A Novel Fuzzy Logic Controller (FLC) for Shortening the TCP Channel Roundtrip Time by Eliminating User Buffer Overflow Adaptively

[1]Wilfred W. K. Lin, [1]Allan K. Y. Wong, and [2]Tharam S. Dillon
[1]Department of Computing, Hong Kong Polytechnic University, Hong Kong SAR,
Emails: cswklin@comp.polyu.edu.hk, csalwong@comp.polyu.edu.hk
[2]Faculty of Information Technology, University of Technology, Sydney Broadway, N.S.W. 2000,
Email address: tharam@it.uts.edu.au

Abstract
The proposed Fuzzy Logic Controller (FLC) is a novel approach for dynamic buffer tuning at the user/server level. It eliminates buffer overflow by ensuring that the buffer length always cover the queue length adaptively. The FLC and the AQM (active queue management) mechanisms at the router/system level together form a unified solution to stifle TCP (*Transmission Control Protocol*) channel buffer overflow over the Internet. The FLC contributes to: a) prevent the AQM resources dished out at the system level from being wasted, b) shorten the service roundtrip time (RTT) by reducing retransmission, and c) alleviate network congestion in the process. Combining fuzzy logic and the conventional PIDC(Proportional + Derivative + Integral Controller) model creates the FLC that operates with the $\{0,\Delta\}^2$ objective function. The fuzzy logic maintains the given Δ safety margin about the reference point, symbolically represented by "0" in $\{0,\Delta\}^2$. The FLC stability and precision is independent of the traffic pattern changes because of its statistical nature. This makes the FLC buffer overflow controller/tuner suitable for applications over the Internet, where the traffic can changes suddenly, for example, from LRD (*long-range dependence*) to SRD (*short-range dependence*) or multifractal.

Keywords: FLC, PIDC, active queue management, buffer overflow, dynamic buffer tuning, traffic pattern, Internet

1 Introduction
Distributed applications running on the Internet are naturally object-based. They gain computation speedup from the intrinsic distributed parallelism of the underlying network. The component logical objects in these applications usually collaborate in a client/server relationship (Lewandowski 1998) also known as *asymmetric rendezvous* (*one-server-to-many-clients*), as shown in Figure 1. TCP/IP (*Transport Control Protocol/Internet Protocol*) channels are usually plagued by many different faults and errors because of the Internet's sheer size and heterogeneity. Routing a packet through a TCP channel physically means traversing different links and nodes of varying quality and capacity. If ρ is the collective TCP/IP channel error probability, then the *average number of trials* (ANT) to transmit a message successfully is defined by $ANT = \sum_{j=1}^{N\to\infty} jP_j$. If $P_j = \rho^{j-1}(1-\rho)$ is the probability to have the transmission success at the j^{th} trial, then $ANT = \sum_{j=1}^{N\to\infty} j[\rho^{j-1}(1-\rho)]$ can be simplified as $ANT \approx 1/(1-\rho)$. Obviously a smaller ρ yields a shorter RTT that enhances the success of running certain types of time-critical applications (e.g. the *soft real-time* type (Stankovic 1998)) over the Internet. RTT is the time for sending a request and receiving the result correctly.

Buffer overflow, which is one of the many channel faults included in ρ, can cause widespread retransmission, long RTT, and network congestion. Therefore, it makes sense to prevent it from occurring at the both system/router and user/server levels. The conventional way to prevent system-level overflow is throttling first followed by message dropping as the final solution. Firstly, the router throttles the sender to lower its transmission rate voluntarily (Tanenbaum 1996). Secondly, if throttling fails, then the router drops the new incoming messages, for example by using the "*drop in front*" strategy. Message dropping as the ultimate solution is deleterious because while it reduces overflow it increases the risk of network congestion.

Recently the IETF (*Internet Engineering Task Force*) proposed *active queue management* (AQM) for more systematic throttling and message dropping (Braden, Clark, Crowcroft, Davie, Deering, Estrin, Floyd, Jacobson, Minshall, Partridge, Peterson, Ramakrishnan,

Shenker, Wroclawski, Zhang 1998). The concept is to establish probabilistic markings so that router can decide when and what messages should be dropped. The algorithmic RED (*Random Early Detection* (Floyd, Jacobson 1993)) approach, which works in this way, is the AQM candidate proposed in RFC2309 for preventing TCP end-to-end congestion. Different analyses of the RED, however, confirmed its instability. This led to the developments of different enhanced RED versions, including FRED (*Fair RED*), WRED (*Weighted RED*) (Bodin, Schelen, Pink 2000) and DWRED (*Distributed WRED*), which is deployed in the Cisco 7500 series routers (DWRED). Meanwhile, the quest for intelligent AQM algorithms has begun, as exemplified by the experimental Fuzzy-PI model (Ren, Ren, Shan 2002). AQM mechanisms, however, cannot prevent user-level buffer overflow. The problem is the unpredictable request arrival rate and the unknown traffic pattern embedded in the merged traffic (marked by "+" in Figure 1) of the asymmetric rendezvous (Wong, Dillon 1999). In fact, previous experience has indicated that any overflow controller, which is based on a specific distribution (e.g. Poisson), will fail over the Internet (Paxson, Floyd 1995) that follows the power law (Medina, Matta, Byers 2000). The Internet traffic can change its pattern suddenly, for example, from LRD (*long-range dependence*) to SRD (*short-range dependence*) or multifractal.

