# A Novel Fuzzy Logic Controller (FLC) for Improving Internet-based Real-Time Application Timeliness by Eliminating User-Level Buffer Overflow

Wilfred W.K. Lin and Allan K.Y. Wong
Department of Computing, Hong Kong Polytechnic University, Hong Kong SAR
Emails: cswklin@comp.polyu.edu.hk, and csalwong@comp.polyu.edu.hk

**Abstract**

A novel FLC (Fuzzy Logic Controller) for eliminating buffer overflow at the user level is proposed. It tunes the buffer size in a dynamic manner to ensure that the buffer length always covers the queue length by the given safety margin $\Delta$. The FLC operation is based on the objective function: $\{0,\Delta\}^2$. Our timing analysis shows that the FLC requires very short execution time for its control cycle and is therefore suitable for real-time applications. The FLC can be used as the complementary half (user level) of a unified solution to prevent buffer overflow in the entire client/server interaction over a TCP channel. The solution allows the overflow preventions at both the system and user levels to act in unison. The other half includes the throttling and AQM (active queue management) activities by the system (router level).

*Keywords: buffer overflow control, dynamic buffer tuning, Internet, fuzzy logic, A-PID*

## 1. Introduction

This aim of the novel FLC (*Fuzzy Logic Controller*) is to shorten the channel response or roundtrip time (RTT) in object-based real-time applications. If all the logical channels involved in an application have improved RTT, then the overall application timeliness will be enhanced. The FLC is intended for application over the Internet to harness the TCP (transmission control protocol) RTT through dynamic buffer tuning. Using fuzzy logic [1] to improve the accuracy of the extant *algorithmic PID* (A-PID) model (P for proportional, I for integral and D for derivative) [2] creates the FLC. Although our preliminary experience with the A-PID indicates that it always prevents user-level overflow, it has two intrinsic shortcomings. The first is that it always locks up buffer space even when it is not needed for overflow prevention. The second is that the buffer length under its control can get dangerously close to the queue length, threatening overflow for serious perturbations. The desire to preserve the A-PID power but without the two shortcomings has motivated the FLC proposal. The experimental results at this stage indicate that the FLC is a better controller because: a) it does not have the two A-PID shortcomings, b) it is as fast as a lone A-PID despite its complexity due to the presence of fuzzy control, and c) it is more accurate.

Real-time applications require timely and correct computation, which is mandatory for the hard and firm types and relatively more relaxed for soft applications [3]. Traditional real-time applications deal with embedded systems that have sensors within the vicinity, and the data they sample is fed immediately to the central processor. For example, the real-time software that controls a helicopter falls into this relatively more centralized category. For other applications such as a power plant, the sensors are dispersed and distant.

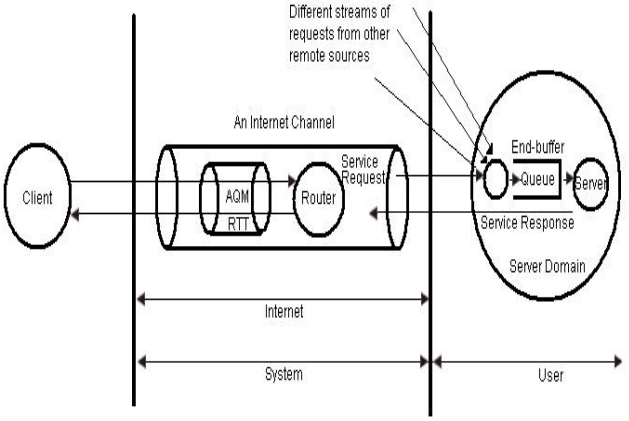


**Figure 1. Client/server interaction over a logical TCP channel**

The sampled data has to travel a bng distance to reach the central processor or a network of processors that collaborate over a small local area network (LAN) within the control center. In this case, the company can integrate all the distributed sensors (or front ends) and the processor(s) in two ways. The first is to put in place a dedicated, proprietary wide area network (WAN), and the second is to make use of the public Internet. The latter choice is more economical and free of the obsolescence problems that usually face the proprietary brands. One of the popular methods to harness timeliness is deadline scheduling [4]. The principle is to determine the order of activities in a timely manner. The norm is to let those jobs, which have deadlines about to expire, to run first [5]. Deadline scheduling usually works well in fault-free environments, but may fail in dynamic environments, such as the Internet, which is full of unpredictable faults. For example, if two cognate objects in an Internet-based real-time application interact over a logical channel of fault probability $\sigma$, the number of packet/message retransmissions can be high. In this case, no matter how hard the deadline scheduling algorithm works, the characteristic of the communication delay due to $\sigma$ would remain the same. This kind of delay can lead to the failure of any smartly designed real-time applications.

