


Figure 6.4.3.1 R^2-FLC execution time is 280 clock cycles for the Poisson distribution

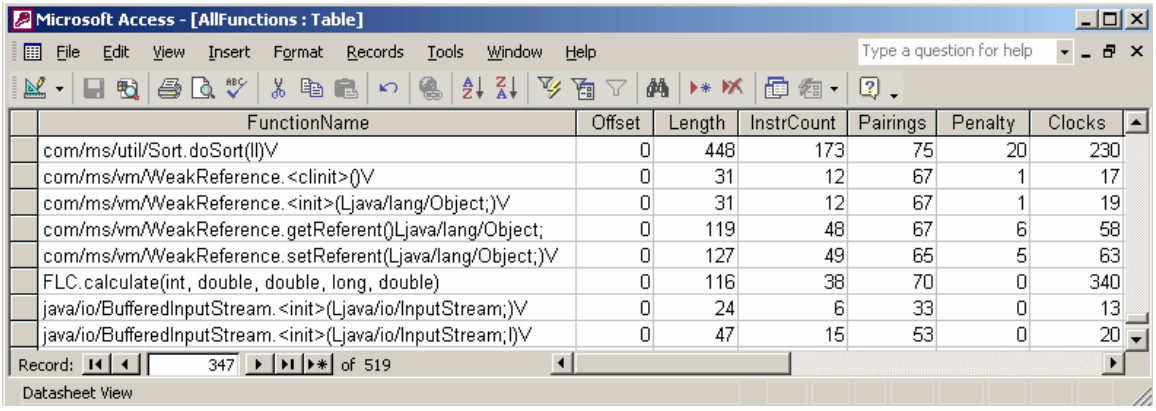


Figure 6.4.3.2 R^2-FLC execution time is 340 clock cycles for the heavy-tailed distribution

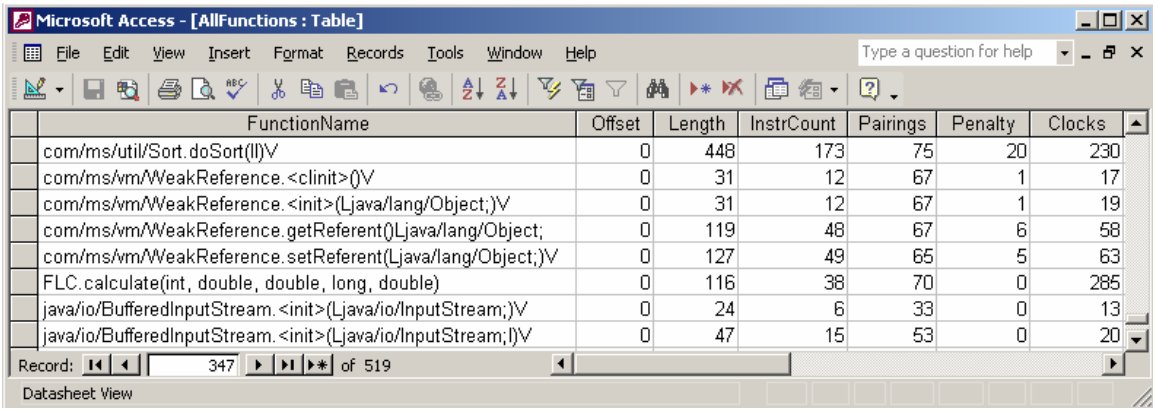


Figure 6.4.3.3 R^2-FLC execution time is 285 clock cycles for the trace [Trace]

6.4.4 SUMMARY OF THE EXPERIMENTAL RESULTS SHOWN ABOVE

	Measured average number of T cycles per control cycle		
Control models	Poisson distribution	Heavy-tailed distribution	Trace
FLC	250	275	255
A-FLC	265	310	275
R^2-FLC	280	340	285

Table 6.4.4.1 Summary of the experimental results shown above

In the experimental results including the ones tabulated above all three FLC versions require less than 350 clock cycles to execute; that is, their control cycle times are less than 350 clock cycles. The R^2-FLC control cycle time is always relatively the longest for all the distributions, namely, Poisson, heavy-tailed and self-similar. The FLC controller requires the lowest control cycle because it does not include the RTPD (*real-time traffic pattern detection*) in its control process and the threshold values used by the FLC controller do not change during execution. Similarly the A-FLC does not have the RTPD component to determine the type of the traffic either, and its "*don't care*" range threshold value changes during execution. The amount of change, which is administered once the threshold is exceeded, is fixed. The R^2-FLC has the longest control cycle because it needs the RTPD component to determine the type of the traffic so that it adjusts its threshold values accordingly. In fact, the RTPD exists as a software entity that runs in parallel with the R^2-FLC main body. Therefore it does not contribute to lengthen the execution time of the R^2-FLC main body directly, but the result of its traffic pattern detection may invoke extra R^2-

FLC computation to finely adjust the “don't’ care” range threshold value in a dynamic manner.

6.5 CONNECTIVE SUMMARY

To recap, this section presents what I have achieved in the deeper FLC research work:

a) It is experimentally confirmed that an optimal FLC design range does exist [p14].

b) It is confirmed that that the FLC can be made more adaptive by manipulating the *“don’t-care range-threshold”* in a dynamic manner. This is the basis for the new A-FLC (Adaptive/Reconfigurable FLC [p12]) concept.

c) It was discovered that the dynamic buffer tuning capability of the A-FLC can be improved if it is allowed to self-tune itself with respect to the current Internet traffic pattern. This is the conceptual framework for the R^2-FLC (Real-time Reconfigurable FLC) tuner, which is experimentally more efficacious than the FLC and the A-FLC.

CHAPTER 7 IN-DEPTH NNC RESEARCH

7.0 INTRODUCTION

The in-depth NNC research addresses the following major issues left behind by my previous MPhil thesis as follows:

a) The possibility of a correlation between the accuracy and the number of neurons in the hidden layer of the NNC.
b) The need for a timing analysis of the NNC.
c) The possibility of cutting down the NNC control cycle time and lowering the chance of deleterious effect.

Therefore, the objectives of the in-depth research include:

a) Define the correlation between the number of neurons in the NNC hidden layer and the control accuracy; this is carried out by the *sensitivity analysis.*
b) Propose a method(s) to optimize the NNC configuration to lower its control cycle time in an on-line manner.
c) Timing analyses of the optimized NNC model to confirm that it is indeed more suitable for time-critical applications over the Internet.
d) Study the impact of different traffic waveforms/distributions on the stability and accuracy on the NNC control process in different experiments.

7.1 SENSITIVITY ANALYSIS OF THE HIDDEN LAYER

The NNC model, which works by *backpropagation* and supervised training, is shown in Figure 7.1.1. The NNC operates in two distinctive phases, namely, *training/learning,* and *dynamic buffer tuning*. In action it is a twin system consisting of the "*Chief*" NNC module and the "*Learner*" NNC module as shown in Figure 7.1.2. The *Chief*, which has already learnt previous patterns, carries out actual dynamic buffer tuning while the *Learner* undergoes training to acquire new knowledge to deal with new phenomena. Before training starts all the weights of the arcs in the *Learner's* neural network are randomized. As training progresses the error (difference) between the "trainee" output and the NNC desired/deserved output Δ decays gradually. After training the *Chief* and the *Learner* swap positions. The NNC stability is analyzed by measuring the mean deviation (MD) from the chosen QOB_R reference in terms of "*the number of neurons in the NNC hidden layer versus different traffic patterns*".

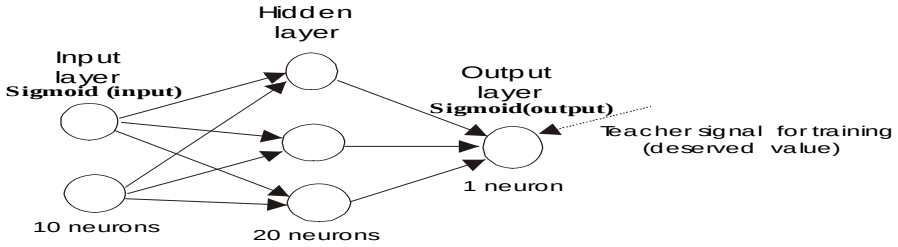


Figure 7.1.1 A backpropagation model

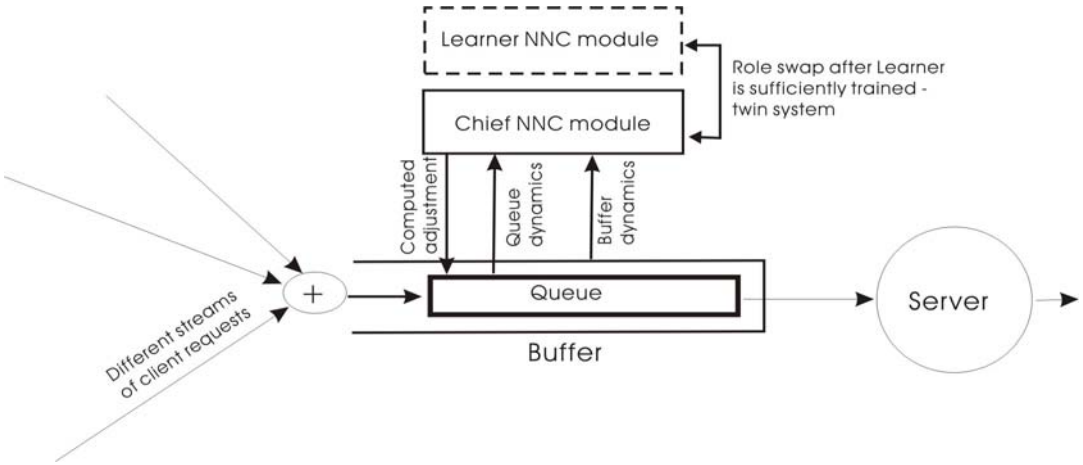


Figure 7.1.2 The NNC – a twin system of two NNC clones

7.1.1 EXPERIMENTAL RESULTS

The NNC model was verified by simulations over the Aglets, which is a mobile agent platform specifically designed for Internet applications [Mitsuru1998]. The Aglets is chosen for three reasons: a) it is stable, b) it has rich user experience, and c) it makes the verification results scalable for the open Internet. The set up for the verification simulations is shown in Figure 7.1.1.1, where the driver and the server are aglets (*agile applets*) collaborating within a single computer. The driver picks a waveform (e.g. Poisson) or trace from the table and uses it to generate the inter-arrival times for the simulated merged traffic for the server queue. A trace contains the RTT data pre-collected from a TCP channel, and it usually embeds an unknown traffic pattern. The aim of using data traces in simulations is to verify that the NNC control precision and stability are indeed traffic independent.

The waveform picked by the driver was first checked for its LRD (long-range dependence) or SRD (short-range dependence) behavior. The checking process is

indicated by the box deisgnated "*traffic pattern analysis*". Different tools were used to identify the waveform's exact nature once its LRD/SRD character had been determined. For example, the R/S (*rescaled adjusted statistics*) estimator in the *Selfis Tool* [Karagiannis2003] was used to compute the *Hurst* (H) parameter/value for different traces. The character is identified as follows: $0.5 < H \leq 1$ for LRD and $0 < H \leq 0.5$ for SRD. Other tools were then employed to identify the exact waveform/distribution, for example, the *modified QQ-plot* for heavy-tailed identifications.

In this section two sets of experimental results among the many collected for analytical purposes are presented for demonstration. The first set, "Case 1", demonstrates how the NNC behaves with random (i.e. SRD) traffic. The second set, namely, "Case 2", demonstrates that the NNC stability is independent of the self-similar nature of the traffic (i.e. LRD). The plots are obtained with the help of the *Selfis* tool.

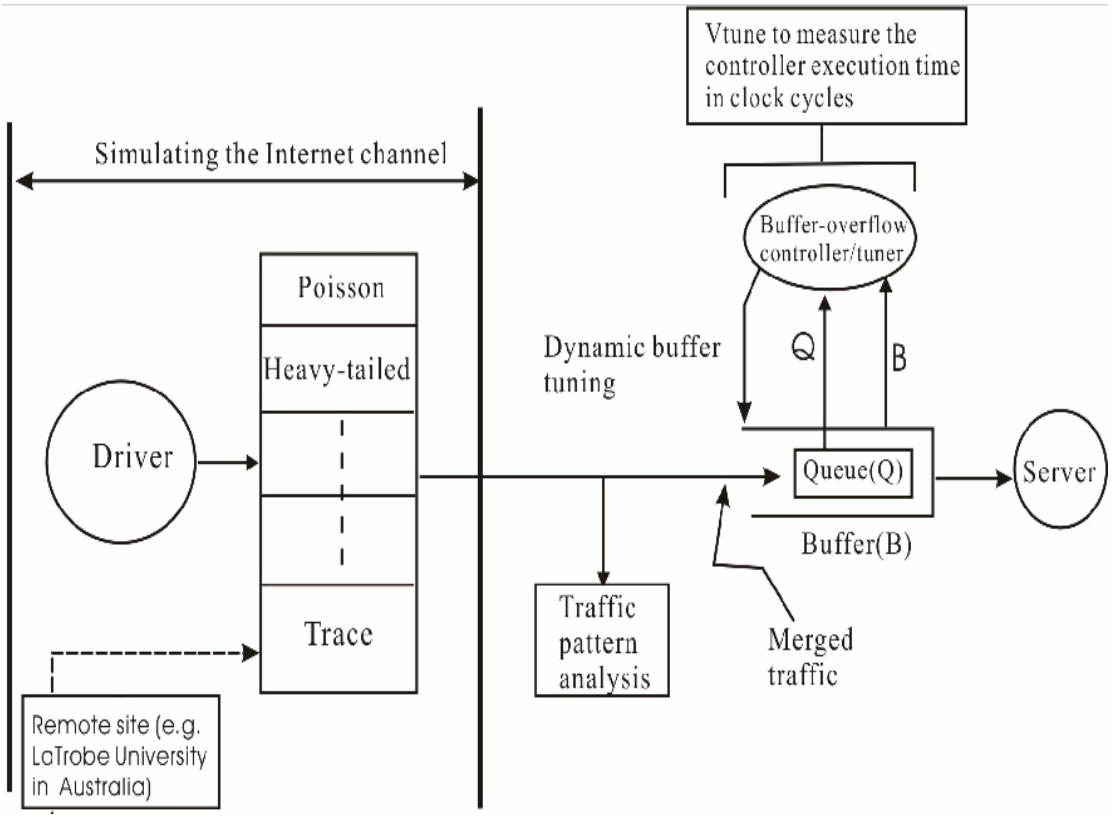


Figure 7.1.1.1 The NNC verification environment

Case 1 - Random Traffic

For the random RTT trace for demonstration here the *Selfis's* R/S plot yields H=0.483 and 99.66% confidence of its SRD character (Figure 7.1.1.2). Figure 7.1.1.3 shows that both NNC and PIDC produce no overflow for the trace, but the former eliminates the shortcomings of the latter. The exponential/random nature of the trace is also confirmed by comparing its mean (m) and standard deviation (δ), which are 100 ms and 101 ms respectively. The "$100 \approx 101$" (i.e. $m \approx \delta$) condition indicates that the traffic comes from a Poisson process, which is SRD by nature.

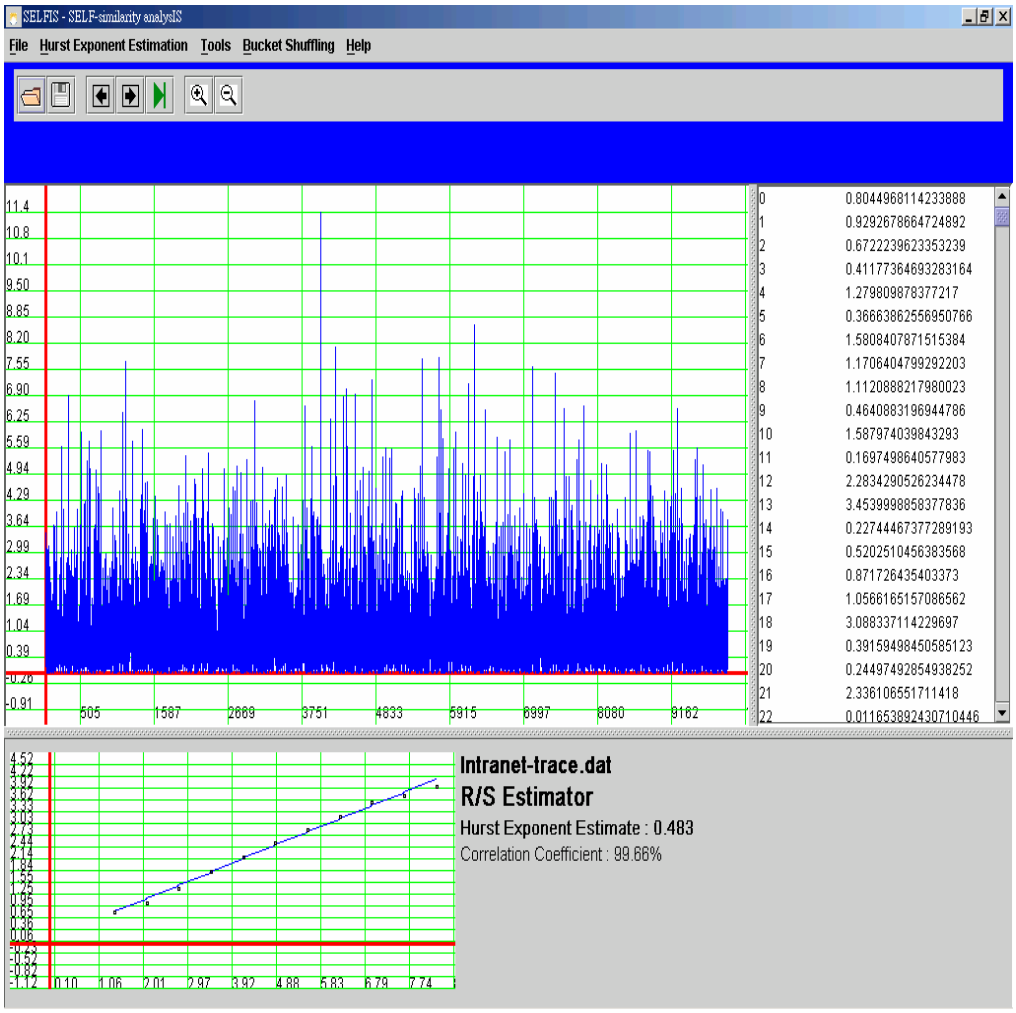


Figure 7.1.1.2 SRD character confirmed by R/S estimator of *Selfis*

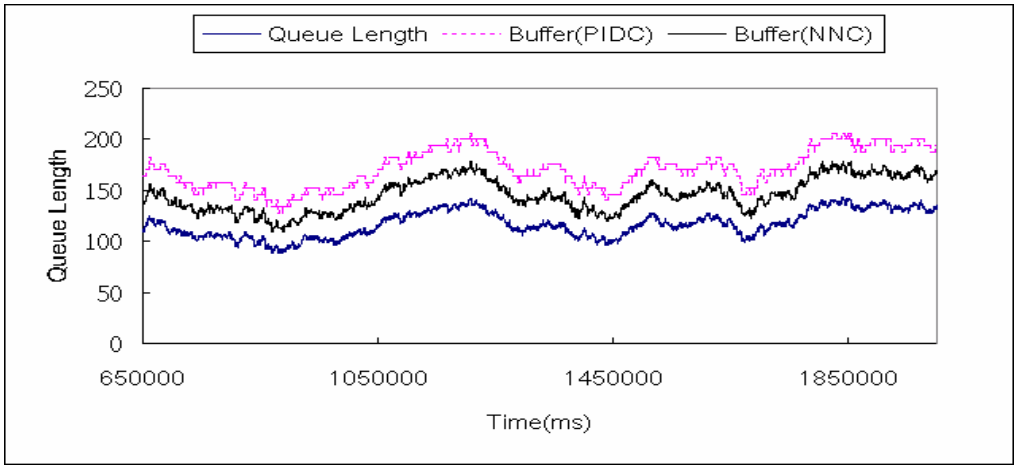


Figure 7.1.1.3 Experimental results for the Intranet Traffic

Case 2 – Self-similar Traffic

Self-similar traffic [Tsybakov1998] contains bursts that easily inundate the server queue buffer. It is important therefore for the NNC to have the capability to tune the buffer responsively at runtime to ensure that it always covers the queue length. Different experimental results verified that the NNC indeed has this capability. The self-similar traffic patterns were generated by the tool proposed by G. Kramer [Kramer]. For example, the trace for Figure 7.1.1.4 is generated by this tool, and for it the R/S plot yields H=0.615, with 98.67% confidence of its LRD character. Both PIDC and NNC (no CA support for this case) produce no overflow for different self-similar traffic patterns, as shown by Figure 7.1.1.5. The NNC maintains the safety margin Δ of the $\{0,\Delta\}^2$ objective function consistently minus the PIDC shortcomings.