In the practical sense, it is illogical for a user-level receiver to discard new requests in order to prevent local buffer overflow. Such an act not only increases retransmission and network congestion, but also wastes the AQM effort already dished out by the system. It is sensible therefore to install a user-level dynamic buffer tuner/controller that auto-tunes the buffer size so that it always covers the queue length (Wong, Dillon 1999). User-level auto-tuning and system-level AQM together provide a unified solution for stifling the chance of TCP buffer overflow.

The "P+D" is one of the earliest user-level dynamic buffer tuners. It uses the *proportional* (P) (i.e. the current "*queue length over buffer length (QOB)*" ratio) and the *derivative* (D) (i.e. the current rate of change dQ/dt in the queue length (Q)) control elements to eliminate buffer overflow. This model worked well in simulations but failed frequently in real-life applications. The cause of failure is the unrealistic expectation of using a static set of control parameters to cover the whole spectrum of channel and buffer dynamics. The quest for a better user-level overflow controller led to the proposal of the PID controller (PIDC), which incorporates integral (I) control to enhance the anticipative power of the "P+D". The PIDC effectively eliminates user-level overflow (Ip, Lin, Wong, Dillon, Wang 2001), even though it has two distinctive shortcomings. The desire to eliminate these shortcomings and preserve the PIDC merits at the same time motivates the FLC research.

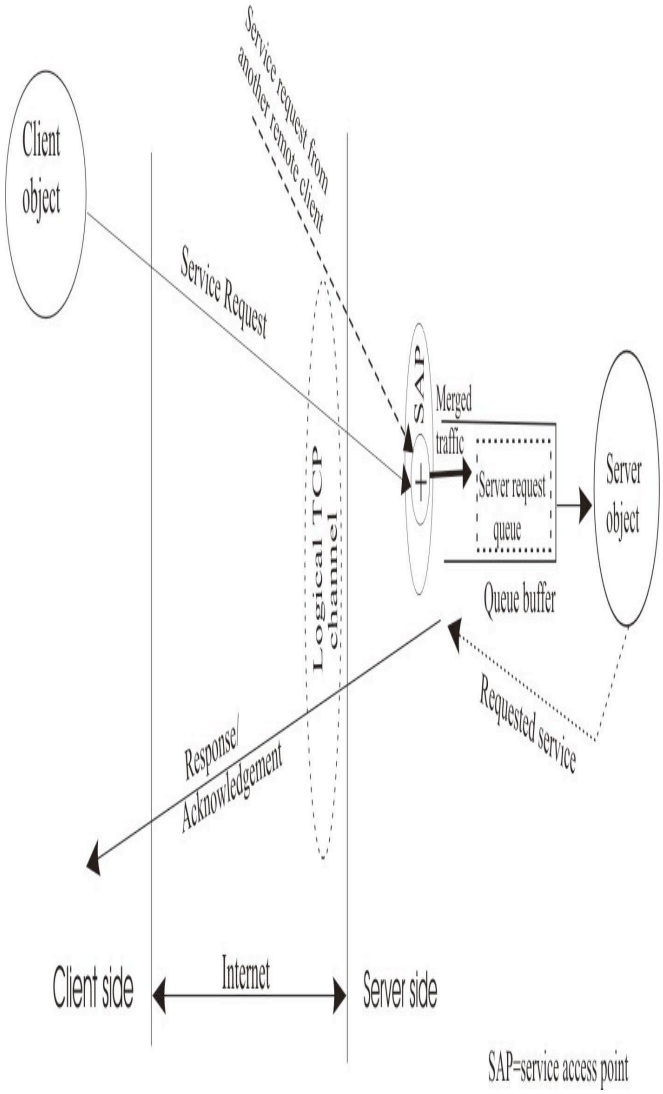

Figure 1. Asymmetric rendezvous as the client/server relationship

2 Related Work

From literature two basic types of buffer overflow controllers can be identified:

a) *FBL (fixed buffer length)*: The FBL models drop the incoming messages as the ultimate solution (Lakeshman, Madlow 1997), and therefore they are naturally deleterious. All the known AQM approaches are of the FBL type, for example, RED, FRED, WRED and Fuzzy-PI (Braden Clark, Crowcroft, Davie, Deering, Estrin, Floyd, Jacobson, Minshall, Partridge, Peterson, Ramakrishnan, Shenker, Wroclawski, Zhang 1998, Ren, Ren, Shan 2002).

b) *VBL (variable buffer length)*: The VBL models try to adaptively tune the buffer length so that it always covers the queue length to stifle any chance of overflow, as exemplified by the "P+D" and PIDC dynamic buffer tuners.

The working mode of a buffer overflow controller can be one of the following:

a) *Algorithmic*: They are based on mathematical models and usually their control parameters remain unchanged at runtime (Karray Gueaieb, Al-Sharhan 2002), for example, the PIDC.

b) *Intelligent/Expert*: They use soft computing techniques (e.g. genetic algorithms, fuzzy logic, and neural network) to tune their control parameters in a dynamic manner (Ren, Ren, Shan 2002).

The novel expert *Fuzzy Logic Controller* (FLC) uses fuzzy logic to eliminate the PIDC shortcomings and preserve its power. It is conceptually the "*fuzzy logic + PIDC*" combination. Point F in Figure 2 illustrates how the PIDC eliminates the overflow under the "P+D" control. The plot is based on the RTT trace from the TCP channel interconnecting the Hong Kong PolyU and the remote LaTrobe University site in Australia. It clearly shows the PIDC shortcomings as follows:

a) *Memory wastage*: The buffer size tends to stay at the high value after each correction by locking up unused memory even when it is no longer needed.

b) *No safety margin*: The PIDC has no safety/tolerance margin to prevent the queue length from getting dangerously close to the buffer length to cause overflow under serious perturbations.