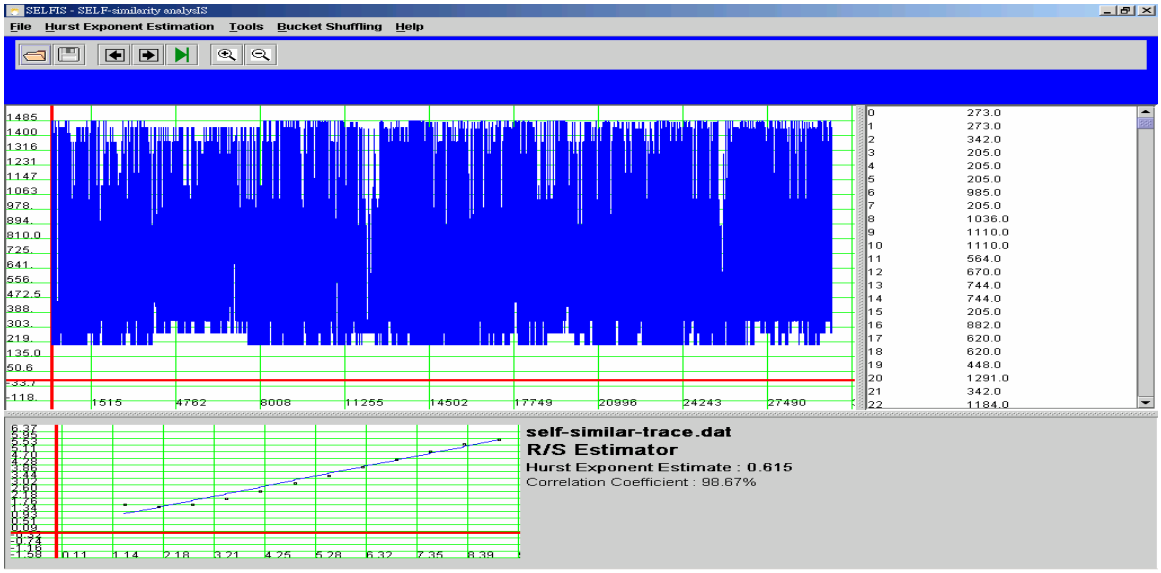


Figure 7.1.1.4 LRD confirmed by the R/S estimator in *Selfis*

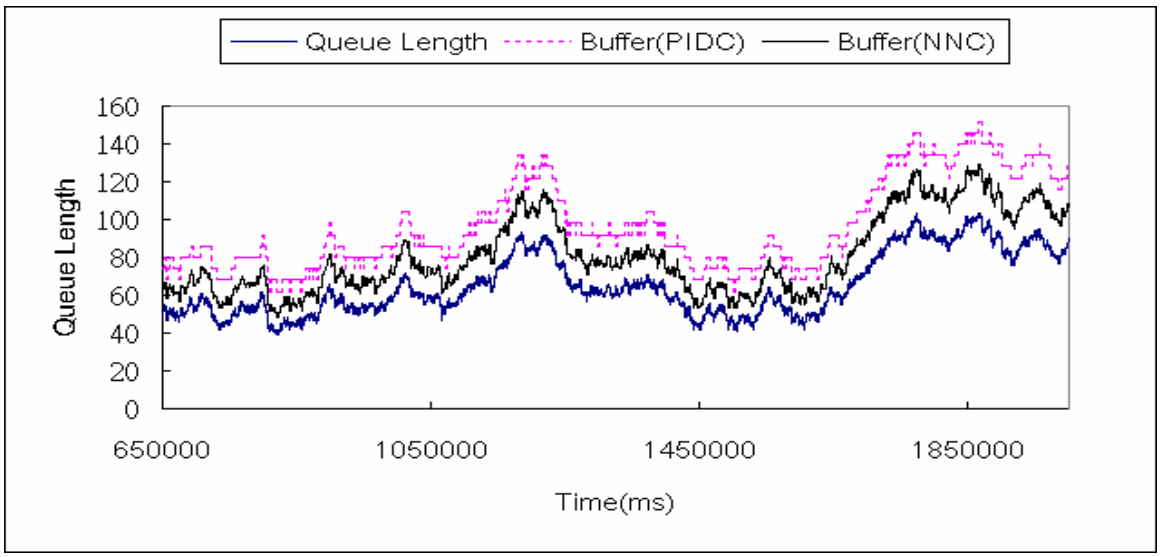


Figure 7.1.1.5 NNC and PIDC performances for the self-similar trace confirmed in Figure 7.1.1.4

The NNC stability is analyzed by measuring the mean deviation (MD) from the chosen QOB_R reference in terms of "*the number of neurons in the NNC hidden layer versus different traffic patterns*". The preliminary empirical results shown in Figure 7.1.1.6 indicate that having 20 neurons in the NNC hidden layer is more or less the break point. Using more neurons does not produce better performance by yielding a lower MD. For the Poisson trace, the mean deviation error settles down for 15 hidden neurons in the hidden layer but for other traffic patterns at least 20 neurons are needed. All the experimental results from this stage indicate that it is safer to use 20 neurons for the hidden layer for Internet applications because its traffic pattern, which includes all the patterns in Figure 7.1.1.6, switches quickly without warning.

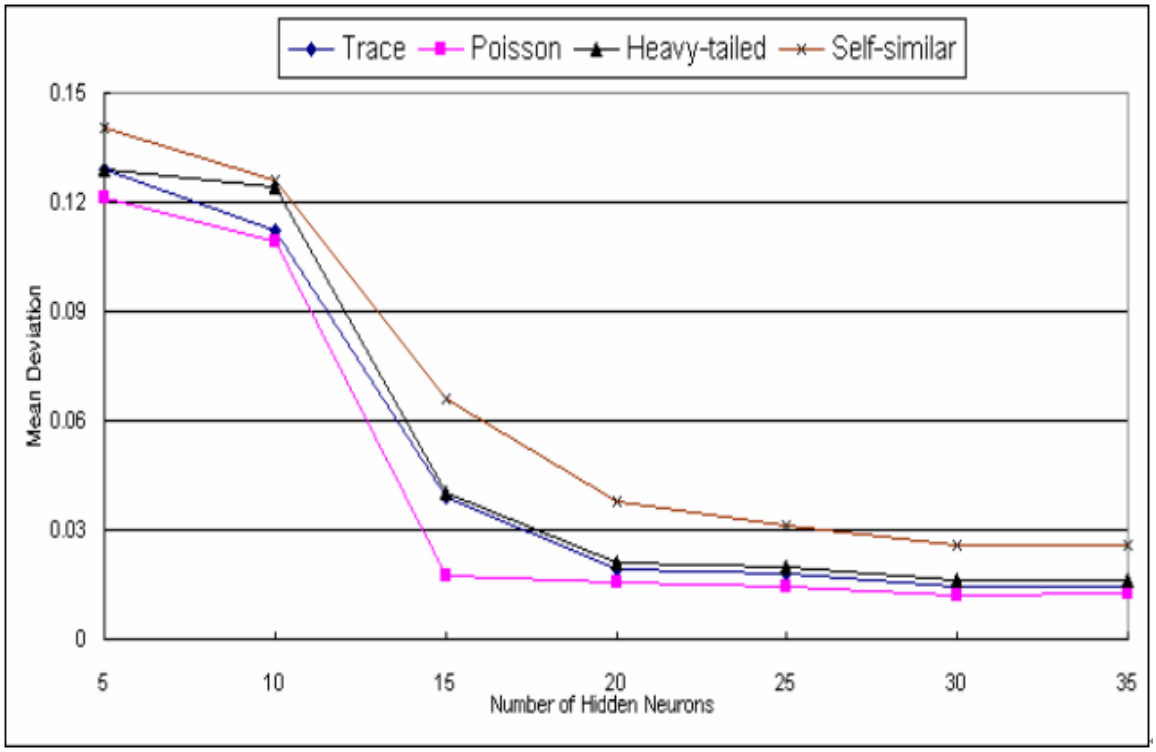


Figure 7.1.1.6 Mean deviation error for using different numbers of neurons in the NNC hidden layer versus different possible Internet traffic patterns

7.2 REAL-TIME NNC PRUNING

The aim is to optimize the NNC configuration in an on-line manner to adaptively lower its control cycle time. After a thorough literature search it was found that the existing pruning techniques are for off-line application. In the off-line process, the neural network (NN) is first run to obtain some data for analysis and then optimized manually before it is run for the next round. This is basically a trial and error process [Gallant1992, Hagan1996]. The off-line approach is not suitable for real-time application because the NN should be able to adapt its configuration on the fly to suit the current operational conditions. With the on-line and timeliness

requirements in mind the *Hessian-based pruning* (HBP) technique is proposed. The HBP optimizes the NNC configuration at run-time in an adaptive, dynamic and cyclical manner. The "NNC plus HBP" combination is the new O-NNC (Optimized NNC) controller. In action the HBP is a renewal process, and the optimisation in every renewal cycle has two phases of operations, as shown in Figure 7.2.1:

a) *First phase*: The *Learner* computes the weights of all the arcs in its neural network. After that all the insignificant arcs are marked by the principle of *dynamic sensitivity analysis*.

b) *Second phase*: After the *Learner* becomes the *Chief* all the marked arcs are virtually pruned (excluded) from its computation to shorten the control cycle time. Virtual pruning means that the physical skeletal NNC configuration is intact and provides the *bare basis* for every pruning operation.

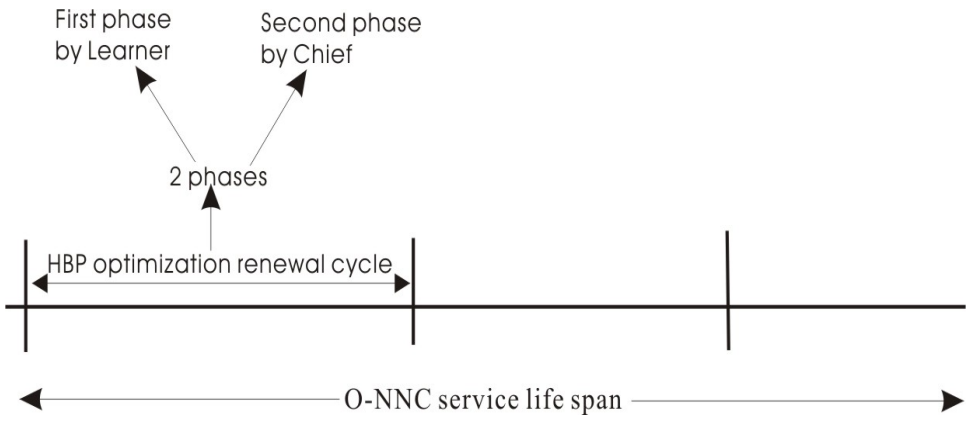


Figure 7.2.1 The HBP is as a renewal process

The choice of HBP over other techniques is dictated by the fact that the NNC optimisation process is real-time and simplicity is the key to success. Other techniques from the literature normally require complex mathematical manipulations. Besides, the published experience for the feed-forward neural network pruning is exclusively off-line. This makes them unsuitable for the on-line NNC application. The HBP operation is based on *dynamic sensitivity analysis*. The rationale is to mark and skip a neural network connection if the error/tolerance of the neural computation is insensitive to its presence. For the NNC the error/tolerance is the $\pm\Delta$ band about the QOB_R reference. The core of the HBP technique is this concept: "*if a neural network converges toward a target function so will its derivatives* [Gallant1992]". In fact, the main difference among all the identified *performance-learning* laws from the literature [Hagan1996] is how they leverage the different parameters (e.g. weights and biases).

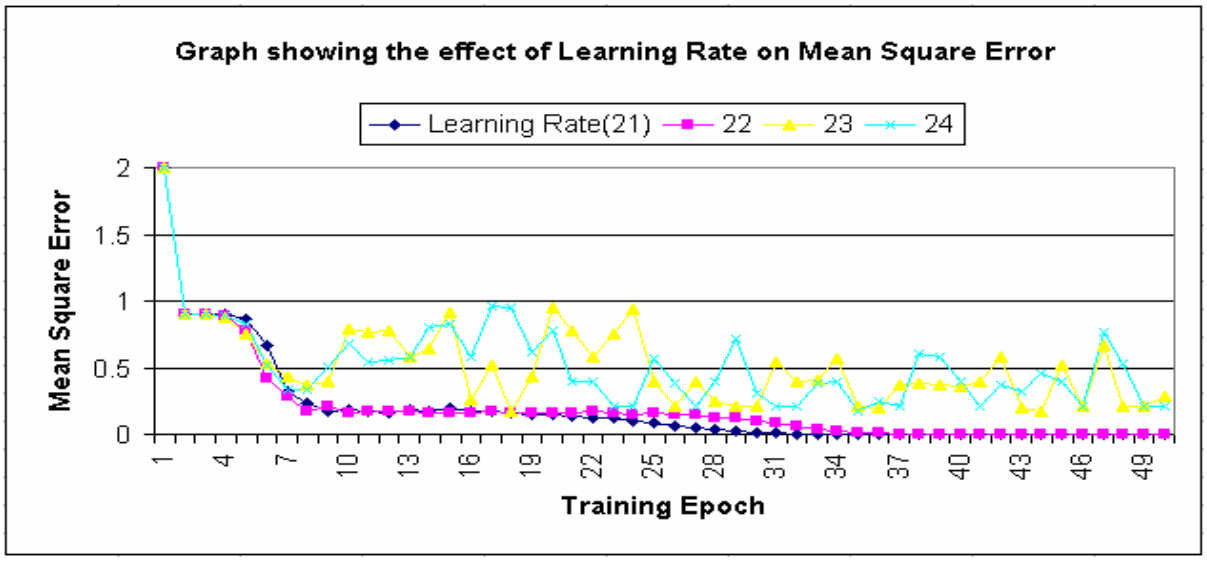


Figure 7.2.2 The graph showing the effect of learning rate on mean square error

The HBP adopts the Taylor Series [Finney1994] (equation (7.1)) as the vehicle to differentiate the relative importance of the different neural network (NN) parameters. The meanings of the parameters in equation (7.1) are: F() - the function, *w* - the NN connection weight, Δw – the change in *w*, $\nabla F(w)$ - the gradient matrix (7.2), and ∇^2F(w) - the *Hessian* matrix (7.3). The symbols in the equations mean the following: *T* for transpose, *O* for higher order term, *n* for the n^{th} term, and $\partial / \partial w_1$ for partial differentiation. Thus the expansion about w of $F(w+\Delta w)$ is given by equation (7.1).

$$F(w+\Delta w) = F(w) + \nabla F(w)^T \Delta w + \frac{1}{2} \Delta w^T \nabla^2 F(w) \Delta w + O(||\Delta w||^3) + \ldots\ldots\ldots (7.1)$$

$$\nabla F(w): \left[\frac{\partial}{\partial w_1} F(w) \qquad \frac{\partial}{\partial w_2} F(w) \ldots \frac{\partial}{\partial w_n} F(w) \right]^T \ldots\ldots\ldots (7.2)$$

$$\nabla^2 F(w): \begin{bmatrix} \frac{\partial^2}{\partial w_1^2} F(w) & \frac{\partial^2}{\partial w_1 w_2} F(w) & \ldots & \frac{\partial^2}{\partial w_1 \partial w_n} F(w) \\ \frac{\partial^2}{\partial w_2 \partial w_1} F(w) & \frac{\partial^2}{\partial w_2^2} F(w) & \ldots & \frac{\partial^2}{\partial w_2 \partial w_n} F(w) \\ \ldots & \ldots & \ldots & \ldots \\ \frac{\partial^2}{\partial w_n \partial w_1} F(w) & \frac{\partial^2}{\partial w_n \partial w_2} F(w) & \ldots & \frac{\partial^2}{\partial w_n^2} F(w) \end{bmatrix} \ldots\ldots\ldots (7.3)$$

The preliminary O-NNC results confirm that the HBP performs as expected. The findings from the preliminary HBP experiments concur with similar experience published previously [Oh1998]. That is, the weighing factors (*synaptic weights* or *learning rates*) affect the convergence speed. Many different experiments were

carried out to study the effect of different learning rates, and one set of results is presented in Figure 7.2.2 for demonstration purposes. It shows how the correlation between the learning rate and the *mean square error* (MSR) varies. A learning rate is the magnitude of change when a connection weight is adjusted in training. For example, the desired output is $w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha \partial F(x) / \partial w_{i,j}^{m}$, with $w_{i,j}^{m}(k)$ as current weight and α as the learning rate. The MSR, defined as, $MSR = \boldsymbol{E}[(target\ output - actual\ output)^2]$, measures the control accuracy, with $\boldsymbol{E}$ as the averaging operator. The MSR should decrease when the convergence gets closer to the QOB_R reference. The experimental results, however, indicate that bigger learning rates may yield oscillatory convergence, as shown by the rates 23 and 24 in Figure 7.2.2. In contrast, the smaller rates 21 and 22 produce much smoother control. Under the equation (7.1), the learning/training process should converge to the QOB_R reference, which is mathematically known as the *target global minimum surface*. The convergence makes the gradient vector $\nabla F(w)$ insignificant and eliminates the "$\nabla F(w)^T \Delta w$" term from equation (7.1). This implies not only that the larger ordinal terms in equation (7.1) can be ignored but also a simplified form (equation (7.4)) is possible for the equation. Further simplification of equation (7.4), based on: $\Delta F = F(w+\Delta w) - F(w)$, yields equation (7.5).

$$F(w+\Delta w) = F(w) + \frac{1}{2} \Delta w^T \nabla^2 F(w) \Delta w \ldots (7.4)$$

$$\Delta F = \frac{1}{2} \Delta w^T \nabla^2 F(w) \Delta w \ldots (7.5)$$

The HBP optimization cycle has two phases. The first phase is applied only to the *Learner* and the second to the current *Chief* role. The details involved are as follows (first three points belong to the first phase and the fourth point to the second phase):

a) Use Taylor series (equation (7.1)) to identify the significant neural network parameters.

b) Choose appropriate learning rates for the significant parameters to avoid convergence oscillations, as illustrated in Figure 7.2.2.

c) Mark the synaptic weights that have insignificant impact on the Taylor series.

d) After the *Learner* has become the *Chief*, it excludes all the marked connections in its neural computation. The exclusion, represented by equation (7.6), is, in effect, virtual pruning of the insignificant connections. It is a logical, virtual process because the skeletal NNC neural network configuration remains intact except for excluding the marked connections in the subsequent O-NNC control. The pruning decision is based on the *Lagrangian* index S (to be explained later).

Since the optimisation starts anew every time the *Learner* has completed training, which means new weights for the neural network connections, the optimized outcome should be unique, and this makes the HBP optimisation process dynamic and adaptive.

$w_i + \Delta_{wi} = 0 \ldots (7.6)$

$$S = \frac{1}{2} \Delta \boldsymbol{w}^T \nabla^2 F(w) \Delta w - \lambda (\boldsymbol{U}_i^T \Delta w + \boldsymbol{w}_i) \ldots (7.7)$$

If *Δw* in equation (7.5) is replaced by equation (7.6), then the Lagrangian equation (7.7) is formed. Now, equation (7.1) has become a typical *constrained optimization* problem [Bertsekas1982]. The symbols: U_i^T and λ in equation (7.8) are the unit vector and the Lagrange multiplier respectively. The optimum change in the weight vector w_i (equation (7.6)) is shown in equation (7.8). Every entry in w_i associates with a unique Lagrangian index S_i (equation (7.9)). In the first phase of the HBP optimisation process the S_i values are sorted so that the corresponding less significant w_i (neural network connection) can be excluded from the *Chief's* neural computation, starting from the lowest S_i. The pruning stops if the exclusion of the current S_i affects the accuracy of convergence process. Only after the virtual pruning process has been completed does the *Learner* become the *Chief*.

$$\Delta_{wi} = -\frac{w_i}{[\nabla^2 F(w)^{-1}]_{i,i}} \nabla^2 F(w)^{-1} U_i \quad \ldots(7.8)$$

$$S_i = \frac{w_i^2}{2[\nabla^2 F(w)^{-1}]_{i,i}} \quad \ldots(7.9)$$

7.2.1 EXPERMENETAL RESULTS

Different experiments with different waveforms (e.g. SRD and LRD) were conducted to verify the efficacy and correctness of the HBP technique and the O-NNC. The set up for the experiments is the same as Figure 7.1.1.1. The preliminary results confirm that HBP technique shortens the O-NNC control cycle time

consistently. The skeletal configuration of the O-NNC in the experiments is the same as the NNC prototype with 10 input neurons, 20 neurons for the hidden layer, and one output neuron. This configuration is fully connected, with 200 connections between the input layer and the hidden layer, as well as 20 connections between the hidden layer and the output layer. The O-NNC result in Figure 7.2.1.1 is produced by a configuration that has a hidden layer of 187 arcs instead of the 220 full connections because 33 of them are pruned by the dynamic HBP. Different experimental results indicate that the O-NNC has the capability to yield the same level of buffer overflow elimination efficacy as the un-optimized NNC, but with shorter convergence time to reach QOB_R. Figure 7.2.1.2 shows that O-NNC always ensures that the QOB value is within the tolerance band of $|2\Delta|$ (QOB_R=0.8). It compares the QOB deviation profiles of the three controllers. As illustrated by Table 7.2.1.1, the O-NNC, however, has a larger mean deviation (MD) than the un-optimized NNC, $MD = \left[\sum_{i=1}^{k} |\Delta - QOB_i|\right] \Big/ k$.

The PID controller (PIDC) is algorithmic in nature, and it is therefore also referred to as the Algorithmic PID controller [Ip2001]. Therefore PIDC and A-PID are synonymous in my research. The PIDC makes use of the Convergence Algorithm (CA), which is implemented at the *micro level*. At this level the CA exists as an independent logical object that runs in parallel with the PIDC main body. In this form the CA is called the M^3RT entity that can be invoked for service anytime and anywhere by message passing.

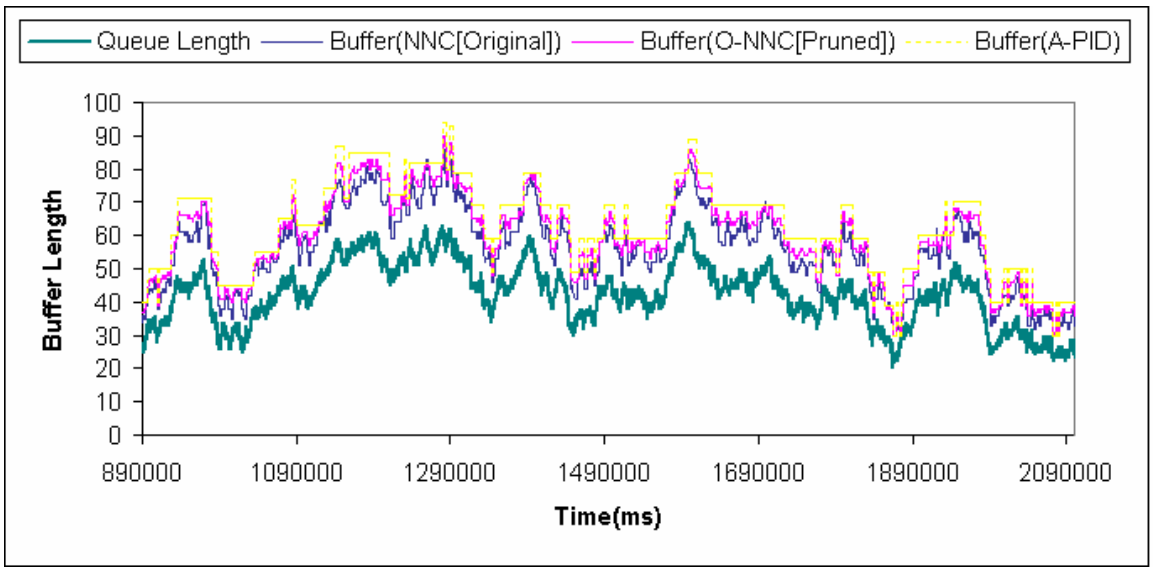


Figure 7.2.1.1 A set of experimental results to compare NNC, O-NNC and A-PID

Controller/tuner	Mean Deviation
NNC (Original)	0.0536
O-NNC (Pruned)	0.0916
A-PID	0.1279

Table 7.2.1.1 Mean deviations for Figure 7.2.1.2

Controller/tuner	The measured average number of clock cycles per tuner control cycle
NNC (Original and un-optimized)]	10800
O-NNC (Pruned/optimized)	9250 ($9250/10800 \approx 0.857$; 85.7%)

Table 7.2.1.2 Comparing the average number of clock cycles per tuner cycle

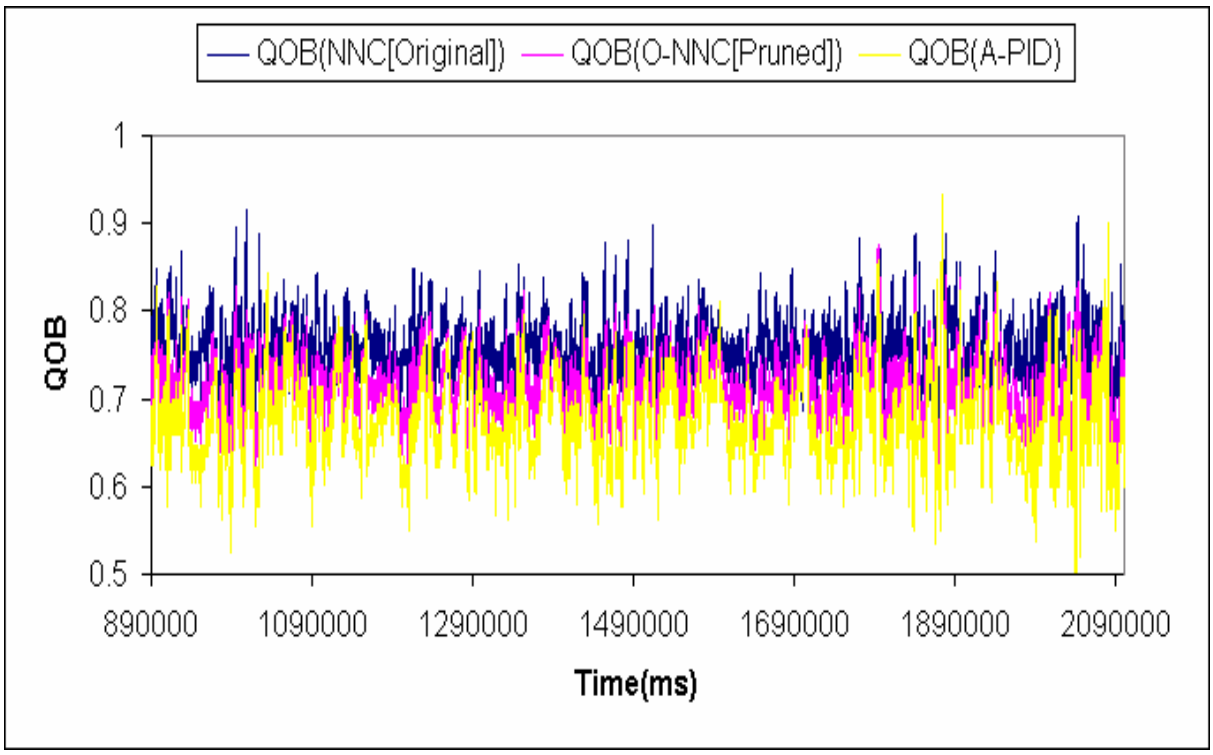


Figure 7.2.1.2. Indication of the HBP convergence stability

The average control cycle time or CCT for the O-NNC is only 9250 clock pulses compared to the 10800 for the NNC (Table 4). The CCT in clock pulses are measured with the *Intel's VTune Performance Analyzer* [VTune2002], and they can be converted easily into the *physical control cycle time* for any platform by $P-CCT = CCT * {}^{1}\!/_{Hz}$, where *Hz* is the platform's operating speed in hertz. Figure 7.2.1.2 also compares the three controllers O-NNC, NNC and A-PID in terms of the convergence smoothness. Figure 7.2.1.3 provides more convergence stability details for Figure 7.2.1.2 in term of the individual deviations over time from the QOB_R reference of the $\{0,\Delta\}^2$. The performance of the NNC (Original) and the O-NNC

(Pruned) is better than A-PID with respect to the deviation error. Figure 7.2.1.4 is another comparison of the three controllers. Figure 7.2.1.4 compares their efficacy in the dynamic buffer adjustment/tuning process. Figure 7.2.1.5 compares the QOB profiles of the three controllers, and Figure 7.2.1.6 to Figure 7.2.1.8 show the deviations of the individual controllers. From the many different experimental results we conclude that both the NNC and the O-NNC performs as well as the A-PID but without its shortcomings. Despite its consistency in converging accurately to the QOB_R reference of the $\{0,\Delta\}^2$, the O-NNC dynamic tuning process is more oscillatory than the un-optimized NNC. The oscillation is an undesirable side effect from the dynamic HBP optimization cycles. In the future work this problem will be studied in detail so that the oscillation can be smoothened.