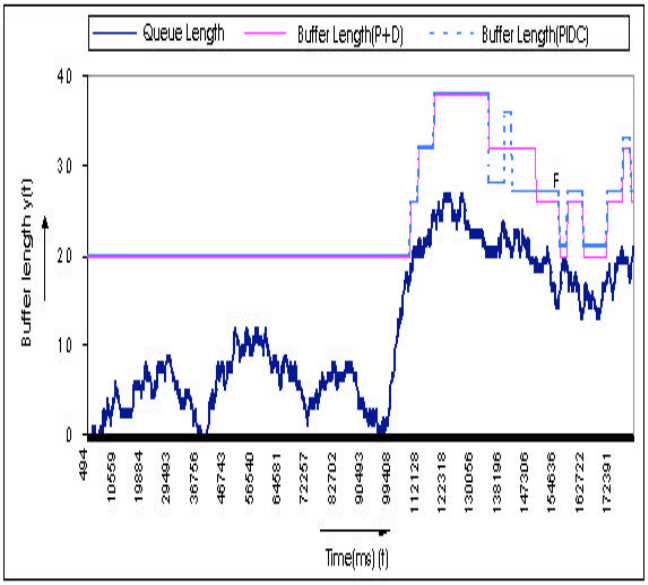

Figure 2. Comparing PIDC and "P+D" performances with the same RTT trace

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 3. The basic PID controller (PIDC) algorithm

The pseudocode in Figure 3 abstracts the PIDC control mechanism, which works with the following parameters: a) dQ/dt which is the rate of change in the queue length Q for derivate control, b) QOB_i which is the ratio of "*queue length over buffer length*" at the i^{th} control cycle for proportional control, c) ICM (*Integral Control Mechanism*) which is the I control, d) L_{now} as the current buffer length, e) $L_{minimum}$ as the minimum queue length estimated from past experience, and e) the chosen thresholds.

3 The FLC controller

The FLC uses fuzzy logic to fine tune the control process of its PIDC component. The fuzzy logic divides the PIDC control domain into a set of smaller fuzzy control regions and mans each with a specific fuzzy rule or a "*don't care*" state. The fuzzy rules adaptively maintain the given Δ safety margin about the $\{0, \Delta\}^2$ objective function's reference point, symbolically marked by "0". For the FLC prototypes the references are the different chosen QOB ratios (i.e. QOB_R values). If the QOB_R is 0.8 then Δ is 0.2 so that the FLC should operate in the QOB range between 0.6 and 1.2. The FLC maintains Δ by tuning the ICM value with respect to the current Q and (dQ/dt) measurements. The fuzzy rules decide how the ICM should be tuned under different conditions. If the FLC control process enters an inert "*don't care*" state, no computation is required. In this way inert states offset the FLC computational complexity and shorten the control cycle time. For this reason the cycle time of the more complex FLC is comparable to that of the PIDC. Figure 4a shows the matrix of fuzzy regions for the experimental FLC[4x6] design. The "*dot*" defines the QOB_R value of 0.8 (80%), and X marks a "*don't care*" state.

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

Figure 4a. An FLC[4x6] design/configuration example

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 control decision, which depends on the current QOB_i and dQ/dt values, may be *Addition (buffer elongation) or* "+", *Subtraction (buffer shrinkage) or* "-", and *don't care*. The FLC[4x6] prototype is supported by the following fuzzy rules (L_{new} and L_{old} denote the

adjusted buffer length and the buffer length before tuning respectively):

Rule 1: If (QOB_i is ML) AND (dQ/dt is NL) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 2: If (QOB_i is ML) AND (dQ/dt is NM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 3: If (QOB_i is ML) AND (dQ/dt is NS) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 4: If (QOB_i is ML) AND (dQ/dt is PS) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 5: If (QOB_i is ML) AND (dQ/dt is PM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 6: If (QOB_i is ML) AND (dQ/dt is PL) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 7: If (QOB_i is L) AND (dQ/dt is NL) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 8: If (QOB_i is L) AND (dQ/dt is NM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 9: If (QOB_i is L) AND (dQ/dt is NS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$
Rule 10: If (QOB_i is L) AND (dQ/dt is PS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$
Rule 11: If (QOB_i is L) AND (dQ/dt is PM) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$
Rule 12: If (QOB_i is L) AND (dQ/dt is PL) Then Action is "+" (Addition) AND $L_{new} = L_{old} + ICM$
Rule 13: If (QOB_i is G) AND (dQ/dt is NL) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 14: If (QOB_i is G) AND (dQ/dt is NM) Then Action is "-"(Subtraction) AND $L_{new} = L_{old}$ - ICM
Rule 15: If (QOB_i is G) AND (dQ/dt is NS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$
Rule 16: If (QOB_i is G) AND (dQ/dt is PS) Then Action is "X"(Don't care) AND $L_{new} = L_{old}$
Rule 17: If (QOB_i is G) AND (dQ/dt is PM) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$
Rule 18: If (QOB_i is G) AND (dQ/dt is PL) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$
Rule 19: If (QOB_i is MG) AND (dQ/dt is NL) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$
Rule 20: If (QOB_i is MG) AND (dQ/dt is NM) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$
Rule 21: If (QOB_i is MG) AND (dQ/dt is NS) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$
Rule 22: If (QOB_i is MG) AND (dQ/dt is PS) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$
Rule 23: If (QOB_i is MG) AND (dQ/dt is PM) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$
Rule 24: If (QOB_i is MG) AND (dQ/dt is PL) Then Action is "+"(Addition) AND $L_{new} = L_{old} + ICM$

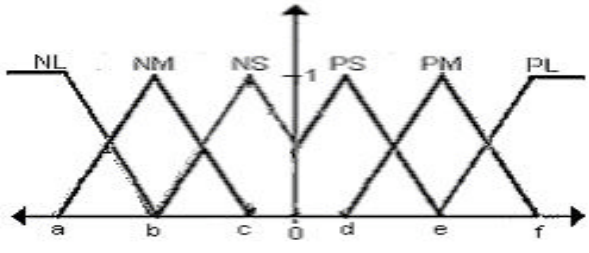

Figure 4b: Membership function for dQ/dt

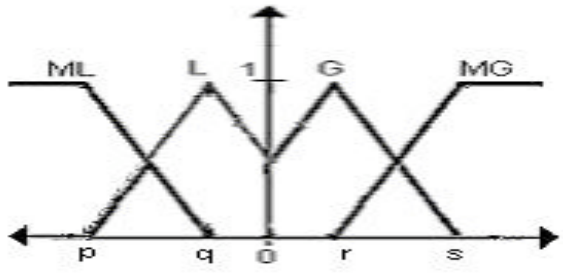

Figure 4c. Membership function for QOB

The dQ/dt and QOB membership functions for the FLC[4x6] design are shown by Figure 4b and 4c respectively. The y-axis of Figure 4b is the degree of membership measurement, and 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 4c 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 *p*, *q*, *r* and *s* are respectively 0.7, 0.75, 0.85 and 0.9. The current dQ/dt and QOB values decides which fuzzy region that the FLC should operate with. 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 *p* and *q*, four fuzzy regions are possible: [ML,NM], [ML,NS], [L,NM] and [L,NS]. They are the -,-,- and X operations shown in Figure 4a. The majority rule selects the minus (-) or buffer shrinkage operation as the final decision. In the case of a draw, for example, -, -, X and X, then the current operation prevails.

4 Experimental Results

The FLC prototypes of different designs were verified by simulations on the Aglets mobile agent platform (Mitsuru, Guenter, Kouichi 1998), which is chosen because: a) it is stable, b) it is rich in user experience, and c) it makes the experimental scalable for the open Internet. The set up for the experiments is shown in Figure 5, in which the driver and the server are aglets (*agile applets*) that collaborate in as client/server relationship within a single computer. The driver picks a known waveform (e.g. Poisson) or trace, which embeds an unknown waveform, from the table. It uses the pick to generate the inter-arrival times for the simulated merged traffic into the server buffer. A "trace" is a file of pre-collected RTT for a TCP channel (e.g. between Hong Kong PolyU and the LaTrobe University in Australia). The use of traces in simulations helps confirm that the FLC control precision and stability is indeed independent of sudden Internet traffic pattern changes. This confirmation is necessary because the real-life Internet traffic can change without warning, for example, from LRD (*long-range dependence* such as *self-similar*) to SRD (*short-range dependence* such as *Poisson*).

The execution or control cycle time (CCT) of the FLC can be measured accurately with Intel's *VTune Performance Analyser* (Vtune). The measurement is in the number of neutral clock cycles N_{CCT}, which can be easily converted into the physical time for any specific platform. For example, if a platform operates at the speed of MHz the physical time T_{CCT} is $P_{CCT} = {N_{CCT}}/{MHz}$. The symbols Q and B in Figure 5 indicate the current queue length and buffer length respectively.

The waveforms in the experiments are always checked and identified, as indicated by the "*traffic pattern analysis*" box in Figure 5. In this way the response of the FLC to any specific waveform can be visualized. The waveform checking and identification is achieved by using the *Selfis Tool* (Karagiannis , Faloutsos, Molle 2003), which includes different estimators. The R/S (*rescaled adjusted statistics*) and Periodogram estimators can identify the LRD character by computing the Hurst (H) effect/value. The H value differentiates LRD (for $0.5 < H \le 1$) from SRD (for $0 < H \le 0.5$). After the LRD character is confirmed, the *modified QQ-plot* filter can be used to check if it is heavy-tailed

From the preliminary experimental results 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 user-level overflow efficaciously, c) it has a shorter CCT (255 clock cycles) than the PIDC (432 clock cycles), and d) the fuzzy logic successfully eliminates the PIDC shortcomings. The "*buffer overflow controller/tuner*" remark in Figure 5 indicates the place where the specific controller can be installed (e.g. FLC or PIDC) for a particular simulation.

.