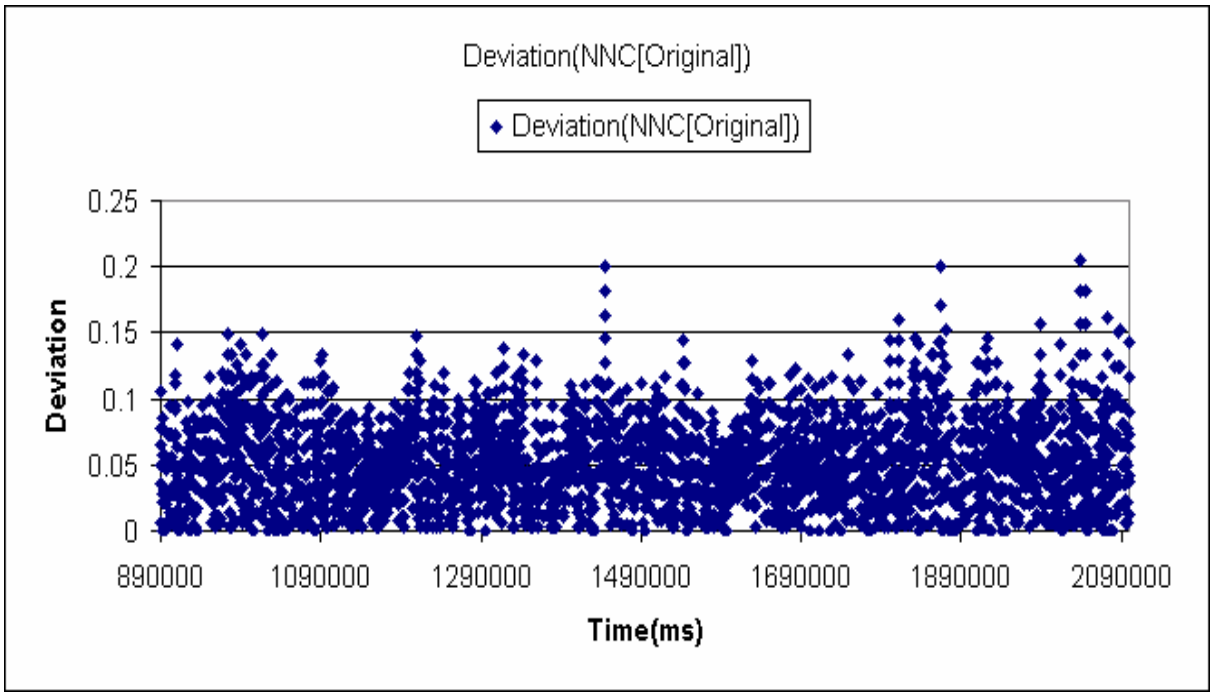


Figure 7.2.1.3a. Deviation profile of the original NNC

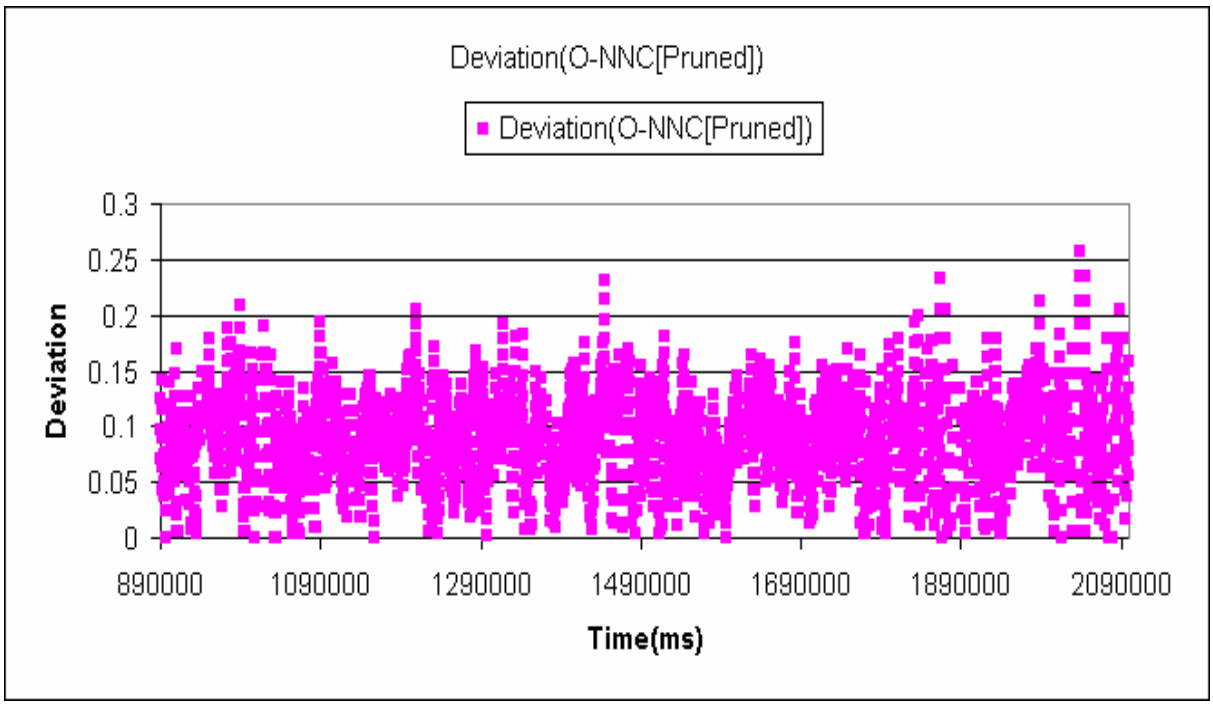


Figure 7.2.1.3b. Deviation profile of the O-NNC

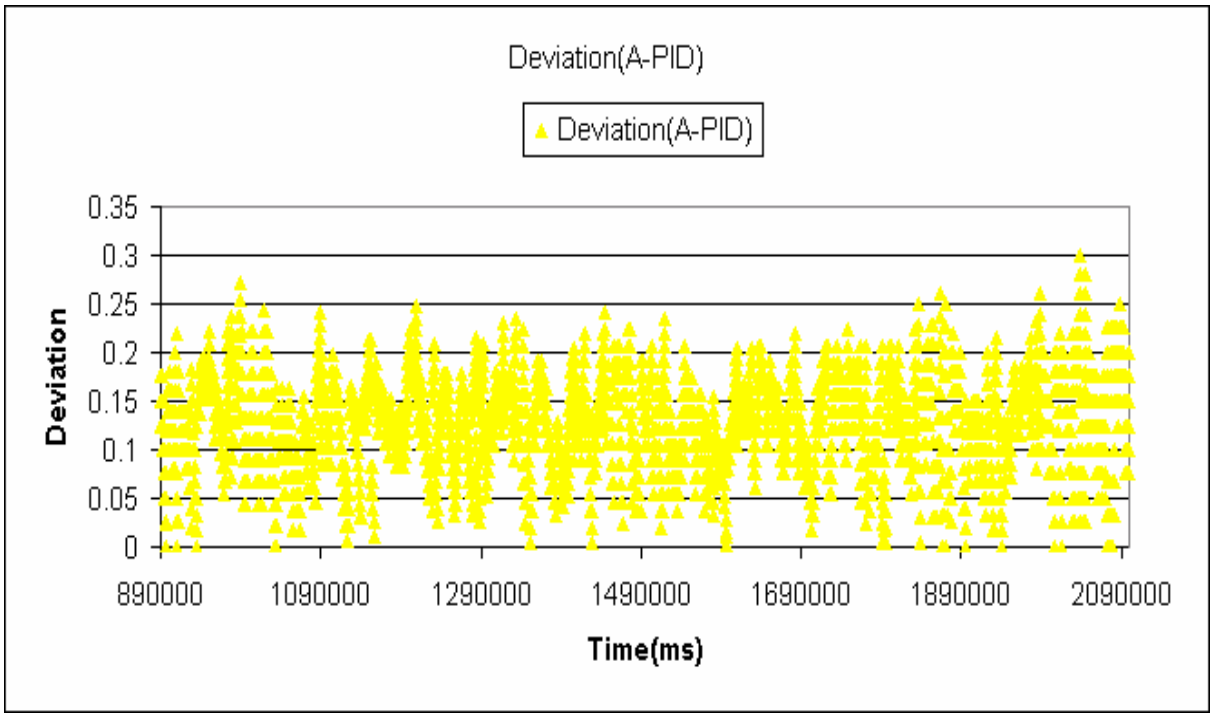


Figure 7.2.1.3c. Deviation by the A-PID controller

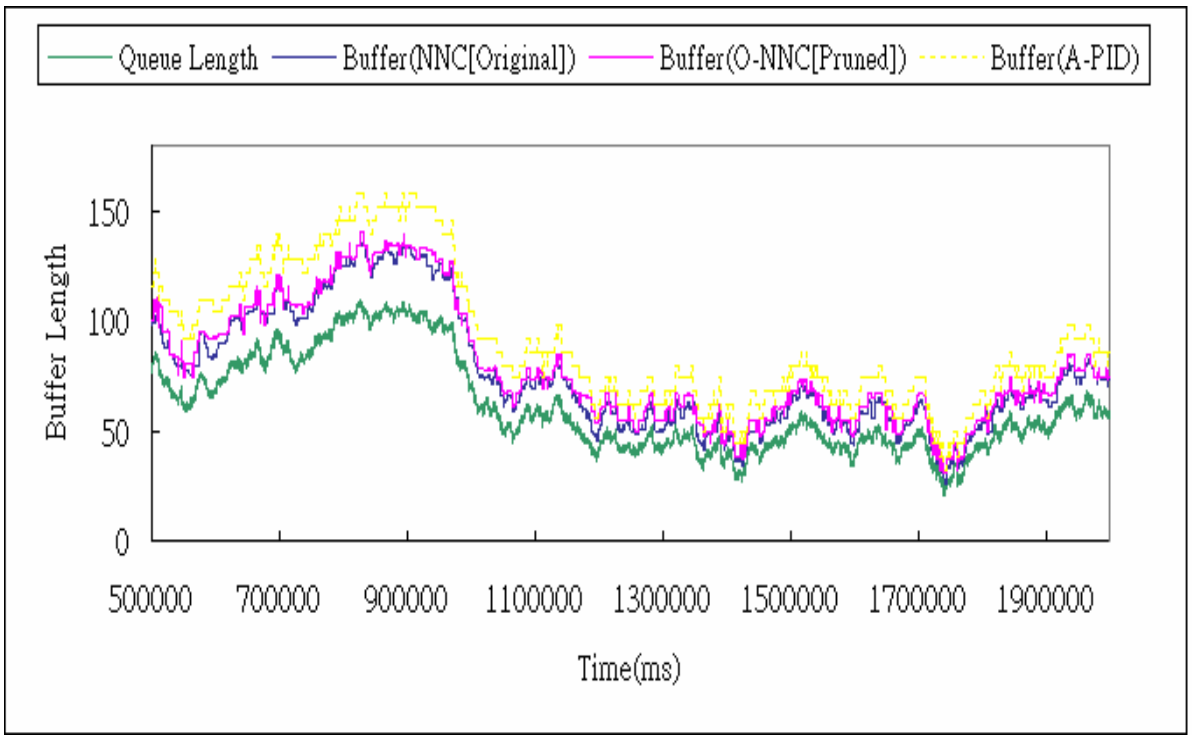


Figure 7.2.1.4. Another comparison of three controllers

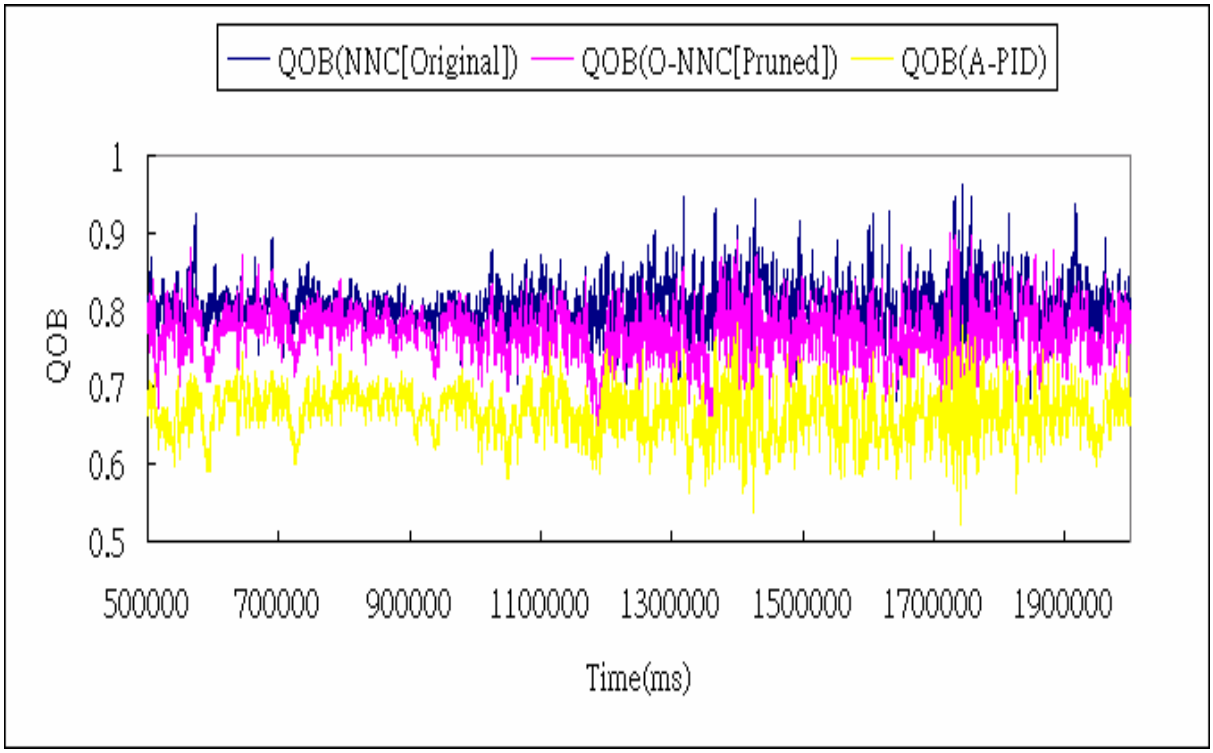


Figure 7.2.1.5. The QOB profiles of the three controllers in Figure 7.2.1.9a

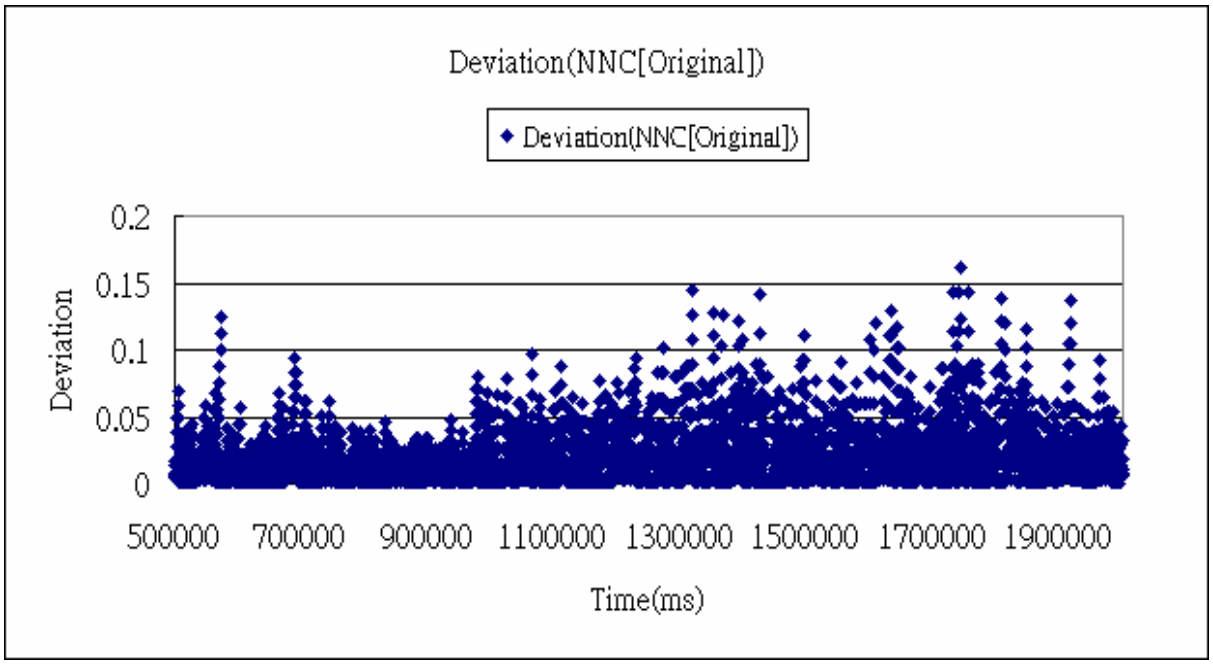


Figure 7.2.1.6. The deviation profile by the original NNC

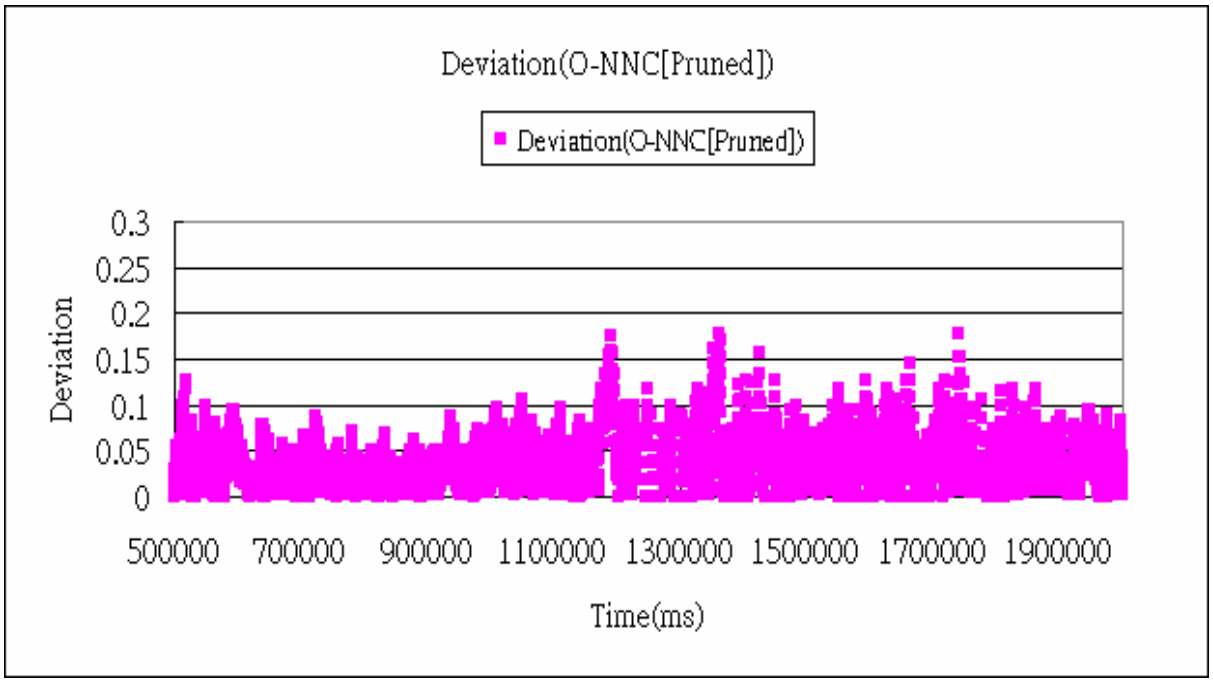


Figure 7.2.1.7. The deviation profile by the O-NNC

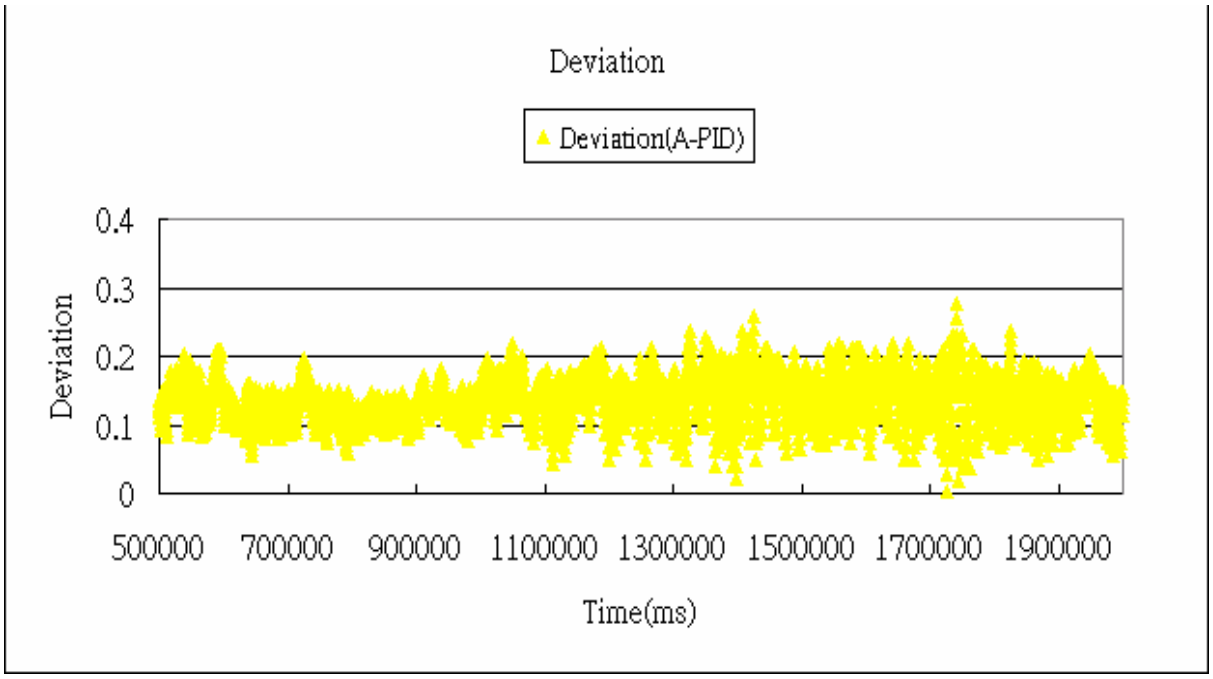


Figure 7.2.1.8. The deviation profile by the A-PID

7.3 CONNECTIVE SUMMARY

The in-depth NNC research has achieved the following:

a) It was confirmed empirically that there is indeed a correlation between the number of neurons in the NNC hidden layer and the control accuracy. The *sensitivity analysis* shows that mean deviation error depends on the number of neurons in the hidden layer as well as the traffic pattern (Figure 7.1.1.6).

b) The HBP technique is proposed to let the NNC self-optimize itself on the fly so that its control cycle time can be consistently reduced. The experimental results confirm that this technique cuts the NNC control cycle time by more than 10%. This makes the optimized NNC or O-NNC is more suitable for time-critical applications over the Internet.

c) In all the experiments different waveforms (e.g. SRD and LRD) were used. The experimental results confirm that the control accuracy and stability of NNC and O-NNC models are independent of the traffic patterns.

CHAPTER 8 LOCATION-AWARE TEST-BED

8.0 INTRODUCTION

The original forms of the following intelligent dynamic buffer size tuners were proposed and verified in my previous MPhil research in the Aglets environment: GAC, FLC, and NNC. In this thesis, we developed improvements, particularly the use of a real time traffic detector, which is employed in conjunction with FLC. As a result they improve the fault tolerance and shorten the service roundtrip time (RTT) of a client/server interaction. The timing analyses by Intel's VTune Performance Analyzer [VTune2002] indicate these novel intelligent tuners and their PIDC predecessor are all suitable for time-critical applications because of their short execution times (Table 8.0.1). The results in the table are based on repeated VTune measurements with the corresponding Java tuner prototypes. Although the three original intelligent dynamic buffer tuners eliminate the two shortcomings of their PIDC predecessor, the FLC is by far the most stable, simplest and fastest. The FLC needs only 255 clock cycles to execute and does not produce any overflow at all.

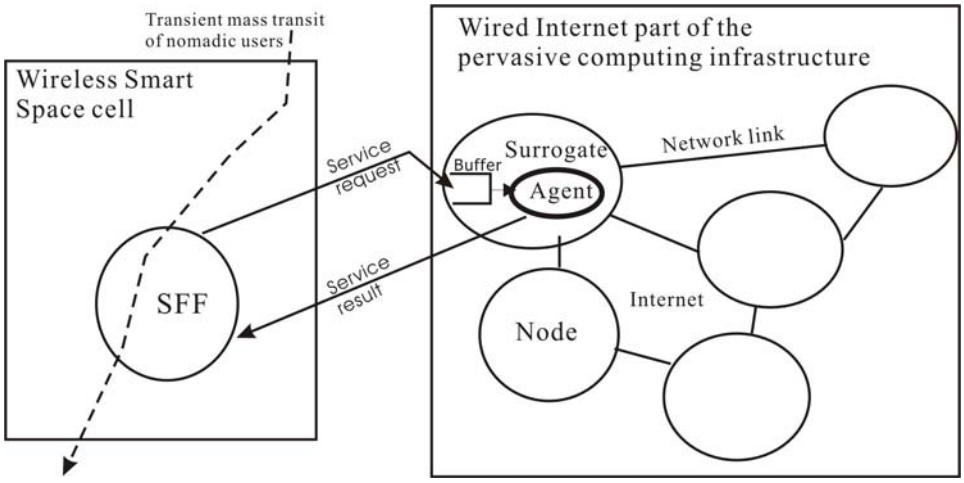


Figure 8.0.1 A pervasive computing environment

	Number of Java lines for implementation	Average number of clock/T cycles per control pass measured by using Intel's VTune Performance Analyzer [VTune2002]
Basic PID controller (or PIDC)	105	205
Fuzzy Logic Controller [FLC(6x4)]	116	255
Neural Network Controller (NNC) with MCA/ M^3RT support	240	10800
Genetic Algorithm Controller (GAC)	111	475

Table 8.0.1 Average execution times (one control pass) for four controllers by VTune

In this chapter, we investigate buffer tuning in the case of nomadic users with small form factor (SFF) devices passing through a wireless smart space.

In order to thoroughly and vigorously investigate how dynamic buffer size tuning can benefit time-critical applications a natural environment is needed as the test-bed. The FLC is naturally the tuner candidate for the tests because of its stability and speed. The natural environment in which critical timing is always a consideration

is the pervasive computing as shown by Figure 8.0.1. Every pervasive computing environment has two parts: the *wireless smart space* and the *wired Internet part*, which provides the *pervasive computing infrastructure* (PCI). The smart space is a wireless cell served by at least one surrogate, which provides the necessary assistance to the clients and serves as a gateway to other PCI nodes. A client in the smart space is actually a SFF (small-form-factor) device carried by the nomadic user. The duration of stay by a human nomadic user in the smart space is normally short and is characterized by the mass transit traffic through the cell (e.g. train station or airport). When nomadic users are passing through the smart space, they may make different kinds of requests to a popular server, which could be located in the surrogate (as shown in Figure 8.0.1 as an agent). If the agent server cannot provide the service, it enlists help from other nodes in the PCI through the surrogate in a transparent manner. This kind of cooperation is called *cyber foraging* [Garlan2002]. If *dt* (i.e. delta t) represents the average transit interval/duration through the smart space, then many requests *dRQ* (i.e. delta RQ; where RQ means requests) may be made to the agent server within the interval. The rate of request, namely, dRQ/dt can be steep. The requests are usually queued in the agent's request buffer before they are served. The number of requests, however, is tied with the characteristic of the mass transit traffic and dRQ/dt. At peak periods the agent is easily inundated by a sudden influx of requests, which leads to the following undesirable consequences:

a) *Overflow in the agent's buffer*: If this happens, then there could be widespread retransmissions by the SFF clients leading to more congestion and longer

service roundtrip time (RTT). The SFF clients do not even have the chance to exploit the benefit of cyber foraging before they leave the smart space.

b) E-business failure: The nomadic users become unhappy because they could not make use of the wireless cell to complete the necessary business on the run. This means that the e-business, which provides the agent server, would be the ultimate victim. This can be prevented if the communication congestion is resolved in a user-transparent manner so that the benefits from cyber foraging can be obtained.

A solution to enhance the chance of cyber foraging exploitation is dynamic buffer size tuning. The aim is to tune the agent's buffer size adaptively on the fly so that the buffer length always covers the queue length. In this way the chance of buffer overflow at the user level is eliminated. The FLC dynamic buffer size tuner for user-level application easily achieves this goal.

Location sensitivity is an essential element in both mobile and pervasive computing. In mobile computing this sensitivity lets the Internet-based system know exactly the locations of the SFF clients [Garlan2002]. Pervasive computing takes mobile computing one step further by tracing and anticipating a nomadic user's intent and movement so that service can be prepared proactively in an invisible (non-intrusive) manner. An important attribute for a pervasive system is to effectively maintain a smart space [Weiser1991] and support it with rich information technology capabilities. This is demonstrated by several well-know experimental examples today, namely, Endeavor (at UC Berkeley), Aura (at Carnegie Mellon University) and

Oxygen (at MIT). There is presently no dominant location sensing mechanism/technology because any extant mechanism (e.g. Cricket, Blue-tooth, GPS, active badge, e911, and the IEEE802.11 family) is good for only a narrow band of situations. Therefore, effective location sensing is still an active area of research [Hightower2001]. Once the position is sensed and known, the client's intent can be anticipated and supported. One possible intention is location-aware information retrieval [Cool2002]. In this aspect the client, which is a SFF (small-for-factor) mobile device (e.g. PDA or a portable PC carried by a user) communicates with the pervasive-computing infrastructure [Brown2001, Brown2002]. The client-infrastructure communication is wireless and the surrogate, which is a sever node wired to the rest of the Internet, provides the necessary assistance to serve the user's requests through the client device.

A reasonable business scenario of location-aware information retrieval in Hong Kong is a foreign buyer who has just arrived at the airport trying to locate a list of reputable furniture manufacturers in town. After the plane has landed the buyer immediately engages the local pervasive-computing environment through a SFF device and discovers the appropriate surrogate. A surrogate is any assigned hardware device, which is physically wired to the Internet by a high-speed network and assists a mobile client temporarily. Through wireless communication provided by the smart space the buyer passes its request for a list of manufacturers to the surrogate (gateway). This surrogate tries to find the information within its database or it may pass the request to other information stations (nodes) in the PCI. A surrogate solicits help from other collaborating Internet nodes under the following conditions: a) it is

too busy because there are too many similar requests, and b) the site has impoverished bandwidth and thus it is necessary to re-direct the request to another surrogate to speed up the service and reduce the overall roundtrip time (RTT). Soliciting help from other wired nodes is known as cyber foraging [Satyanarayanan2001, Patterson2003]. Dynamic buffer size tuning by using the FLC can prevent a surrogate from being inundated by the clients' requests. The consequence of inundation is buffer overflow, which can happen easily during periods of peak demand if the situation is not handled properly. User-level buffer overflow as such makes the service provision link unreliable/undependable, and a client may need to repeatedly resend the same request many times.

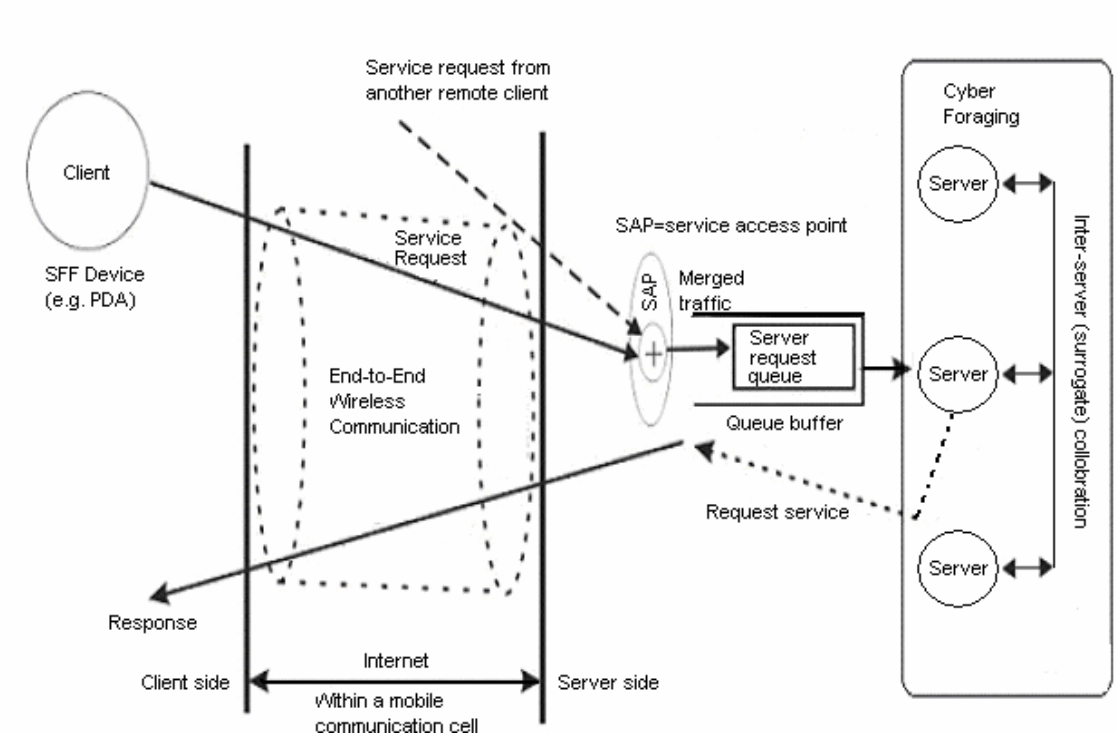


Figure 8.0.2 Client/server (surrogate) end-to-end wireless interaction

In a public place such as the airport service, requests to the local surrogate or gateway ties in with the traffic of the physical travellers. If the physical traveller traffic is LRD (*long-range dependence*), then the service requests traffic to the surrogate would likely follow suit. If the surrogate has a fixed buffer size to accommodate these requests, then overflow can occur. This is a serious problem because no matter how powerful the underlying pervasive-computing infrastructure is, the user cannot benefit from it. Buffer overflow means that some requests would be delayed from reaching the stage of cyber foraging, leading to much longer service roundtrip times. The observation by [Lewis1996], makes the point that cyber foraging yields speedup because different servers/surrogates work in parallel to provide the necessary service. Under Markovian conditions cyber foraging can be represented by the M/M/n model, where *n* is the number of collaborating surrogates or information stations. The speedup S by cyber foraging with *n* nodes can, therefore, be visualized as $S = \frac{(1-\rho/n)}{(1-\rho)}$, where ρ is the surrogate utilization.

In reality the transient mass transit population would definitely increase the volume of the communication between SFF mobile clients and a surrogate [Malla2003], especially at peak hours. It is inevitable that in any smart space, which is supported by a predefined number of SFF-surrogate connections, new connection requests are dropped once the maximum number is exceeded. As a result further client requests will be lost and retransmissions increase [Jamjoom2004]. From our own experience and that of others, we note that any sudden change in the traffic pattern of client requests to a surrogate can make the latter's buffer overflow. The Internet traffic pattern involves both wired and wireless communications. It is

normally unpredictable because it can change suddenly, for example, from SRD (*short-range dependence*) such as Markovian to LRD (*long-range dependence*) such as heavy-tailed and self-similar, or multifractal [Medina2000, Molnar1999].

8.1 LOCATION-AWARE SIMULATIONS

The FLC's efficacy in supporting more dependable location-aware information retrieval is verified by simulation. There are two different sets of experiments. The first set evaluates the execution time of the FLC Java prototype because dynamic buffer tuning is naturally time-critical. If the execution time is too long, the computed solution cannot remedy the actual problem in a real-time manner because it has long passed. The computed solution would end up correcting a spurious problem leading to undesirable/deleterious effects. The timing analysis is carried out with the *Intel's VTune Performance Analyzer* [VTune2002], which measures the FLC execution time in the number of neutral clock cycles. The second set of experiments verifies that the FLC indeed eliminates surrogate buffer overflow independent of the IAT (*inter-arrival time*) traffic patterns.

The experiments were carried out on the Aglets mobile agent platform, which is chosen for the following reasons: a) it is stable, b) it has rich user experience, and c) it is designed for the Internet and this makes the experimental results scalable for the open Internet. The set up for the experiments is shown in Figure 8.1.1, in which the driver and the server are aglets (*agile applets*) collaborating in a client/server relationship within a single computer. The driver picks a known waveform (e.g.

Poisson) or a trace, which embeds an unknown waveform for the wireless client/surrogate request traffic, from the table. It uses the pick to generate the inter-arrival times for the simulated merged traffic into the surrogate buffer. A "trace" is a file of pre-collected RTT, and the use of traces in simulations helps confirm that the FLC control precision and stability are indeed insensitive to the sudden changes in the incoming request traffic pattern. This confirmation is necessary because real-life Internet related traffic usually follows the power law and changes suddenly, for example, from LRD (*long-range dependence*) such as *self-similar* and *heavy-tailed* to SRD (*short-range dependence*) such as *Markovian.*

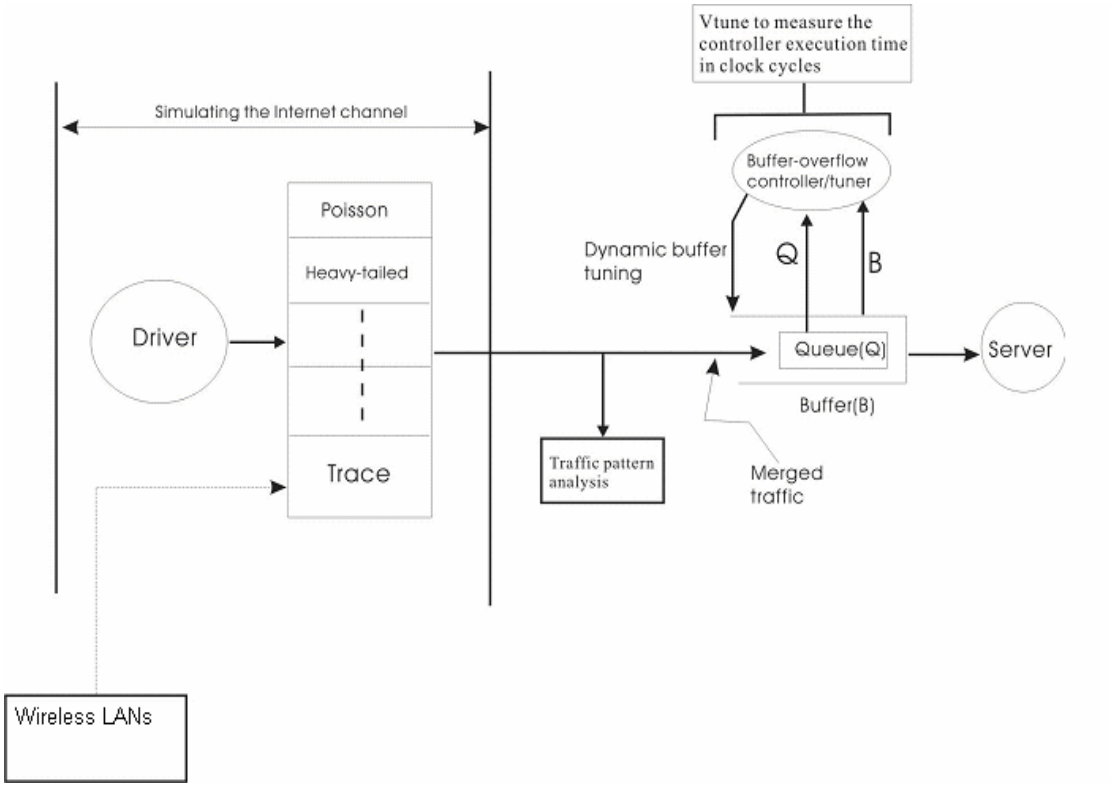


Figure 8.1.1 Verification of FLC stability in SFF-client/surrogate interactions

The waveforms in the experiments are always checked and identified, as indicated by the "*traffic pattern analysis*" box in Figure 8.1.1. In this way the response of the FLC to any specific waveform can be visualized in one-to-one correspondence. Waveform checking and identification is achieved by using the *E-RTPD Tool* [ATNAC2004], which includes different traffic filters/estimators (e.g. the real-time modified QQ-plot or RT-QQ). The basis of the RTPD tool is the *R/S* (*rescaled adjusted statistics*) mechanism. It is renamed the enhanced R/S or E-R/S because the *Convergence Algorithm* is incorporated as a component. It measures the Hurst (H) value and differentiates LRD (for $0.5 < H < 1$) from SRD (for $0 < H < 0.5$) for a discrete stochastic process X. After the LRD character is confirmed, for example, the RT-QQ filter can be invoked to check and confirm if the traffic pattern is heavy-tailed. Some traces used in the experiments are from the in-house SFF 802.11b connections [Trace] with the Lucent ORINOCO pc24e-h-fc wireless LAN card as the interface. In this section three different sets of experimental results are presented. Case 1 shows how the FLC makes the Hong Kong PolyU wireless environment more dependable. Case 2 shows how well the FLC can work with the wireless traces from the Stanford Mosquito Net. Case 3 shows that the FLC has worked well in the Faculty of Information Technology in the University of Technology Sydney campus.

Case 1: Department of Computing, The Hong Kong Polytechnic University

The aim is to evaluate how the FLC dynamic buffer tuning process performs in the Hong Kong PolyU wireless SFF-Client/surrogate environment. For the wireless

LAN traffic trace chosen for demonstration the R/S plot of the *RTPD Tool* yields H=0.7069, with 97.89% confidence for its LRD character (Figure 8.1.2).

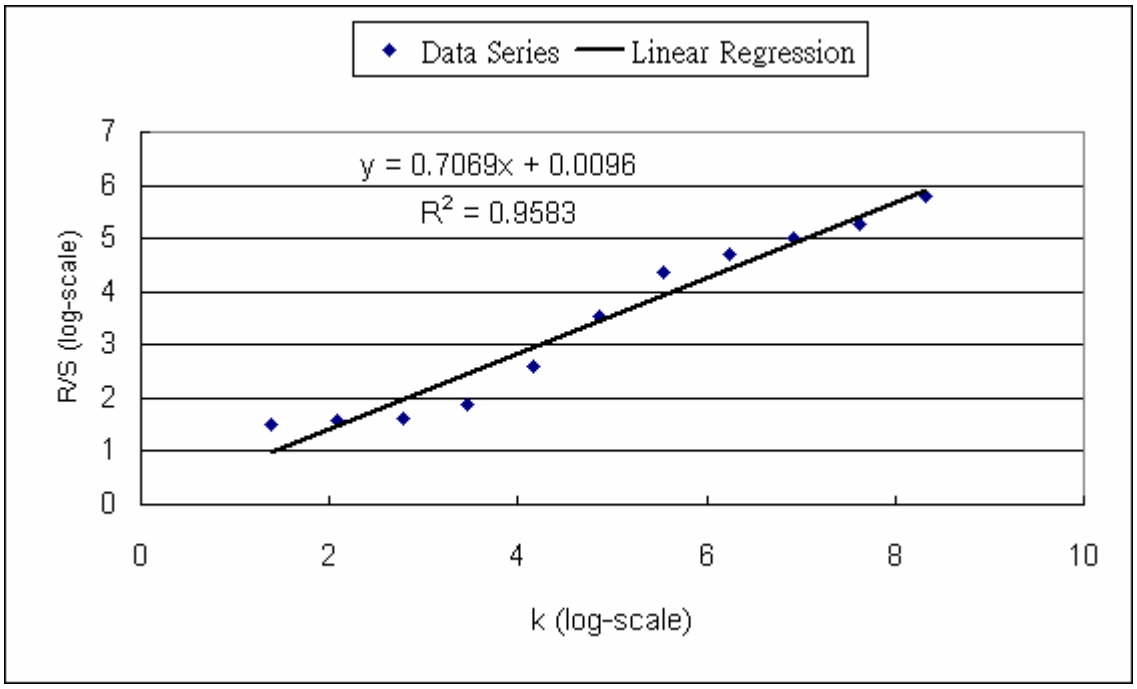


Figure 8.1.2 Trace analysis/identification by RTPD's R/S estimator

From the preliminary experimental results, as shown in Figure 8.1.3, the following are concluded: a) the FLC maintains the Δ safety margin correctly and consistently for different QOB_R values and traffic conditions, b) it eliminates the surrogate buffer overflow efficaciously, and c) it has a shorter control cycle time than the PIDC's, which was also tested for comparison purposes. The "*buffer overflow controller/tuner*" remark in Figure 8.1.1 indicates where the FLC or PIDC can be installed for the particular simulation. Figure 8.1.4 is plotted for the same trace as Figure 8.1.3 and it shows that the FLC convergence to the QOB_R reference is quicker, smoother and more accurate than the PIDC.

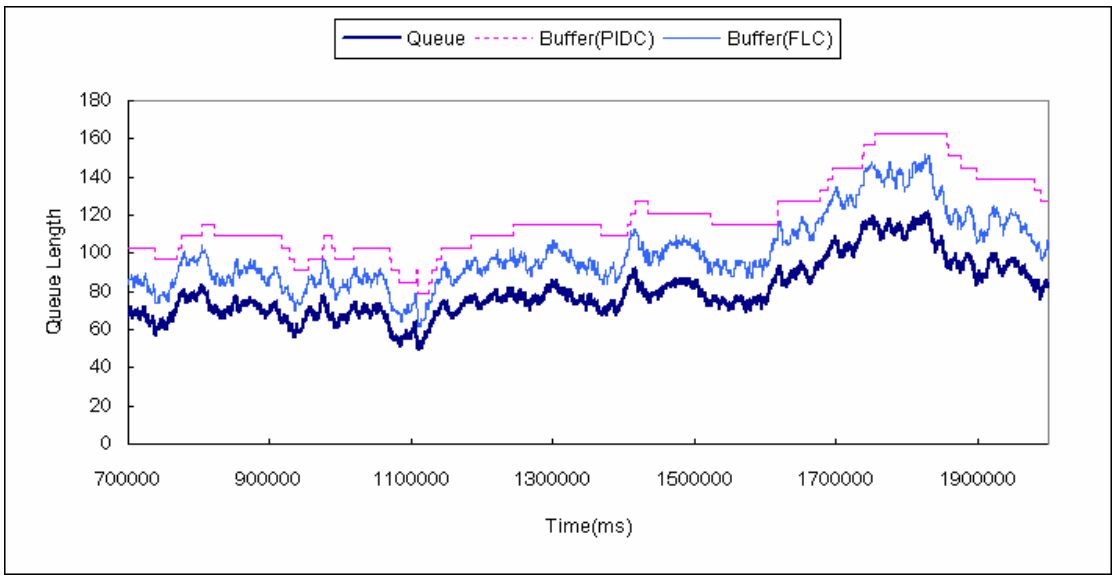


Figure 8.1.3 FLC and PIDC performances in SFF-client/surrogate buffer overflow control

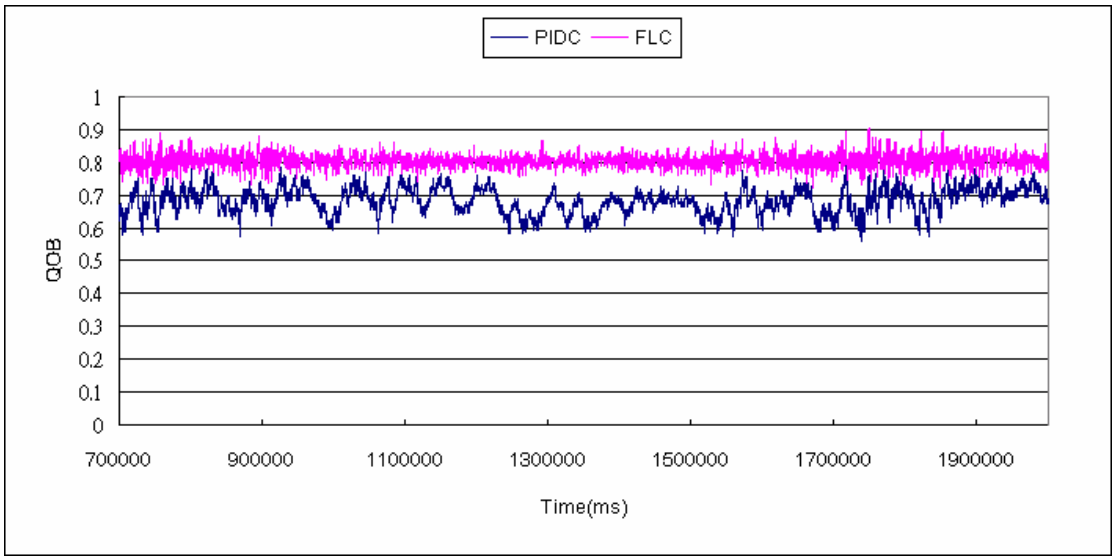


Figure 8.1.4 More accurate and faster FLC trend line than the PIDC's

Case 2. Stanford Mosquito Net

This simulation shows how FLC would respond in a different SFF-clients/surrogate traffic environment. In this case the wireless traffic is the Stanford Mosquito Net [Tang2000]. The plot by E-R/S in the E-RTPD shown in Figure 8.1.5 indicates that the trace is LRD (H=0.716) with 98.34% of confidence.

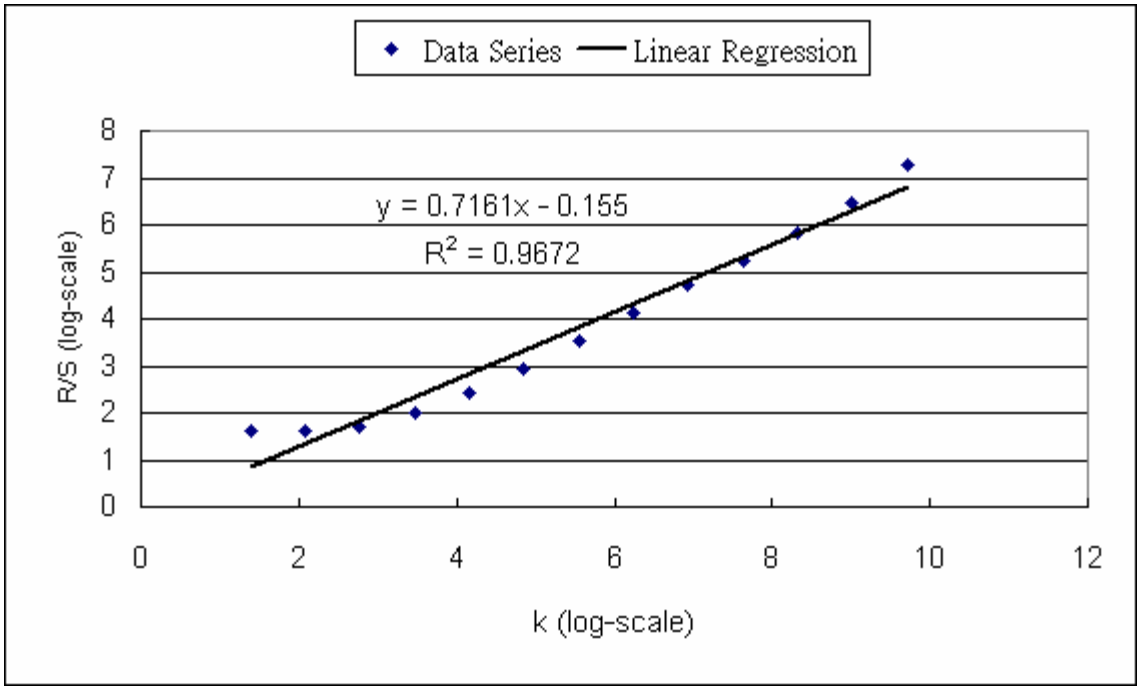


Figure 8.1.5 Trace analysis/identification by RTPD(R/S estimator) H=0.716

Figure 8.1.6 compares the FLC and PIDC performance for the same trace. It shows that the FLC controlled buffer length always covers the queue length by the safety margin of $\Delta = 0.2$. The buffer length controlled by the PIDC, however, differs by locking up unused buffer space consistently. This kind of unnecessary memory locking may deprive the system of recyclable memory and lead to poor performance.

Figure 8.1.7 shows that the FLC controlled output is smoother and more accurate than the PIDC's.