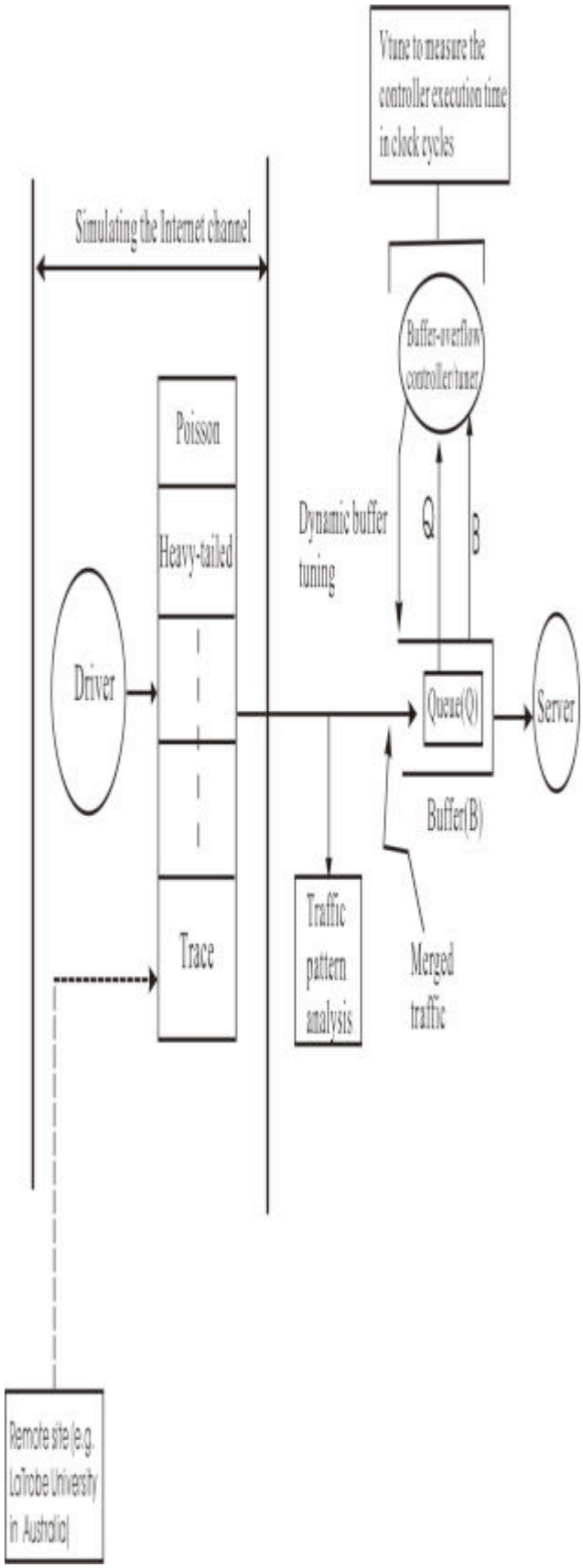

Figure 5. The FLC verification environment

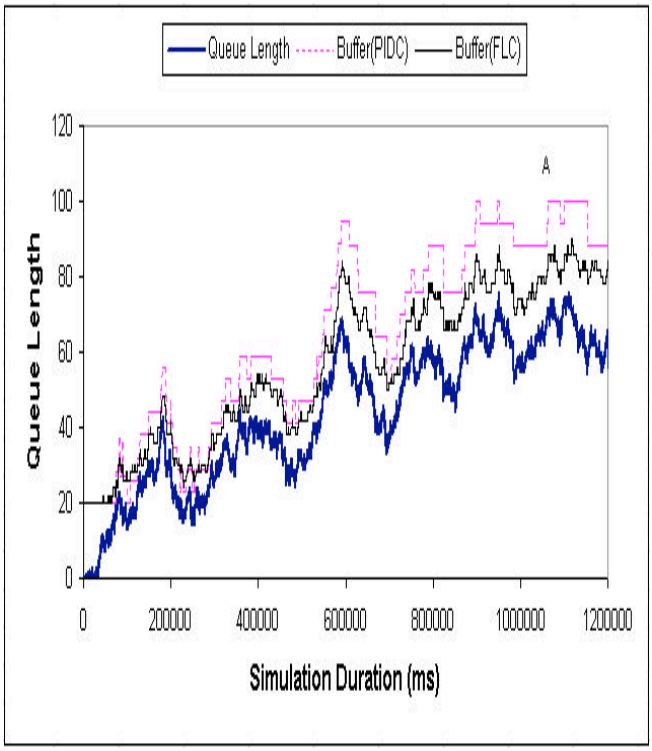

Figure 6. FLC and PIDC response to the RTT trace for the "LaTrobe/PolyU" TCP channel