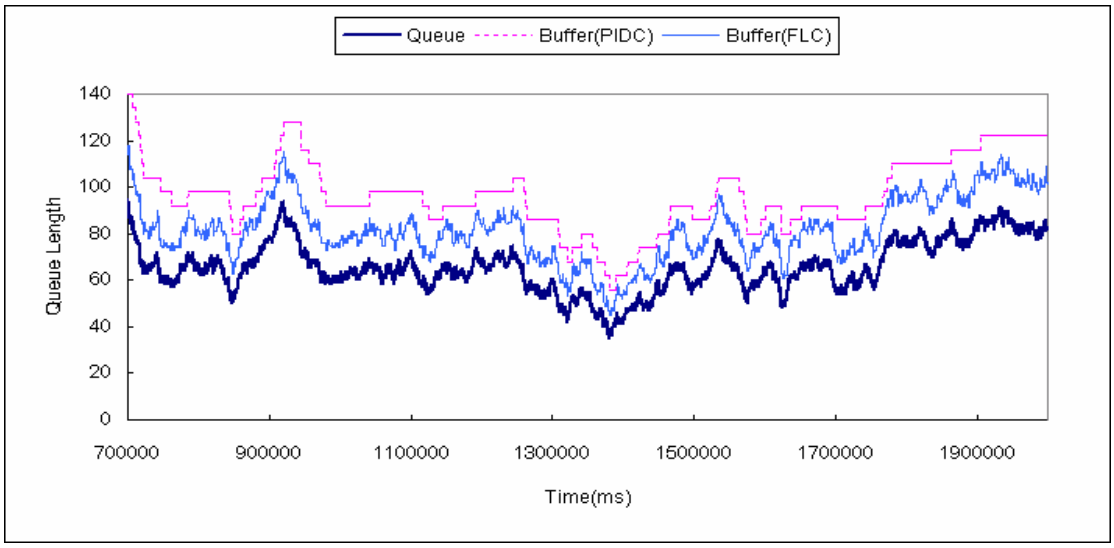


Figure 8.1.6 FLC and PIDC responses to the Stanford Mosquito Net trace

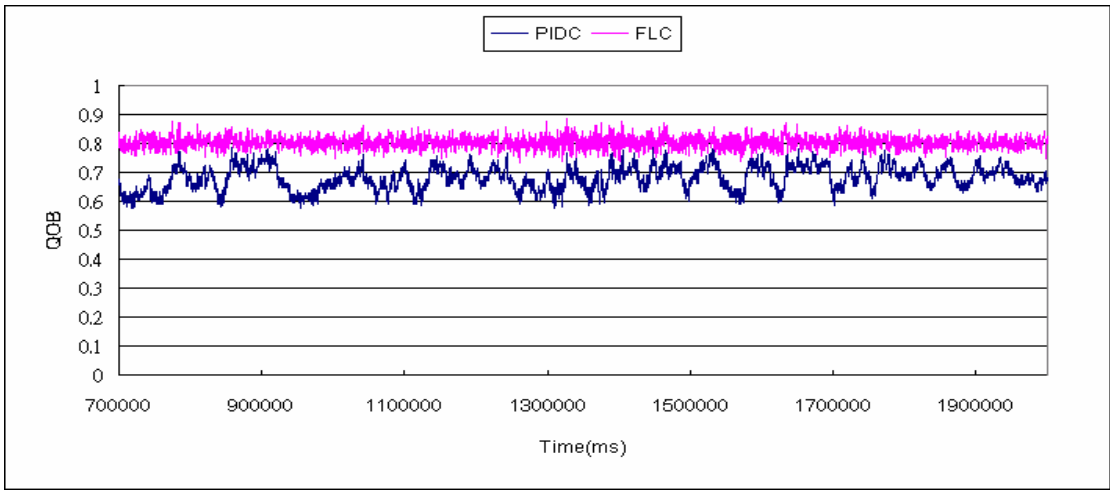


Figure 8.1.7 Performance comparison between the FLC and the PIDC

Case 3. Faculty of Information Technology, University of Technology Sydney

It evaluates how the FLC dynamic buffer tuning process performs for the UTS wireless SFF versus client/surrogate environment. The UTS wireless traffic traces selected for demonstration here have H=0.54 with 95.8% confidence for its LRD character. The plot in Figure 8.1.8 is produced by the E-R/S of the E-RTPD package. The experimental results given in Figure 8.1.9 show the following: a) the FLC maintains the Δ safety margin correctly and consistently for different QOB_R values and traffic conditions, b) it eliminates the surrogate buffer overflow efficaciously, and c) it has a shorter control cycle time than the PIDC's, which was also tested for comparison purposes. Figure 8.1.10 is plotted for the same trace as Figure 8.1.9 and it shows that the FLC convergence to the QOB_R reference is quicker, smoother and more accurate than the PIDC.

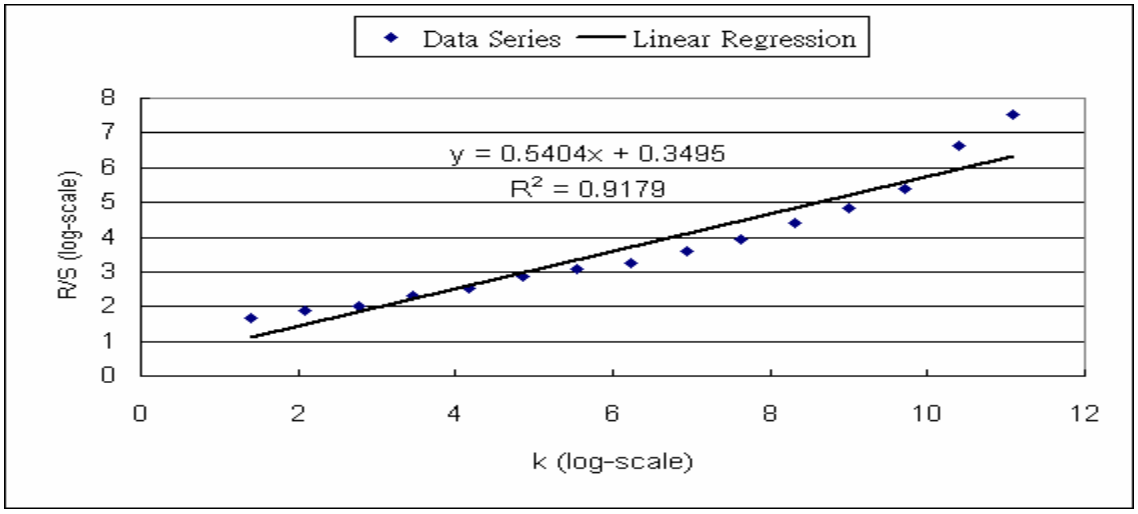


Figure 8.1.8 UTS Trace analysis/identification by RTPD's R/S estimator

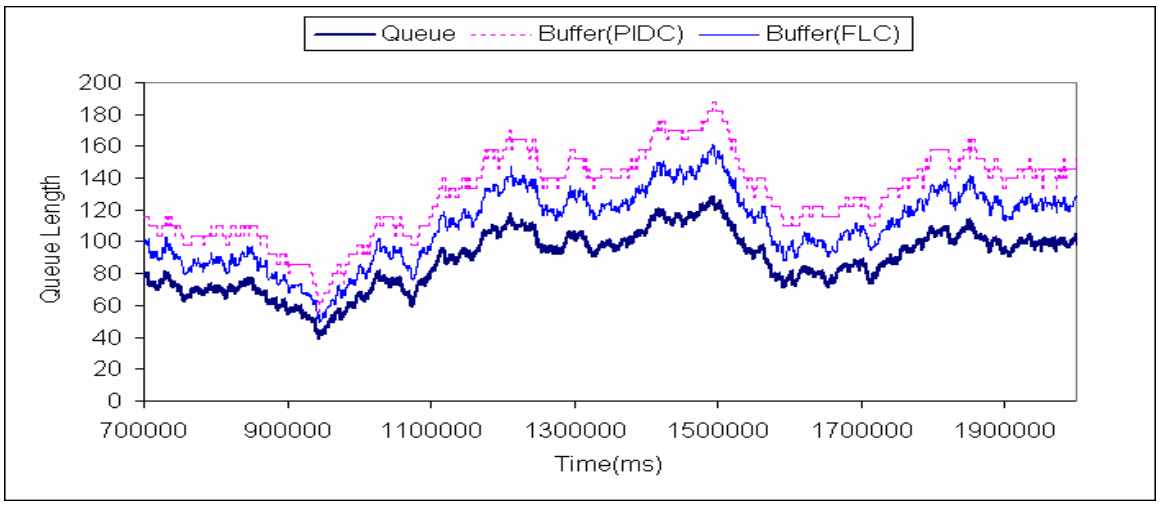


Figure 8.1.9 FLC and PIDC SFF-client/surrogate buffer overflow control performances for the UTS trace used in Figure 8.1.8

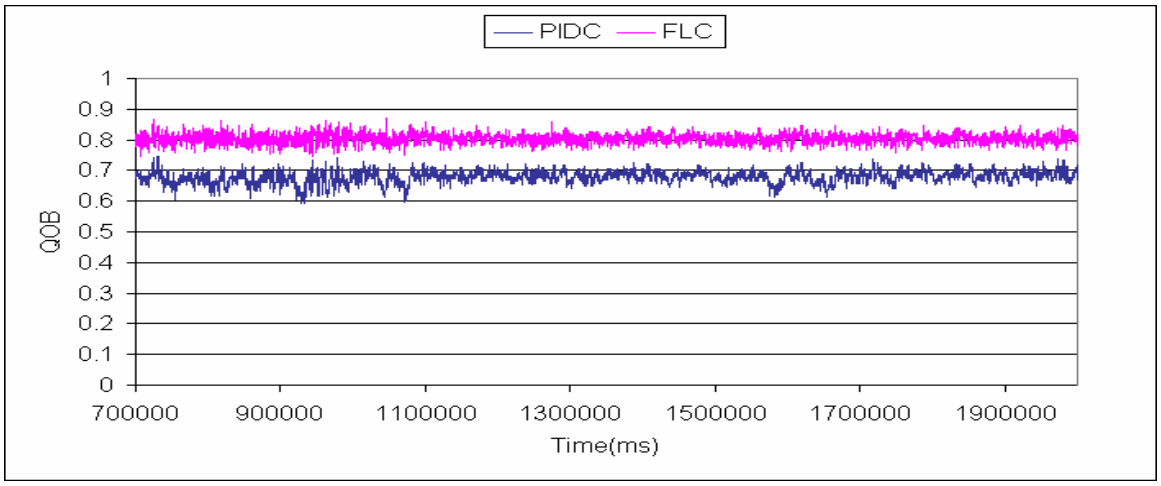


Figure 8.1.10 More accurate and faster FLC trend line than the PIDC's for the UTS trace

8.2 CONNECTIVE SUMMARY

Within a smart space for mobile/pervasive computing the number of SSF clients trying to hook onto the surrogate ties in with the transient mass transit traffic. The asymmetric rendezvous between the surrogate and the many clients that demand its service may inundate the surrogate request buffer to overflow. If this happens, the clients would lose the chance to benefit from the cyber foraging supported by the background mobile/pervasive computing infrastructure. The FLC, however, can tune the surrogate buffer size on the fly to make sure that it always covers the request queue size by the given Δ safety margin. As a result it eliminates any chance of transient buffer overflow due to the transient transit mass and makes the SFF-client/surrogate interaction more dependable. The simulations with different wireless traces indicate that the FLC is indeed an efficacious solution for more dependable location-aware applications such as pervasive information retrieval. From the literature search while preparing for the location-aware experiments, it was found that dynamic buffer size tuning is very useful for e-health applications that are usually time-critical [Epocrates]. Tele-diagnosis over the Internet is a typical time-critical example because timeliness of the diagnostic result determines if a patient would be saved in time in an emergency case. Dynamic buffer size tuning can reduce "*procrastination*" due to retransmissions caused by user-level buffer overflow. As a result it could help save lives.

CHAPTER 9 CONCLUSION, ACHIEVEMENTS AND FUTURE WORK

In my MPhil thesis I proposed four original dynamic buffer size tuners for user-level applications. They are as follows:

1) PIDC ("*proportional (P) + integral(I) + derivative(D)*" Controller): It is algorithmic and always eliminates user-level buffer overflow even with two shortcomings: a) it locks unused buffer space, and b) it does not have a safety margin and therefore the queue length can get dangerously close to the buffer length threatening possible overflow.

2) GAC (*Genetic Algorithm Controller*): It is the "PIDC + genetic algorithm (GA) + $\{0,\Delta\}^2$ objective function" combination. The GA moderates the PIDC process so that the outcome is always within the $\pm\Delta$ safety margins about the steady-state reference symbolically represent by "0" in $\{0,\Delta\}^2$. The GA eliminates the PIDC shortcomings but produces occasional buffer overflow because it does not guarantee the global-optimal solution of the solution *hyperplane*.

3) FLC (*Fuzzy Logic Controller*): It was proposed to preserve the GAC merits and eliminate the occasional buffer overflow. It is this combination: "*PIDC + fuzzy logic + $\{0,\Delta\}^2$ objective function*". The fuzzy logic moderates the PIDC control process functionally similar to the GA.

4) NNC (*Neural Network Controller*): It works with the $\{0,\Delta\}^2$ objective function but does not include PIDC. Its proposal was inspired by the successful experience of using neural networks in AQM (active queue management) algorithms that prevent

network congestion at the system/router level. AQM methods differ from the dynamic buffer size tuners by using a fixed-size buffer.

These tuners succeed in providing performance enhancement and fault tolerance to client/server interactions over logical TCP channels of the Internet by eliminating the user-level overflow. They are suitable for time-critical applications because they have short control cycle times as measured by the Intel's VTune Performance Analyzer. The sizes of their Java prototypes and execution/cycle times are listed in Table 9.1.

Control models	Lines of Java code for controller implementation (Ln)	Average line of code in Pentium III assembler program	Clock/T cycles per assembly line (Pentium lll 933MHz) (T)	*Average number of T cycles required for convergence* (NTC)	Measured average number of T cycles per convergence computation cycle (TCC)
Proportional Integral Derivative Controller (PIDC)	105	525	9	4725	205
Genetic Algorithm Controller (GAC)	111	555	9	4995	475
Fuzzy Logic Controller (FLC)	116	580	9	5220	255
Neural Network Controller (NNC) (Back propagation architecture [*Input-Hidden-Output*]: *10-20-1*)	240	1200	9	10800	10800

Table 9.1 Empirical comparison of the four proposed controllers

Although my MPhil research had significant contributions in user-level buffer overflow control and provision of shorter service roundtrip time (RTT) for client/server interactions over the Internet, it has left several unaddressed issues as follows:

1) *In the aspects of traffic ill effects*: a) Is it possible to calibrate the ill effects off-line so that the tuners can use these calibrations to ward off traffic

impedance by fine-tuning its dynamic buffer tuning process adaptively? b) If so, then how can the current Internet traffic pattern be deciphered on the fly (on-line) so that the off-line calibrations can be applied selectively?

2) *For FLC*: a) Is it possible to have an optimal design? b) Is it possible to make the tuner self-reconfigurable (especially with respect to traffic pattern changes)?
3) *For NNC*: a) Is it possible to prune the NNC configuration on the fly so that its control cycle time can be consistently and adaptively reduced? b) Is there a correlation between control accuracy and the number of hidden neurons in the NNC back-propagation architecture? (The procedure to provide the answer is called *sensitivity analysis*.)

Providing solutions to these unaddressed issues has become the motivation of my PhD research. In the process I have achieved the following:

1) *For real time traffic analysis*: Two traffic filters have been proposed: *real-time modified QQ-plot* (or simply RT-QQ) and *self-similarity (S^2) filter*. These filters identify the Internet traffic patterns on the fly. The RT-QQ recognizes heavy-tailed distributions and the S^2 filter identifies self-similarity.
2) *For FLC*: a) an optimal range is found for FLC design, and b) a way was found to make the FLC adaptive/reconfigurable by squeezing the "*don't care*" state range threshold in a dynamic manner.
3) *For NNC*: a) the HBP (*Hessian Based Pruning*) approach was proposed for

pruning or optimizing the NNC configuration on the fly and as a result its average execution time (i.e. control cycle time) is reduced, and b) sensitivity analysis was conducted and the results confirm that more hidden neurons do not necessarily mean better NNC performance

The results from my PhD research have contributed to 19 publications (5 journals and 14 conferences) so far, and I have achieved all the objectives planned for my thesis at the outset. Following the experience gained in my research I propose that the future work should include the following:

a) to investigate the issue of how to choose the limits for effective Gaussianity tests in the CAB mechanism,

b) to deepen the investigation into why "*heavy-tailedness*" is not a necessary condition of self-similarity, and

c) to investigate how the dynamic buffer size controllers, especially the FLC, can best support pervasive computing based e-applications such a telemedicine.

My PhD research has achieved the planned objectives, which provide solutions to all the unaddressed issues left behind by my previous MPhil thesis. The new findings include the following:

1) For FLC (Fuzzy Logic Controller): a) an optimal FLC design range is confirmed empirically and b) a reconfigurable/adaptive FLC model is proposed and verified.

2) For NNC (Neural Network Controller): a) sensitivity analysis confirms that there is no obvious advantage in having more than 20 hidden neurons in the NNC's back-

propagation neural network (NN) and b) the Hessian Based Pruning (HBP) method is proposed and verified for optimizing the NN architecture on the fly and this reduces the NNC control cycle time successfully by at least seven percent.

3) For real-time traffic analysis: I successfully made use of the accumulated experience by the COMP Team and in return I proposed and verified two real-time traffic filters/estimators: real-time modified QQ-plot (or RT-QQ) and self-similarity (S^2) filter. The inclusion of these filters into the real-time traffic detector (RTPD) proposed by the Team converts it into the Enhanced RTPD or E-RTPD. I successfully used these filter to help the reconfigurable FLC (i.e. A-FLC [p12]) to fine-tune itself on the fly to nullify the ill effects on its stability and accuracy by traffic pattern changes.

The findings from my PhD research, as listed above, not only provide a solid basis and directions for future exploration in the area of dynamic buffer size control but also contributed to 19 publications (5 journal and 14 conferences) as follows:.

Five refereed journal papers

[p1] Wilfred W.K. Lin, Allan K.Y. Wong and Tharam S. Dillon, Application of Soft Computing Techniques to Adaptive User Buffer Overflow Control on the Internet, to appear in the IEEE Transactions on Systems, Man and Cybernetics, Part C

[p2] Wilfred W.K. Lin, Allan. K. Y. Wong and Richard S.L. Wu, Applying Fuzzy Logic and Genetic Algorithms to Enhance the Efficacy of the PID Controller in

Buffer Overflow Elimination for Better Channel Response Timeliness over the Internet, to appear in the Concurrency and Computation: Practice & Experience

[p3] Wilfred W.K. Lin, Allan K. Y. Wong and Tharam S. Dillon, A Novel Fuzzy-PID Dynamic Buffer Tuning Model to Eliminate Overflow and Shorten the End-to-End Roundtrip Time for TCP Channels, Lecture Notes in Computer Science, Volume 3358 / 2004, pp.783-787

[p4] Wilfred W.K. Lin, Allan K. Y. Wong and Tharam S. Dillon, HBP: An Optimization Technique to Shorten the Control Cycle Time of the Neural Network Controller (NNC) that Provides Dynamic Buffer Tuning to Eliminate Overflow at the User Level, International Journal of Computer Systems Science & Engineering, 19(2), March 2004, pp. 85-94

[p5] Wilfred W.K. Lin and Allan K.Y. Wong, A Novel Neural Network Controller to Eliminate Buffer Overflow in Client/Server Based Internet Applications, WSEAS Transactions on Systems, 2(3), July 2003, pp.607-615

Fourteen refereed conference papers

[p6] Wilfred W.K. Lin, Allan K.Y. Wong and Tharam S. Dillon, A Novel R^2-FLC Dynamic Buffer Size Tuner to Support Time-Critical Applications over the Internet by Improving Logical Channel Fault Tolerance to Shorten Roundtrip Time, to appear

in the 11th International Symposium on Pacific Rim Dependable Computing (PRDC-2005) Changsha, Hunan, China

[p7] Wilfred W.K. Lin, Allan K.Y. Wong and Tharam S. Dillon, FLC: A Novel Dynamic Buffer Tuner for Shortening Service Roundtrip Time over the Internet by Eliminating User-Level Buffer Overflow on the Fly, to appear in the 6th International Workshop on Advanced Parallel Processing Technologies(APPT'05), Hong Kong

[p8] Wilfred W.K. Lin, Allan K. Y. Wong, Tharam S. Dillon and Richard S.L. Wu, A Novel Real-Time Self-Similar Traffic Detector/Filter to Improve the Reliability of a TCP Based End-to-End Client/Server Interaction Path for Shorter Roundtrip Time, to appear in the 2nd International Conference on E-Business and Telecommunication Networks, Reading, United Kingdom

[p9] Wilfred W. K. Lin, Tharam S. Dillon and Allan K.Y. Wong, An Internet-Based Distributed Manufacturing System Utilizing a Recurrent Neural Network Controller for Dynamic Buffer Size Tuning to Prevent User-level Buffer Overflow and Shorten the Service Roundtrip Time, to appear in the 3rd International IEEE Conference on Industrial Informatics 2005, Perth, Australia (Best presentation award)

[p10] Wilfred W.K. Lin, Tharam S. Dillon and Allan K.Y. Wong, Apply FLC-based Dynamic Buffer Size Tuning to Shorten the Information Retrieval Round Trip Time

in the Mobile Location-aware Environments, Proceedings of the 4th International Conference on Mobile Business, Sydney, Australia, pp. 507-513

[p11] Wilfred W.K. Lin, Allan K. Y. Wong and Tharam S. Dillon, A Novel Traffic Independent NNC for Dynamic Buffer Tuning to Shorten the RTT of a TCP Channel, Proceedings of the 3rd International Conference on Information Technology and Applications, Sydney, Australia, pp. 647-652

[p12] Wilfred W.K. Lin, Tharam S. Dillon and Allan K.Y. Wong, A Recurrent Neural Network Controller for Dynamic Buffer Size Tuning to Provide More Dependable Client Server Communications, Proceedings of the International Conference on Dependable Systems and Networks (Fast Abstract), Yokohama, Japan, pp. 20-21

[p13] Wilfred W. K. Lin, Allan K. Y. Wong and Tharam S. Dillon, A Novel Fuzzy Logic Controller (FLC) for Shortening the TCP Channel Roundtrip Time by Eliminating User Buffer Overflow Adaptively, Proceedings of the 28th Australasian Computer Science Conference 2005 (ACSC'2005), Newcastle, Australia, pp. 29-37

[p14] Wilfred W. K. Lin, Richard S.L. Wu, Tharam S. Dillon and Allan K. Y. Wong, A Novel Real-Time Traffic Pattern Detector for Internet Applications, Proceedings of the 2004 Australian Telecommunication Networks and Applications Conference(ATNAC), Sydney, Australia, December 2004, pp. 224-227

[p15] Wilfred W.K. Lin, Allan K. Y. Wong and Tharam S. Dillon, A Novel Adaptive Fuzzy Logic Controller (A-FLC) to Reduce Retransmission and Service Roundtrip Time for Logical TCP Channels over the Internet, Proceedings of the 2004 International Conference on Embedded And Ubiquitous Computing (EUC04), LNCS 3207, Aizu, Japan, August 2004, pp.941-951

[p16] Allan K. Y. Wong, Wilfred W.K. Lin and Tharam S. Dillon, HBP: A Novel Technique for Dynamic Optimisation of the Feed-Forward Neural Network Configuration, Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics, Setubal, Portugal, August 2004, pp.346-349

[p17] Wilfred W.K. Lin and Allan K.Y. Wong, A Novel Fuzzy Logic Controller to Eliminate Buffer Overflow at the User Level over the Internet, Proceedings of the 24th IEEE International Real-Time Systems Symposium, (WIP Session), Cancun, Mexico, December 2003, pp.71-74

[p18] Wilfred W.K. Lin and Allan K.Y. Wong, A Novel Adaptive Fuzzy Logic Controller (FLC) to Improve Internet Channel Reliability and Response Timeliness, Proceedings of the IEEE Symposium on Computers and Communications (ISCC'2003), Antalya, Turkey, July 2003, vol. II, pp.1347-1352.