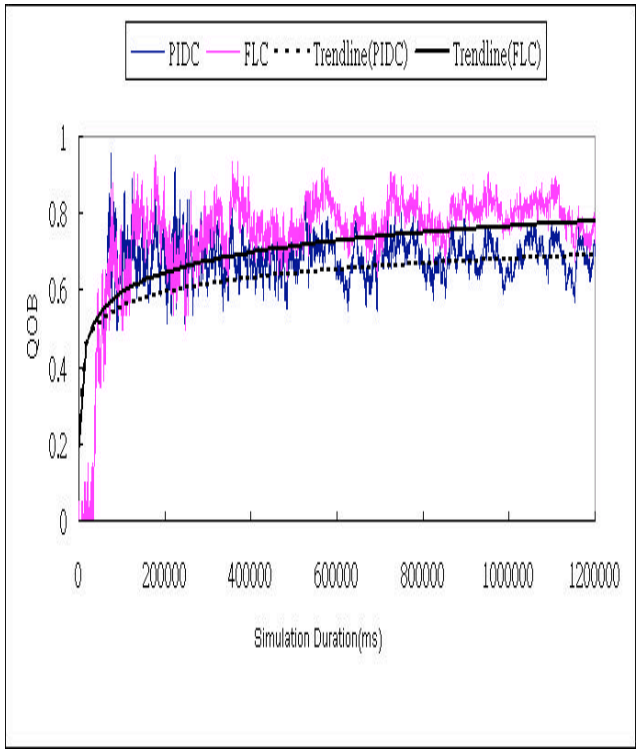

Figure 7. More accurate and faster FLC trend line than the PIDC one

Figure 6 is plotted with the RTT trace for the TCP channel interconnecting the LaTrobe and the Hong Kong PolyU sites. Point A shows that the FLC has successfully eliminated the PIDC shortcomings by maintaining $\Delta = 0.2$ consistently. Figure 7 is for the same trace and shows that the FLC converges to the QOB_R reference more quickly and accurately than the PIDC. This is clearly indicated by the two different trend lines. The R/S plot for this trace yields H=0.518 with 83.05% confidence of its LRD character. The *modified QQ-plot* in Figure 9 indicates that the LRD has a strong heavy-tailed likelihood with $\alpha = 0.46$ because of the reasonable coefficient of determination, namely, $R^2 = 0.9231$. A distribution F is heavy-tailed if and only if (1-F) is varying regularly with index α, for $\lim_{t\to\infty} (1-F(tx))/(1-F(t)) = x^{-\alpha}, x>0.$ A high *coefficient of determination* (e.g. $R^2 = 0.9231$ in Figure 9) indicates an accurate regression (Jain 1992).

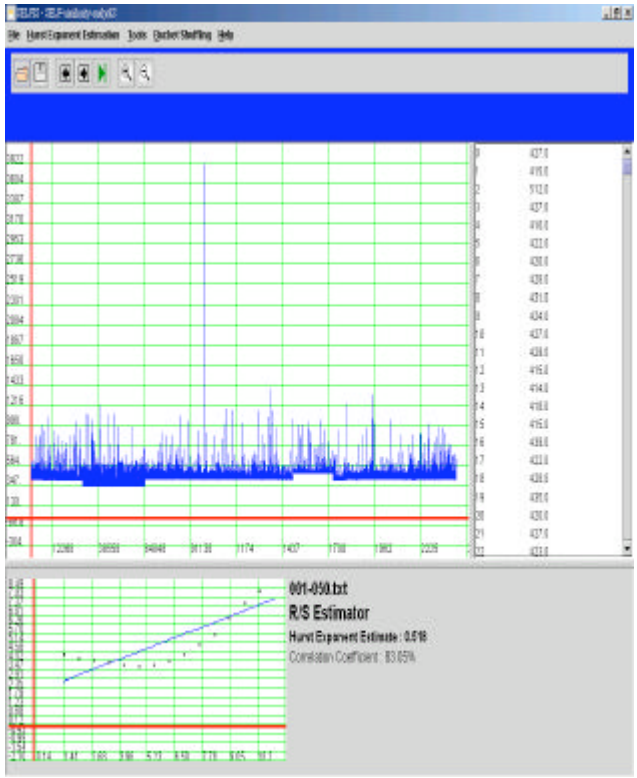

Figure 8. Trace (for Figure 6 and 7) analysis with the Selfis Tool

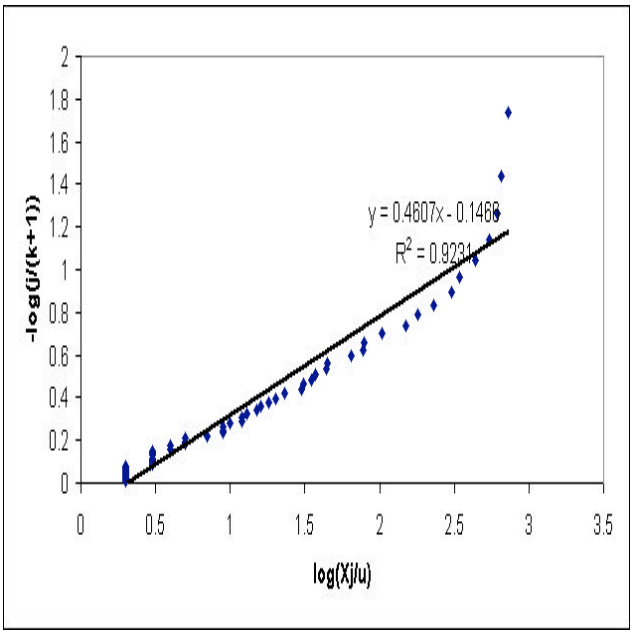

Figure 9. Modified QQ-plot for the LRD trace used in Figure 6, 7 and 8; $\alpha = 0.46$

The rationale of the *modified QQ-plot* 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, higher the better. The *modified QQ-plot* is one of the many tools that can identify the heavy-tailed character. It was chosen for the experiments because other tools such as the *Hill Estimator* and the *De Haan's Moment* method (Resnick 97) are relatively more difficult to use.

4.1 Self-similar Traffic

Self-similar traffic contains bursts, and if the mean inter-arrival time: IAT_{burst} of a burst is shorter than the FLC control cycle time, then the controller becomes inaccurate. That is why it is important for the FLC to respond correctly to self-similar traffic, which is widespread in the open Internet. For the relevant experiments the self-similar traffic patterns are generated with the tool proposed by Glen Kramer (Kramer 2000). Figure 10 is produced by using one of the self-similar patterns generated by this tool. In this case the R/S plot confirms the LRD character with H=0.627 and 98.61% confidence. The Periodogram estimator in Figure 11 also produces the same LRD confirmation with H=0.758. In reality the R/S plot and the Periodogram estimator do not yield the same H value. Figure 12 shows that the FLC has eliminated buffer overflow successfully for the self-similar traffic used in Figure 10 and 11.

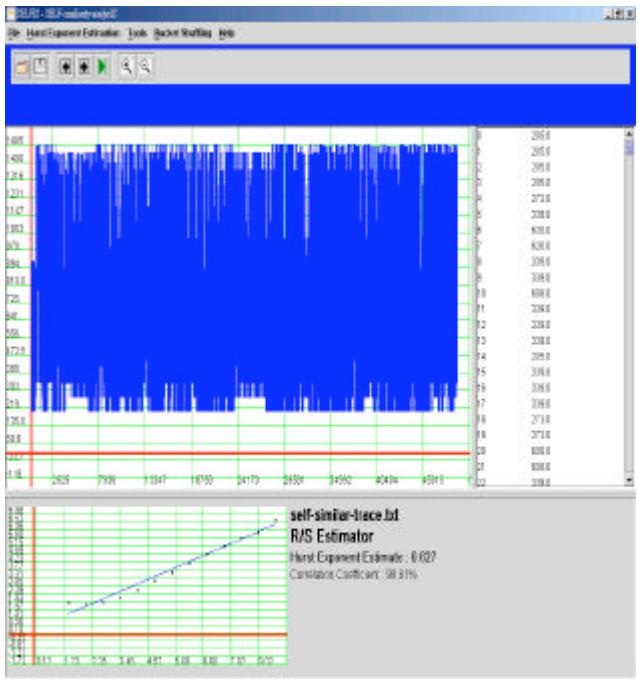

Figure 10. Trace analysis/identification with Selfis (R/S estimator invoked)

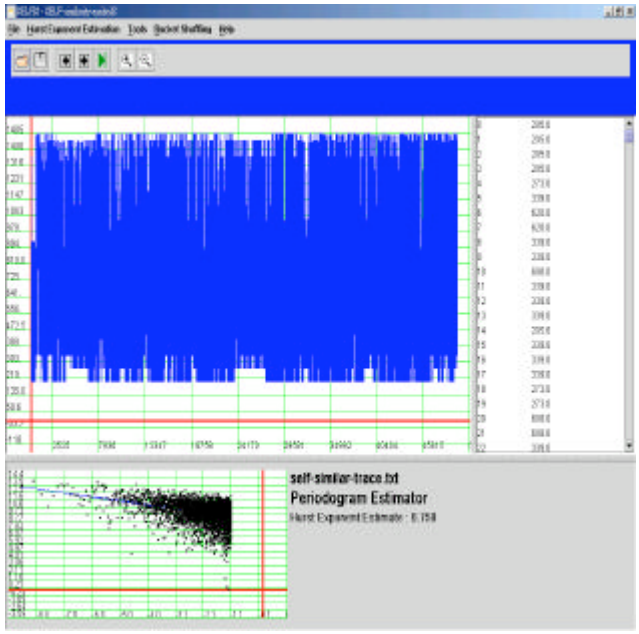

Figure 11. Trace analysis/identification with Selfis (Periodogram estimator invoked)

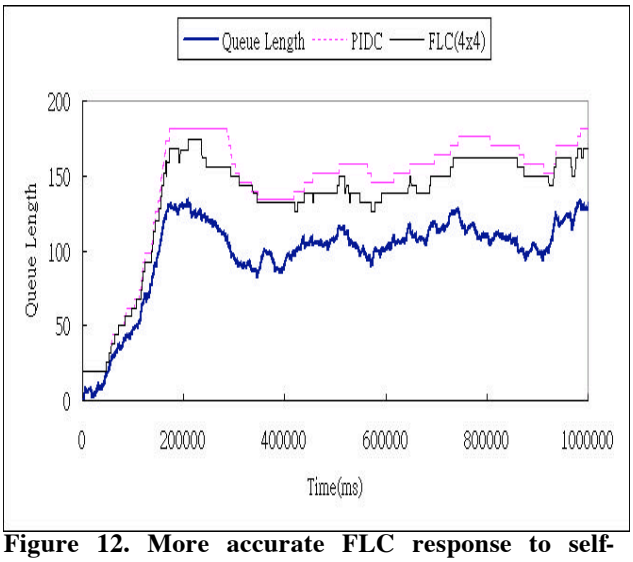

Figure 12. More accurate FLC response to self-similar traffic (same trace as Figure 10)

4.2 Poisson Traffic

The Poisson (exponential) traffic, which is SRD, is common in the Internet. It was confirmed by different experiments that the FLC indeed eliminates buffer overflow for this kind of traffic efficaciously. Figure 13 is the Periodgram result for the trace used for one of the experiments. For this trace the estimator yields H=0.482 with 99.84% confidence of its SRD character. The exponential nature of the trace is also confirmed by comparing its mean (m) and standard deviation (δ), which are 99 ms and 93 ms respectively. The "$99 \approx 93$" (i.e. $m \approx \delta$) condition confirms the random/exponential behaviour. Figure 14 shows how FLC maintains Δ consistently for this random trace.

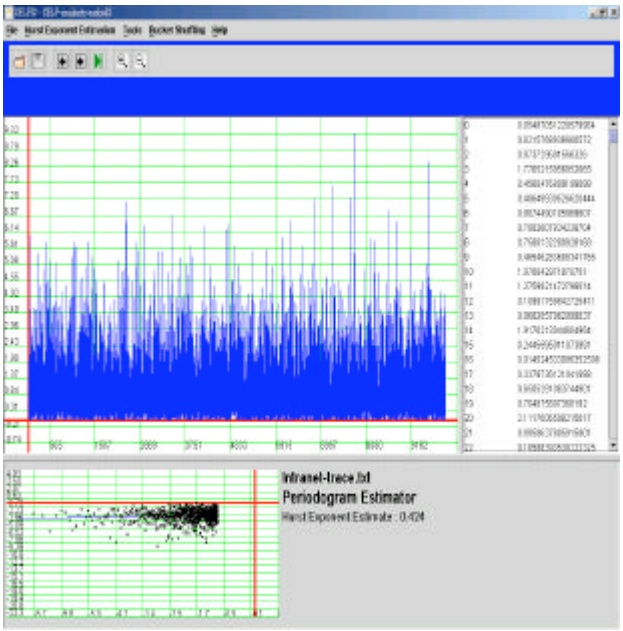

Figure 13. Poisson trace analysis/identification by Selfis (Periodogram estimator)