[p19] Wilfred W.K. Lin, Allan K. Y. Wong and Tharam S. Dillon, HBM: A Suitable Neural Network Pruning Technique to Optimize the Execution Time of the Novel Neural Network Controller (NNC) that Eliminates Buffer Overflow, Proceedings of the 8th 2003 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2003), Las Vegas, USA, June 2003, vol. II, pp. 555-560

The contributions by the published papers above can be divided into different groups as follows:

Group 1: It consists of the following: p1, p2, p3, p6, p7, p10, p13, p15, p17, p18. The specific contributions by these papers are: a) an optimal design range is confirmed for FLC design, and b) a way was found to make the FLC adaptive/reconfigurable by squeezing the "*don't care*" state range threshold in a dynamic manner.

Group 2: It consists of the following: p1, p4, p5, p9, p11, p12, p16, p19. The specific contributions by these papers are: a) the Hessian Based Pruning method can indeed optimize the NNC configuration on the fly and as a result reduces its average execution time (i.e. control cycle time), and b) sensitivity analysis confirms that more hidden neurons in the NNC architecture does not necessarily yield better performance

Group 3: It consists of the following: p8 and p14. The specific contributions by these papers are real-time traffic analysis and pattern detection. The inclusion of my *real-*

time modified QQ-plot (or simply RT-QQ) and the S^2 filter into the extant RTPD (real-time traffic pattern detector) convert it to the enhanced version (i.e. E-RTPD).

AREA OF FUTURE RESEARCH

In the research process I have uncovered different relevant problems, and after scrutinizing carefully I suggest that the following items should be investigated first in the near future because of their "bridging nature" to other relevant issues in dynamic buffer size tuning:

a) The first is to investigate how limits can be appropriately chosen for on-line Gaussianity tests. The successful use of traffic filters depends on whether stationarity for an aggregate in a discrete stochastic process can be confirmed. For example, the RT-QQ and S^2 filters for on-line application work for the "*Hurst and stationarity*" conditions.

b) The second is to deepen the investigation into why "*heavy-tailedness*" is not a necessary condition of self-similarity. So far, this issue has rarely been explored. More confirmation is needed so that the real need of designing different real-time filters for heavy-tailed distributions and self-similar waveforms is there.

c) Although the PIDC, FLC and NNC tuners proposed in my MPhil thesis were deployed, they were applied only in the wired Internet environment. In fact, the Internet is getting more mixed in the sense that it is made up of wireless and wired (W&W) parts. The W&W setup is typical of pervasive computing environments, which are called mobile distributed systems (MDS). This kind of setup is getting

more popular in different areas of applications such as telemedicine. Therefore, there is a need to investigate how the dynamic buffer size controllers, especially the improved versions for the FLC and NNC. In particular the FLC should be examined more carefully in the light of how it can best support pervasive computing based e-applications.

CHAPTER 10 BIBLIOGRAPHICAL REFERENCES

[Abry2000] P. Abry, P. Flandrin, M. S. Taqqu and D.Veitch, Wavelets for the Analysis, Estimation and Synthesis of Scaling Data, in Self Similar Network Traffic Analysis and Performance Evaluation, K. Park and W. Willinger, Eds., Wiley, pp.39-89

[Aloisio1980] A. Aloisio, The Central Limit Theorem for Real and Banach Valued Random Variables, Wiley, 1980

[Alvisi1998] L. Alvisi and K. Marzullo, Message Logging: Pessimistic, Optimistic, Causal, and Optimal, IEEE Transactions on Software Engineering, 24(2), February 1998, 149 -159

[Amir1995] Y. Amir, L. Moser, P. Melliar-Smith, D. Agarwal, and P. Ciarfella, The Totem Single-ring Ordering and Membership Protocol, ACM Transactions on Computer Systems, 13(4), November 1995, 311-342

[Arvotham2001] S. Arvotham, R. Riedi and R. Barabniuk, Connection-Level Analysis and Modeling of Network Traffic, Proc. of the IEEE/ACM Internet Measurement Workshop, 2001

[Athuraliya2001] S. Athuraliya, S.H. Low, V.H. Li and Q.H. Yin, REM: Active Queue Management, IEEE Network, 15(3), May-June 2001, 48-53

[Avizienis2004] A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr, Basic Concepts and Taxonomy of Dependable and Secure Computing, IEEE Transactions on Dependable and Secure Computing, 1(1), January-March 2004, 11-33

[Aweya1998] J. Aweya, Neurocontroller for Buffer Overload Control in a Packet Switch, IEE Proc. Communications, 145(4), August 1998, 227-233

[Aweya2002] J. Aweya, M. Ouellette and D.Y. Montuno, Multi-level Active Queue Management with Dynamic Thresholds, Computer Communications, 25(8), May 2002, 756-771

[Balakrishnan1997] H. Balakrishnan et al, A comparison of Mechanisms for Improving TCP performance over Wireless Links, IEEE/ACM Transactions on Networking, 5(6), 1997, 756-769

[Braden1998] Braden et.al., Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, April 1998

[Barford2004] P. Barford and J. Sommers, Comparing Probe- and Router-Based Packet-Loss Measurement, IEEE Internet Computing 8(5), September - October 2004, 50-56

[Berkan1997] R.C. Berkan, Fuzzy Systems Design Principles: Building Fuzzy IF-THEN Rule Bases, IEEE Press, 1997

[Bertsekas1982] D.P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, New York, Academic Press, 1982

[Brown2001] P. J. Brown and G. J. F. Jones, Context-aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering, Personal and Ubiquitous Computing, 5(4), December 2001, pp.253-263

[Brown2002] P.J. Brown, G.J.F. Jones, Information Access and Retrieval: Exploiting Contextual Change in Context-aware Retrieval, Proceedings of the 2002 ACM symposium on Applied computing, Madrid, Spain, March 2002, 650 - 656

[Cao2001] Cao et al, On the Nonstationarity of Internet Traffic, Proc. of the ACM SIGMETRICS'01, 2001, 102-112

[Cen2003] S. Cen, P.C. Cosman, G.M. Voelker, End-to-End Differentiation of Congestion and Wireless Losses, IEEE/ACM Transactions on Networking, 11(5), October 2003, 703-717

[Chatranon2004] G. Chatranon, M.A. Labrador and S. Banerjee, A Survey of TCP-Friendly Router-based AQM Schemes, Computer Communications, 27(15), September 2004, 1424-1440

[Comer1995] D. Comer, Internetworking with TCP/IP: v.1 Principles, Protocols, and Architecture, Englewood Cliffs, N.J. Prentice Hall, 1995

[Cool2002] C. Cool and A. Spink, Issues of Context in Information Retrieval (IR): An Introduction to the special issue. Information Processing & Management, 38(5), September 2002, pp.605-611

[Courriou2004] J.P. Courriou, Process Control : Theory and Applications, London ; New York : Springer, 2004

[Cottrel1999] L. Cottrel, M. Zekauskas, H. Uijterwaal and T. McGregor, Comparison of Some Internet Active End-to-End Performance Measurement Projects, http://www.slac.stanford.edu/comp/net/wan-mon/iepm-cf.html, 1999

[Cottrel2001] L. Cottrel, Passive vs. Active Monitoring, http://www.slac.stanford.edu/comp/net/wan-mon/passive-vs-active.html, March 2001

[Crawford2000] J.M. Crawford, A Scalable Architecture for Maximizing Concurrency, The 8th NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies in cooperation with Seventeenth IEEE Symposium on Mass Storage Systems, College Park, USA, March 2000, 253-258

[Cristian1999] F. Cristian and C. Fetzer, The Timed Asynchronous Distributed System Model, IEEE Transactions on Parallel and Distributed Systems, 10(6), June 1999, 642 -657

[Crovella1997] Mark E. Crovella and Azer Bestavros, Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes, in

IEEE/ACM Transactions on Networking, 5(6), December 1997, 835-846

[Dunigan2003] T. Dunigan, TCP Auto-tuning Zoo, ww.csm.ornl.gov/~dunigan/net100/auto.html, 2003

[Elnozahy1999] M. Elnozahy, L. Alvisi, Y. M. Wang and D. B. Johnson, A Survey of Rollback-Recovery Protocols in Message-Passing Systems. Technical Report CMU-CS-99-148, School of Computer Science, Carnegie Mellon University, June 1999

[Epocrates] e-Pocrates, e-Pocrates http://www.epocrates.com

[Feng1999] W. Feng, D. Kandlur, D. Saha and K. Shin, "Blue: A New Class of Active Queue Management Algorithms" IBM Research Report, April 1999

[Finney1994] R. L. Finney, G. B. Thomas and M. D. Weir, Calculus, Addison-Wesley, 1994

[Fisk2000] M. Fisk and W. Feng, Dynamic Adjustment of TCP window Sizes, Technical Report LAUR 00-3321, Los Almos National Laboratory, 2000

[Floyd1993] S. Floyd and V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, 1(4), August 1993, 397-413

[Gallant1992] A. R. Gallant and H. White, On Learning the Derivatives of an Unknown Mapping and Its Derivatives Using Multiplayer Feedforward Networks, Neural Networks, vol. 5, 1992

[Gartner1999] F. Gärtner, Fundamentals of Fault-tolerant Distributed Computing in Asynchronous Environments, ACM computing surveys, 31(1), March 1999, 1-26

[Garbinato2000] B. Garbinato and R. Guerraoui, An Open Framework for Reliable Distributed Computing, ACM Computing Surveys 32(1) 2000, 1-4

[Hagan1996] M. Hagan et al, Neural Network Design, PWS Publishing Company, 1996

[Han1998] S. Han and K. G. Shin, A primary –Backup Channel to Dependable Real-Time Communication in Multihop Networks, IEEE Transactions on Computers, 47(1), January 1998, 46-61

[Hassan2000] M. Hassan, Performance of TCP/IP over ATM networks, Boston, Mass. : Artech House, 2000

[Hightower2001] J. Hightower and G. Borriello, Location Systems for Ubiquitous Computing, IEEE Computer, 34(8), August 2001, 57-66

[Hurst1965] H.E. Hurst, Long-term Storage : An Experimental Study, London Constable, 1965.

[Ip2001] May T.W. Ip, Wilfred W.K. Lin, Allan K.Y. Wong, Tharam S. Dillon and Dian Hui Wang, An Adaptive Buffer Management Algorithm for Enhancing Dependability and Performance in Mobile-Object-Based Real-time Computing, Proc. of the IEEE ISORC'2001, Magdenburg, Germany, May 2001, 138-144

[Ip2002] Allan K.Y. Wong, May T.W. Ip and Tharam S. Dillon, M^3RT: An Internet End-to-End Performance Measurement Approach for Real-Time Applications with Mobile Agents, Proc. of the ISPAN''002, 2002, 119-124

[Jacobson1988] V. Jacobson, Congestion avoidance and control. ACM SIGCOMM Computer Communication Review, 1988, 18(4), 314 - 329

[Jamjoom2004] H. Jamjoom, P. Pillai and K.G. Shin, Resynchronization and Controllability of Bursty Service Requests, IEEE/ACM Transactions on Networking, 14(4), August 2004, pp. 582- 594

[Jain1992] R. Jain, The Art of Computer Systems Performance Analysis, Techniques for Experimental Design, Measurement, Simulation, and Modeling, Wiley, 1992

[Jalote1994] P. Jalote, Fault Tolerance in Distributed Systems, Prentice Hall, 1994

[Jamjoom2004] H. Jamjoom, P. Pillai and K.G. Shin, Resynchronization and Controllability of Bursty Service Requests, IEEE/ACM Transactions on Networking, 14(4), August 2004, 582- 594

[Kang2002] K.D. Kang, S.H. Son and J.A. Stankovic, STAR: secure real-time transaction processing with timeliness guarantees, the 23rd IEEE real-Time Systems Symposium, 2002(RTSS'2002) 3-5 Dec. 2002, 303 -314

[Karagiannis2003] T. Karagiannis, M. Faloutsos, M. Molle, A User-friendly Self-similarity Analysis Tool, ACM SIGCOMM Computer Communication Review, 33(3), July 2003, 81-93 (http://www.cs.ucr.edu/~tkarag/Selfis/Selfis.html)

[Karray2002] K.D. Kang, S.H. Son and J.A. Stankovic, STAR: secure real-time transaction processing with timeliness guarantees, the 23rd IEEE real-Time Systems Symposium, 2002(RTSS'2002) 3-5 Dec. 2002, 303 -314

[Kim1998] W.J. Kim; B.G. Lee; FRED-fair random early detection algorithm for TCP over ATM networks, Electronics Letters , Volume: 34 Issue: 2 , 22 January 1998, 152 -154

[Koo2001] J. Koo, B.H. Song, K.S. Chung, H.J. Lee and H.K. Kahng, MRED: A New Approach to Random Early Detection, Proceedings of 15th International Conference on Information Networking, 31 January -2 February 2001, 347 -352

[Kramer] Generator of Self-Similar Network Traffic, http://wwwcsif.cs.ucdavis.edu/~kramer/code/trf_gen1.html

[Kratz] M.F. Kratz and S.I. Resnick, The QQ-Estimator and Heavy Tails, http://www.orie.cornell.edu/trlist/trlist.html

[Kris2003] G.P. Krishna, M.J. Pradeep and S.R. Murphy, An Efficient Primary-Segmented Backup Scheme for Dependable Real-Time Communication in Multiphop Networks, IEEE/ACM Transactions on Networking, 11(1), February 2003

[Lakshman1996] T. Lakshman, A. Neidhardt and T. Ott, The Drop from Front Strategy in TCP over ATM and Its Internetworking with other Control Features, INFOCOMM 1996

[Lakshman1997] T. Lakshman U. Madlow, The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss, *IEEE/ACM Transactions on Networking*, 5(3), June 1997, 336-350

[Laprie1995] J.C.Laprie, Dependable Computing: Concepts, Limits, Challenges, the 5th IEEE International Symposium on Fault-Tolerant Computing (FTCS-25), Pasadena, CA, USA, 42-54

[Leland1994] Will Leland, Murad Taqqu, Walter Willinger, and Daniel Wilson, On the Self-Similar Nature of Ethernet Traffic

(Extended Version), IEEE/ACM Transactions on Networking, 2(1), February 1994, 1-15

[Lewandowski1998] S.M. Lewandowski, Frameworks for Component-based Client/Server Computing, ACM Computing Surveys 30(1), March 1998, 3-27

[Lewis1996] T. Lewis, The Next 10,0002 Years: Part 1," IEEE Computer, 29(4), April 1996, pp. 64-70

[Lin2001NNC] W.W.K. Lin, M.T.W. Ip, D.H. Wang, A.K.Y. Wong and T.S. Dillon, A Neural Network Based Proactive Buffer Control Approach for Better Reliability and Performance for Object-Based Internet Applications, Proc. of the PDPTA2001, Las Vegas, USA, June 2001, 451-456

[Lin2002] W.W.K. Lin, An adaptive IEPM (Internet End-to-end Performance Measurement) based approach to enhance fault tolerance and performance in object based distributed computing over a sizeable network (exemplified by the Internet), Master of Philosophy Thesis, Department. of Computing, Hong Kong Polytechnic University, 2002.

[Lin2002FLC] W.W.K. Lin and A.K.Y. Wong, A Fuzzy Controller for Adaptive Buffer Control Leading to Better Channel Reliability and System Performance for Object-based Internet Applications, Proc. of the PDPTA2002, Las Vegas, USA, June 2002, 345-350

[Malla2003] A. Malla, M. El-Kadi, S. Olariu and P. Todorova, A Fair Resource Allocation Protocol for Multimedia Wireless Networks, IEEE Transactions on Parallel and Distributed Systems, 14(1), January 2003, 63 - 71

[Markatos1998] E. P. Markatos, M. G. H. Katevenis, and P. Vatsolaki. The Remote Enqueue Operation on Networks of Workstations. In Proceedings. of the 2nd International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC'98), number 1362 in Lecture Notes in Computer Science, Springer, February 1998, 1-14

[Michalewicz1996] Z. Michalewicz, Genetic algorithms + Data Structures = Evolution Programs, 3rd Ed., Springer, Berlin, 1996

[Mitrani1987] I. Mitrani, Modelling of Computer and Communication Systems, Cambridge University Press, 1987.

[Mishra1998] S. Mishra and L. Wu, An Evaluation of Flow Control in Group Communication. IEEE/ACM Transactions on Networking, 6(5), October 1998, 571-587

[Mitsuru1998] O. Mitsuru and K. Guenter, IBM Aglets Specification, www.trl.ibm.com/aglets/spec11.htm

[Mitchel1999] E. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1999

[Molnár1999] S. Molnár, T.D. Dang and A. Vidács, Heavy-Tailedness, Long-Range Dependence and Self-Similarity in Data Traffic, Proceedings of 7th International Conference on Telecommunication Systems, Modelling and Analysis, March 1999, Nashville, USA, 18-21.

[Morin1997] C. Morin and I. Puaut, A Survey of Recoverable Distributed Shared Virtual Memory Systems, IEEE Transactions on Parallel and Distributed Systems, 8(9), September 1997, 959 - 969

[Mukherjee1998] S.S. Mukherjee and M.D. Hill. The Impact of Data Transfer and Buffering Alternatives on Network Interface Design, In 4th International Symposium on High-Performance Computer Architecture (HPCA), Las Vegas, USA, 1998, 207-218

[Nakamura2004] M. Nakamura, H. Kamezawa, J. Tamatsukuri, M. Inaba, K. Hiraki, K. Mizuguchi, K. Torii, S. Nakano, S. Yoshita, R. Kurusu, M. Sakamoto, Y. Furukawa, T. Yanagisawa, Y. Ikuta, J. Shitami, A. Zinzaki, Long Fat Pipe Congestion Control for Multi-Stream Data Transfer, Proceedings of the 2004 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04), May 10 - 12, 2004, Hong Kong, SAR, China, 294-300

[Nunamaker1991] J. F. Nunamaker, JR., M. Chen and T. D. M. Purdin, System Development in Information Systems Research, Journal of Management Information Systems, 7(3), Winter 1990-91, 89-106

[Oh1998] S.H. Oh, S.Y. Lee, S. Shin and H. Lee, Adaptive Learning Rate and Limited Error Signal for Multilayer Perceptrons with nth Order Cross-Entropy Error, Proc. of the IEEE World Congress

on Computational Intelligence, The 1998 IEEE International Joint Conference on Neural Network, vol. 3, May 1998

[Padhye1998] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, Modeling TCP Throughput: A Simple Model and its Empirical Validation, Proc. of the ACM SIGCOMM '98 Conference, Vancouver, Canada, 1998

[Patterson2003] C.A. Patterson, R.R. Muntz and C.M. Pancake, Challenges in Location-aware Computing, IEEE Pervasive Computing, 2(2), April-June 2003, 80 – 89

[Paxson1995] V. Paxson and S. Floyd, Wide-Area Traffic: The Failure of Poisson Modeling. IEEE/ACM Transactions on Networking, 3(3), June 1995, 226-244.

[Paxson1999] V. Paxson, End-to-end Internet Packet Dynamics, IEEE/ACM Transactions on Networking, 7(3), June 1999, 277 -292

[Pedrycz1997] W. Pedrycz, Computational Intelligence : an Introduction, Boca Raton, Fla., CRC Press, 1997.