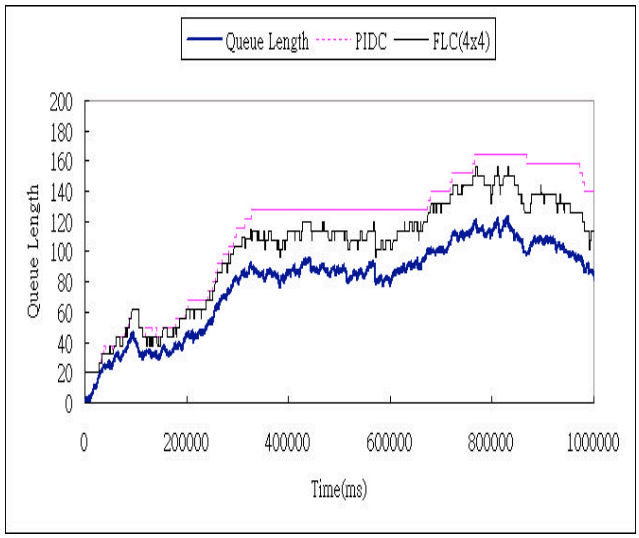

Figure 14. FLC maintains Δ consistently for Poisson traffic to eliminate overflow

5 Conclusion

The preliminary experimental results confirm that the Fuzzy Logic Controller indeed eliminates user-level buffer overflow efficaciously and consistently for both LRD and SRD Internet traffic patterns. The FLC is conceptually the "*fuzzy logic + PIDC*" combination. The fuzzy rules in the FLC tune the integral control (i.e. ICM) adaptively. As a result the dynamic buffer tuning process always maintains the Δ safety margin of the $\{0, \Delta\}^2$ objective function successfully. In this way the FLC effectively preserves the PIDC merits minus its shortcomings. The *VTune* timing analysis of the FLC shows that it has a shorter control cycle time on average than the PIDC working alone. This is the contribution by the inert "*don't care*" states in the FLC that require no action at all. The inertness of these states offsets the computation complexity of the FLC and makes its execution time comparable to that of the much structurally simpler PIDC. The next step planned for the research is to explore how optimal FLC design, for cost and effectiveness, can be achieved.

6 Acknowledgement

The authors thank the Hong Kong Polytechnic University for the research grants GT426, HZJ91 and APF75.

7 References

Bodin, U. Schelen, O. and Pink, S. (2000), Load-Tolerant Differentiation with Active Queue Management, ACM SIGCOMM Computer Communication Review, 30(3), July 2000, 4-16

Braden, B. Clark, D. Crowcroft J. Davie B., Deering, S. Estrin D. Floyd, S. Jacobson, V. Minshall, G. Partridge, C. Peterson, L. Ramakrishnan, K. Shenker, S. Wroclawski, J. and Zhang, L. (1998), Recommendation on Queue Management and Congestion Avoidance in the Internet, RFC 2309, April 1998

DWRED, Distributed Weighted Random Early Detection, Cisco Specification,

http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf Accessed October 2004

Floyd, S. and Jacobson, V. (1993), Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Trans. on Networking, August 1993

Ip, M.T.W., Lin, W.W.K., Wong, A.K.Y., Dillon, T.S. and Wang, D.H. (2001), 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

Jain, R. (1991), The art of Computer Systems Performance Analysis, Techniques for Experimental Design, Measurement, Simulation, and Modeling, Wiley, 1991

Karagiannis, T., Faloutsos, and M., Molle (2003), A User-friendly Self-similarity Analysis Tool, ACM SIGCOMM Computer Communication Review, 33(3), July 2003, 81-93

Karray, F., Gueaieb, W. and Al-Sharhan, S. (2002), The Hierarchical Expert Tuning of PID Controllers Using Tools of Soft Computing, IEEE Trans. on Systems, Man, and Cybernetics, Part B, 32(1) February 2002, 77-90

Kramer, G. (2000), Generator of Self-Similar Network Traffic, http://wwwcsif.cs.ucdavis.edu/~kramer/code/trf_gen1.html Accessed October 2004

Lakeshman, T. and Madlow, U. (1997), The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss, IEEE/ACM Trans. on Networking, 5(3), 1997

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

Medina A., Matta, I. and Byers, J. (2000), On the Origin of Power Laws in Internet Topologies, ACM SIGCOMM, 30(2), 2000

Mitsuru, O., Guenter, K., and Kouichi, O.(1998), IBM Aglets Specification, http://www.trl.ibm.com/aglets/spec11.htm Accessed October 25 2004

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

Resnick, S.I. (1997), Heavy Tail Modelling and Teletraffic Data, The Annals of Statistics, 25(50), 1997, 1805-1869

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

Stankovic, J.A. (1998), Deadline Scheduling for Real-Time Systems, Kluwer, 1998

Tanenbaum, A.S. (1996), Computer Networks, 3rd Edition, Prentice Hall, 1996

VTune, Intel VTune Performance Analyzer, http://www.intel.com/support/performancetools/vtune/classic.htm. Accessed October 25 2004

Wong, A.K.Y. and Dillon T.S. (1999), 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), Hong Kong SAR, Dec.1999, 268-275