[Peitgen2004] H.O.Peitgen, H.Jurgens, D.Saupe, Chaos and Fractals: New Frontiers of Science 2nd edition, Springer, 2004, pp.686

[Philips1987] E.M. Philips and D.S. Pugh, How to Get a PhD: a Handbook for Students and Their Supervisors, Open University Press, 1987

[Prasad2003] R.S. Prasad, M. Murray, C.Dovrolis, K.Claffy, Bandwidth Estimation: Metrics, Measurement Techniques, and Tools, IEEE Network 17(6), Nov.-Dec. 2003, 27-35

[Rama1992] P. Ramanathan and K.G. Shin, Delivery of Time Critical Messages Using a Multiple Copy Approach, ACM Transactions on Computer Systems, 10(2), May 1992, 144-166

[Ramani2000] S. Ramani, B. Dasarathy and K. S. Trivedi: Building a Reliable Message Delivery System Using the CORBA Event Service. IPDPS Workshops, Cancun, Mexico, 2000, 1276-1280

[Ravindran2001] B. Ravindran, P. Kachroo and T. Hegazy, Intelligent Feedback Control-Based Adaptive Resource Management for Asynchronous, Decentralized Real-time Systems, IEEE

Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, 31(2), 2001, 261-265

[Reddy2001] S.A.L.N. Reddy, LRU-RED: an active queue management scheme to contain high bandwidth flows at congested routers, Proceedings of the 2001 IEEE Global Telecommunications Conference, vol. 4 , 25-29 November 2001, 2311 -2315

[Ren2002] F. Ren, Y. Ren and X. Shan, Design of a Fuzzy Controller for Active Queue Management, Computer Communications, v. 25, 2002, 874-883

[Resnick1997] S.I. Resnick, Heavy Tail Modeling and Teletraffic Data, The Annals of Statistics, 25(5), 1997, 1805-1869

[Rumelhart1986] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning Internal Representations by Error Propagation, in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, D.E. Rumelhart and J.L. McClelland, Eds., vol. 1, MIT Press, 1986

[Ryu1996] B.K. Ryu and S.B. Lowen, Point Process approaches to the Modeling and Analysis of Self-similar Traffic I. Model

Construction, Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation vol.3 , 24-28 March 1996, pp.1468 - 1475

[Sarraille] J. Sarraille and P. DiFalco, FD3, http://life.bio.sunysb.edu/morph/fd3.html

[Schmidt1995] D. C. Schmidt and S. Vinoski, Object Interconnections: Comparing Alternative Client- Side Distributed Programming Techniques (Columns 3), SIGS C++ Report Magazine, May, 1995

[Shin2000] K.G. Shin, C.C. Chou and S.K. Kweon, Distributed Route Selection for Establishing Real-Time Channels, IEEE Transactions on Parallel and Distributed Systems, 11(3), March 2000, 318-355

[Sobczak2001] M.Sobczak, the YAMI Project, http://www.maciejsobczak.com/prog/yami/copyright.html, 2001

[Stankovic1998] J.A. Stankovic et al, Deadline Scheduling for Real-Time Systems,

Kluwer Publishers, 1998

[Tanenbaum1996] A.S. Tanenbaum, Computer Networks, 3rd Edition, Prentice Hall, 1996

[Tanenbaum2003] A. S. Tanenbaum, Computer Network, 4th Edition, Prentice Hall, 2003

[Tang2000] D. Tang and M. Baker, Analysis of a Local-area Wireless Network, Proceedings of the 6th annual international conference on Mobile computing and networking, Boston, Massachusetts, United States, pp.1-10http://mosquitonet.stanford.edu/software.html

[Taqqu2003] M.S. Taqqu, Fractional Brownian Motion and Long-Range Dependence, in Theory and Applications of Long-Range Dependence, P. Doukhan et al., Eds., Birkhuser 2003, 5-38

[Trace] http://www4.comp.polyu.edu.hk/~cswklin/research/traces/

[Tsybakov1998] B. Tsybakov and N.D. Georganas, Self-Similar Processes in Communications Networks, IEEE Transactions on Information Theory, 44(5), September 1998, 1713-1725

[VTune2002] Intel's VTune Performance Analyzer, http://ww.intel.com/support/performancetools/vtune/v5

[Wang2004] T. Wang and A. Singh, Communication Using a Reconfigurable and Reliable Transport Layer Protocol, Proceedings of the 2nd International Symposium, ISPA'04, December 2004, Hong Kong, PRC, 788-797

[Weiser1991] M. Weiser, The Computer for the Twenty-First Century, Scientific American, September 1991, 94-104

[Willinger2003] W. Willinger, V. Paxson, R.H. Hiedi and M.S. Taqqu, Long-Range Dependence and Data Network Traffic, in Theory and Applications of Long-Range Dependence, P. Doukhan et al., Eds., Birkhuser 2003, 373-408

[Wong1999A] Allan K.Y. Wong and Tharam S. Dillon, A Fault-Tolerant Data Communication Setup to Improve Reliability and Performance for Internet-Based Distributed Applications, Proc. of the 1999 Pacific Rim International Symposium on Dependable

Computing (PRDC'99), Dec.1999, Hong Kong (SAR), China, 268-275

[Wong1999B] Allan K.Y. Wong and Joseph H.C. Wong, A Convergence Algorithm to Help Enhance the Performance of Distributed Systems on Large Networks, Proc. of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-Span'99), Fremantle Australia, June 1999, 302-307

[Wong2000A] Allan K.Y. Wong and Tharam S. Dillon, A Fault Tolerant Model to Attain Reliability and High Performance for Distributed Computing over the Internet, Journal of Computer Communications, vol. 23, 2000, 1747-1762

[Wong2001] Allan K.Y. Wong, Tharam S. Dillon, Wilfred W.K. Lin and T.W. Ip, M^2RT: A Tool Developed for Predicting the Mean Message Response Time for Internet Channels, The Journal Computer Networks (and ISDN Systems), vol. 36, 2001, 557-577

[WongHC2001] H.C.Wong, May T.W. Ip and Allan K.Y. Wong, An Adaptive and Aggressively Bounded Convergence Algorithm for

Enhancing and Measuring the Performance of Applications Running on Networks with Heavy-Tailed Distributions, Proc. of the 15th International Parallel and Distributed Processing Symposium, San Francisco, April 2001

[Wong2002GAC] Allan K.Y. Wong, W.W.K. Lin, M.T.W. Ip and T.S. Dillon, Genetic Algorithm and PID Control Together for Dynamic Anticipative Marginal Buffer Management: An Effective Approach to Enhance Dependability and Performance for Distributed Mobile Object-based Real-time Computing over the Internet, Journal of Parallel and Distributed Computing, vol. 62, No.9, September 2002, 1433-1453

[Wuytack1999] S. Wuytack, J.L. da Silva, F. Catthoor, G. de Jong, and C. Ykman-Couvreur, Memory Management for Embedded Network Applications, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 18(5), May 1999, 533 –544

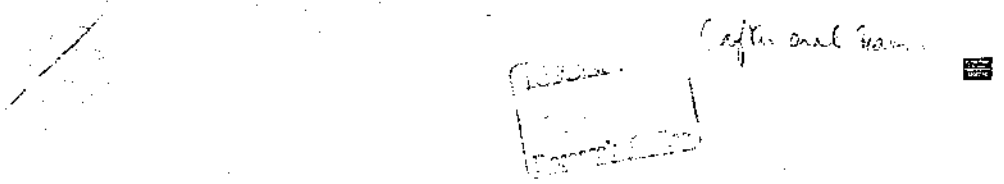
[Yang1995] C. Yang and A.V.S. Reddy, A Taxonomy for Congestion Control Algorithms in Packet Switching Networks, IEEE Network Magazine, v.9, July/August 1995, 34-35

[Zadeh1994] L.A. Zadeh, Fuzzy Logic, Neural Networks, and Soft Computing, Communications of the ACM, 37(3), March 1994, 77-84

[Zhang2003] L.Zhang, V. J. Ribeiro, S. Moon and C. Diot, Small-time scaling behaviours of Internet backbone traffic: an empirical study, Infocomm2003,

[Zheng2001] B. Zheng and M. Atiquzzaman, DSRED: Improving Performance of Active Queue Management over Heterogeneous Networks, Proceedings of the IEEE International Conference on Communications (ICC 2001), vol.8 , 11-14 June 2001, 2375 -2379

APPENDIX I MY MPHIL THESIS COMMENTS BY PROFESSOR LI AS THE EXTERNAL EXAMINER



CONFIDENTIAL

External Examiner's Report

Lin Wan Kei Wilfred (Department of Computing), MPhil candidate
Thesis Title: An Adaptive IEPM (Internet End-to-End Performance Measurement) Based Approach to Enhance Fault Tolerance and Performance in Object Based Distributed Computing over a Sizeable Network (Exemplified by the Internet)

Part A :
Does the thesis submitted by Mr Lin Wan Kei Wilfred form an adequate basis for an oral examination?

(Please tick) : [√] Yes [ ] No

Part B :
The comments below will be released to members of the Board of Examiners before the oral examination. Can they be released to the candidate after the oral examination?

(Please tick) : [√] Yes [ ] No

(My comments can be released to the Board of Examiners and the candidate before/after the oral examination. Therefore, I will combine Parts B and C.)

Significance of the Work: The thesis addresses the issues of fault tolerance and performance of object based distributed computing over a sizable network such as the Internet. These are important problems in realistic Internet Computing, and solutions proposed in this thesis will have practical impact.

Appropriateness of the Methodology Adopted: The approach used in the thesis is to consider the Internet End-to-End Performance Measure (IEPM) of logical channels between clients and servers and to focus on the problem of preventing and eliminating of buffer overflow at the receiver side. The reduction of buffer overflow eventually enhances the capability of fault tolerance and improves response time. The above approach has been proven to be effective.

Originality of the Work: The main contributions of the thesis is to propose and evaluate the performance of three novel adaptive buffer overflow controllers: GAC (genetic algorithm controller), FLC (fuzzy logic controller), and NNC (neural network controller). These adaptive controllers aim to eliminate buffer overflow without locking and wasting too much memory space. The work is original. In fact, there has not been much research performed for queue buffer management in the Internet environment and existing strategies have their limitations in handling unpredictable service requests with serious perturbations.

CONFIDENTIAL

Lin Wan Kei Wilfred (COMP), MPhil candidate, Page 2

Part C :

Comments to be released to the candidate before the oral examination:

Quality of Presentation: The thesis is well written and organized. There can be some improvements. For example, the figures in Chapter 4 (Figs. 4.2, 4.3, 4.7, 4.8, 4.10, 4.12, 4.14) should be in a unified style.

Accuracy and Quality of Results: By extensive experimentation, it was shown in the thesis that compared with the pure PID controller, the three adaptive buffer overflow controllers, which are able to change the PID parameters dynamically to fit random traffic situations on the Internet, do exhibit better buffer length.

Appropriateness of the Conclusions: The thesis successfully proposed and verified three novel controllers that adaptively prevent buffer overflow by maintaining a user-specified safety margin and ensure that the buffer length always covers the queue length.

Adequacy of References: The list of references is adequate, which demonstrates the candidate's knowledge in this field.

Summary: I am very impressed that the work reported in this thesis are based on 4 journal articles and 8 conference papers of which the candidate is a co-author. The thesis is definitely of fine quality and I congratulate the candidate for his excellent research.

Signature : _____

Name : Prof. Keqin LI

Institution : State University of New York at New Paltz

Date : Oct. 3, 2002

APPENDIX II THE CONVERGENCE ALGORITHM

The CA (Convergence Algorithm) is an IEPM (*Internet En-to-End Performance Measurement*) technique [Cottrel1999]. It can estimate the mean service roundtrip time (RTT) of a logical channel quickly and accurately. The Java-based CA prototype: M^{2}RT was verified and validated as a *macro* tool [Wong2001]. In its *macro* form the tool must be installed at the two nodes that represent the ends of the logical channel. *Micro* IEPM tools differ by operating as a logical entity to be invoked anytime and anywhere for service by message passing. The M^{2}RT (*Mean Message Response Time*) experience led to the development of the Java-base *micro* CA (MCA) prototype: M^{3}RT (*Micro Mean Message Response Time*) [Ip2002]. The CA operation treats a traffic pattern simply as a waveform. Its speed and accuracy does not depend on to the pattern being worked on because it is based on the central limit theorem. The

$$M_i = \frac{M_{i-1} + \sum_{j=1}^{j=F-1} m_j^i}{F} \ldots\ldots\ldots (A.1); M_0 = m_{j=0}^{i=1} \ldots\ldots\ldots (A.2); i \geq 1$$

MCA version supporting the PIDC and the NNC is modified from the M^{3}RT object class. The CA operation is summarized by the equations: (A.1) and (A.2), where M_i is the distribution mean estimated for the time in which the *F* (*flush limit*) number of data samples is collected. The previous experience shows that F=14 yields the fastest convergence to the estimated mean [Wong2001]. The other parameters are: a) M_{i-1} is the feedback of the last estimated mean to the current estimation cycle, b) m_j^i is the j^{th} sample in the i^{th} M_i estimation cycle, $j = 1,2,3,(F-1)$, and c) M_0 is the first

data sample when the MCA had first started. Figure FA1 shows the M_i predicted by MCA over time, and the RTT trace is for the TCP channel between the Hong Kong PolyU and the LaTrobe University site in Australia. In this case M_i always settles to the value of 480ms in the steady state.

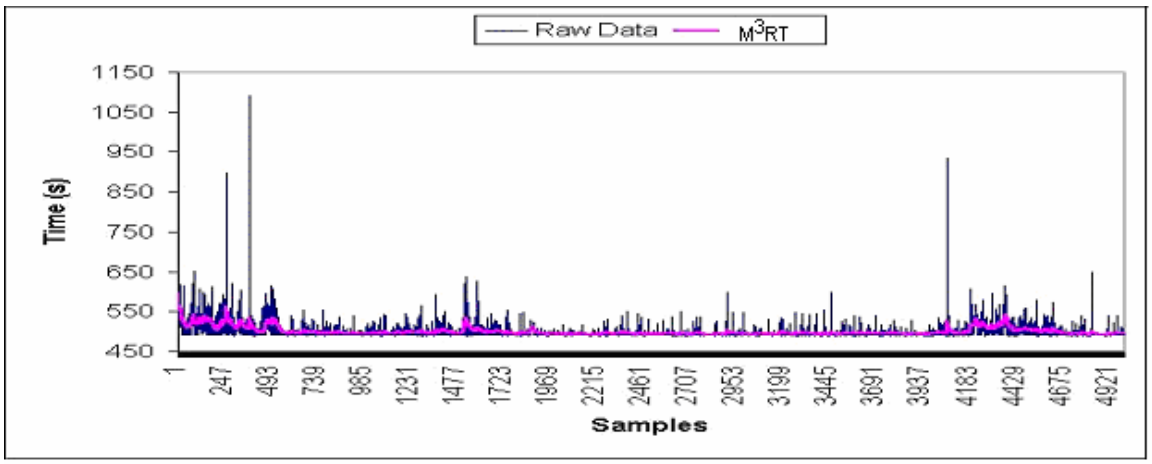


Figure FA1. The M_i prediction by M^3RT for the "Hong Kong PolyU - LaTrobe" TCP channel

APPENDIX III E-MAILS OF ACCEPTANCE (JOURNALS)

To: tharam@
From: cwhite@
Date: 11 06 2004
CC: jerrilyn@

Subject: SMCC-03-06-0053.R3

Prof. Chelsea C. White III, Editor
School of Industrial & Systems Engineering
Georgia Institute of Technology
765 Ferst Drive
Atlanta, Georgia 30332-0205 U.S.A.

Dear Dr. Tharam Dillon,

It is a pleasure to inform you that your manuscript, "Application of Soft Computing Techniques to Adaptive User Buffer Overflow Control on the Internet,SMCC-03-06-0053.R3, has been accepted for publication as a Regular Paper in the IEEE Transactions on Systems, Man, and Cybernetics: Part C. If there are any final comments from the reviewers or the editor, they can be found at the bottom of this letter.

Your paper will be published in the print edition of the Transactions when the next available issue is complete or as space permits. However, it will be published in the "papers accepted for future publication" section of the SMCC IEEE Xplore edition very soon after the proofs are returned.

Please prepare and send your final manuscript to me at the above address at your earliest convenience. We will need three hard copies of the text of your paper, a set of original camera-ready illustrations (one per page) for any illustrations and tables included in the manuscript and a separate list of captions for the illustrations and tables. All of this should be prepared in IEEE style. We will also need bio-sketches and photos of each of the authors.

We also require a floppy disk or CD that includes your manuscript text, figure captions and bio skretch. Pay particular attention to the need for your hard paper copy and electronic copy to be identical. Do not use Acrobat pdf (.pdf) or postscript (.ps and .eps) formats for text disks or files. If you include a graphics disk or files for illustrations, only TIFF (.tif), Postscript (.ps) and Encapsulated Postscript (.eps) formats are acceptable. Be sure to label disks with the operating system, word processing and graphics formats used.

You must submit a completed a copy of the IEEE Copyright form electronically via SMCC Manuscript Central at http://smcc-ieee.manuscriptcentral.com. Please click on the "Copyright

Submission" button in the "Submitted Manuscripts" chart in your Author Center to complete and send the form.

A Completed copy of the Final Manuscript Checklist must accompany the final manuscript materials. You can find this form by clicking on the link titled information for authors of accepted papers on the IEEE Transactions on Systems, Man, and Cybernetics: Part C website at http://www.ieee-smc.org/webpages/publications/index.html. The link also contains instructions for Preparing the Final Manuscript, instructions for labeling the disks or CDs, and links to the IEEE Tools for Authors and other important information for preparing your final manuscript materials. If your computer does not allow you to reach the website from the link, please copy the web address and paste into your browser's home page.

Be sure that you understand the rules governing over length page charges discussed in the Instructions for Authors which can also be found on the Transactions website.

Please reference the manuscript id number when submitting your manuscript. We congratulate you on acceptance of your manuscript for publication as a Regular Paper and look forward to receiving the needed publication material.

Sincerely,

Prof. Chelsea White
Editor-in-Chief
IEEE Transactions on Systems, Man, and Cybernetics: Part C

From: Maggie Bond <mab1@ >
To: <cswklin@ >
Date: 2005/2/22 下午 05:27

Subject: CON 127 - Acceptance for Publication

Dear Wilfred
Re: CON 127 - Applying Fuzzy Logic and Genetic Algorithms to Enhance the Efficacy of the PID Controller in Buffer Overflow Elimination for Better Channel Response Timeliness over the Internet I am pleased to inform you that the above paper has been accepted for publication in Concurrency and Computation: Practice and Experience. I will be sending the manuscript and Copyright Transfer Agreement to John Wiley & Sons, the publishers today. Please deal with them direct if you have any further queries.

Best regards.
Yours sincerely
Maggie Bond
for Professor Tony Hey, CBE - Editor

Maggie Bond, Group Secretary
Declarative Systems and Software Engineering Research Group
School of Electronics and Computer Science
Zepler Building, Room 3215
University of Southampton
Highfield, Southampton SO17 1BJ
Tel: + 44 23 8059 4506
Fax: + 44 23 8059 3045
E-mail: mab1@

Date: Tue, 18 Jan 2005 16:00:35 -0600
From: Springer Alerting <alerts@ >
To: cswklin@
Subject: Springer Author Alerting

Dear Dr. Wilfred W. K. Lin:

We are very pleased to be the first to congratulate you on the electronic publication of your article "A Novel Fuzzy-PID Dynamic Buffer Tuning Model to Eliminate Overflow and Shorten the End-to-End Roundtrip Time for TCP Channels" published in Lecture Notes in Computer Science. If your institution has access to this journal, you may view your paper at: http://www.springerlink.com/index/VF155CH38XFLLB4H (you may need to copy and paste the URL into your browser).

We encourage you to forward this email to colleagues who may be interested in your work, in this topic, or in research being done in this area.

We thank you for your contribution and hope that your experience publishing with Springer was a satisfying one. We look forward to your future contributions and invite you to visit our website at http://www.springeronline.com/authors, where you will find valuable services.

Regards,
Springer Author Services

Please be aware that this e-mail has been prompted by your paper appearing in electronic Online First format and/or paginated pdf-format. The paper copy of the journal follows approximately 30 days after it is first published online, as will any offprints that are due to you.

This is an automated e-mail; please do not reply to this account. If you have a query about your paper please contact the publishing editor of the journal or use our author online form <http://www.springeronline.com/sgw/cda/frontpage/0,11855,5-111-2-71152-0,00.html> .

Springer Heidelberg
TiergartenstraAYe 17
69121 Heidelberg
Germany
Tel.: +49 6221 487 0
Fax: +49 6221 487 8366

Springer New York
233 Spring Street
New York, NY 10013
USA
212-460-1500
Tel.: 800-SPRINGER
Fax: 201 348 4505

APPENDIX IV BEST PRESENTATION AWARD (INDIN2005 CONFERENCE)

IEEE

Department of Industry and Resources

3rd International IEEE Conference on Industrial Informatics

INDIN'05

Frontier Technologies for the Future of Industry and Business

http://www.indin2005.curtin.edu.au

10-12 August 2005
Perth, Western Australia

Congratulations on being awarded a:

Best Presentation Award

Awarded to: Wilfred Lin

For: An internet-based distributed manufacturing system utilising a recurrent neural network controller for dynamic buffer size tuning to prevent buffer overflow and shorten roundtrip time

Track: Distributed Collaborative Systems

On behalf of the INDIN 2005 Conference Program Committee.

Professor Elizabeth Chang

Curtin University of Technology

RMIT UNIVERSITY

UTS