Using the Internet for real-time applications is an attractive, economic solution especially in the long-term sense. The inherent timeliness problem in the Internet, however, must be resolved first, and this is one of the objectives of the reputable IEEE conference, namely, "The IEEE Symposium on Object-based Real-time Distributed Computing (e.g. [6])". One factor that can cause serious Internet communication time delay is buffer overflow in the TCP (transmission control protocol) channels. Applications for the Internet are naturally logically distributed and object-based because only then they

get the full benefits from the inherent distributed parallelism of the physically distributed hardware. The constituent objects in these applications interact by the TCP channels in a client/server relationship by message passing [7] (Figure 1). The same server may receive service requests from different clients, and the incoming request streams converge by superposition [8,9]. The convergence can surge the queue length of a popular server to overflow easily. Besides, it is normal for a TCP channel to have an error probability $\sigma$. Then, the correlation between $\sigma$ and the *average number transmissions* (ANT) to get a success can be characterized by

$$ANT = \sum_{j=1}^{\infty} j s^{j-1}(1-s) = \frac{1}{1-s} \quad ; \quad s = 0 \quad \text{implying} \quad 100$$

percent success. Usually, $\sigma$ is collective by including all the possible faults. If $s = s_R + s_{OSys} + s_{User}$ is assumed, where $s_R$ for all faults but not buffer overflow, $s_{OSys}$ for system-level (channel) overflow only, and $s_{User}$ for user-level buffer overflow only, eliminating any part would significantly enhance the channel RTT due to less retransmissions.

The aim of the novel FLC (*Fuzzy Logic Controller*) proposed here is to eliminate $s_{User}$ efficaciously with the help of fuzzy logic [1] so that ANT is lowered for shorter channel RTT [10]. The FLC uses dynamic buffer tuning to ensure that the buffer length always covers the queue length by the given safety margin $\Delta$. The principle for the maintenance process is the objective function $\{0, \Delta\}^2$. The FLC is invoked when $|\Delta_C| > \Delta$ (i.e. $\Delta_C = B - Q$) is true, where B and Q are the currently sampled buffer and queue lengths respectively.

## 2. Related Work

The computing community always recognizes the possible negative impact by buffer overflow on system performance. The negativity increases with the size and complexity of a network-based system because of its heterogeneity and changeful traffic [11]. For the Internet this negativity is serious because of its sheer size and topological complexity. In the TCP/IP design choke packets are there for the routers to throttle the senders to prevent local buffer overflow [12] at the system level. The senders react voluntarily to slow down transmission. If throttling fails, the buffer overflow prevention relies on the ad hoc mechanism provided for the router. Previous overflow prevention works exclusively with a *static buffer length* (SBL) and the ultimate measure is to drop the new incoming packets. This approach has four basic flaws: a) the retransmissions of the dropped packets by the clients create widespread traffic congestion, b) the system-level SBL effort cannot prevent the user-level overflow due to the traffic superposition, c) overflow at the user level implies that the throttling resources already dished out by the system are wasted, and d) poor system performance because dropping packets at the user end means the loss of chance by the system to rectify the congestion problem earlier. By reviewing and comparing the recent published findings in the field, including our own experience [13, 14, 15] we conclude that buffer overflow prevention needs a unified solution. The rationale is to complement the system-level prevention effort with dynamic buffer tuning at the user (server) level. The complementary mechanism should work with a *variable buffer length* (VBL). The *"tuner/controller"* adjusts the buffer length adaptively so that it always covers the queue length. The appearance of VBL-based controllers is recent (e.g. [16, 17]), and the FLC is the fuzzy version of the VBL-based A-PID, which always stifles user-level buffer overflow so far, even with its two shortcomings. The FLC, which can be abstracted by: $FLC = "PID + Fuzzy\ Logic"$, totally eliminates the shortcomings. The pseudo-code in Figure 2 abstracts the A-PID control logic. The P and D control elements are the QOB (*queue length over buffer length*) ratio and the current $dQ/dt$ rate of change of the queue length (Q). The I control is the current ICM (*integral control mechanism*) output.

***If*** *{(dQ/dt > prescribed_positive_threshold) OR*
*[(dQ/dt is_ positive) AND*
*(QOB > 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 < prescribed_negative_threshold)]}* ***then***
$L_{now} = L_{now} - ICM;\ L_{now} \geq L_{minimum}$

**Figure 2. The logic for the A-PID controller**

## 3. The FLC Model

For all the present FLC designs the fuzzy control tunes only the ICM output for control precision. The single A-PID control domain is divided into finer fuzzy regions, which are manned with either a "*don't care*" (X state in Table 1) or a fuzzy rule [1]. The fuzzy rules are formulated with only three variables, namely, $dQ/dt$, QOB and ICM. When the FLC is in the X state, no action is required. Therefore, the X states together speed up the FLC execution to compensate for its increased complexity compared to the A-PID. A FLC design example (i.e. FLC(6x4), which divides the A-PID control domain into 24 fuzzy regions (6x4 matrix), is shown in Table 1. The *dot* represents the objective function: $\{0, \Delta\}^2$ by marking the reference/optimal point of $QOB = 0.8$ or 80%. This means a safety margin of $\Delta = \pm 0.2$ or $\pm$ 20 % about the reference. The set of fuzzy rules for the FLC(6x4) design is listed in Figure 3. The linguistic variables for the rules are: NL-Negative Large than optimal point, NS-Negative Small than optimal point, PS-Positive Small than optimal point, PL-Positive Large than optimal point, ML-Much Less than optimal point, SL-Slightly Less than optimal point, L-Less than optimal point, G-Greater than optimal point, SG-Slightly Greater than optimal point, and MG-Much Greater than optimal point. The set of fuzzy rules is the fuzzy knowledge base of the FLC (6x4) design. The "+" operator elongates the buffer length while the "-" shrinks it.

| | | dQ/dt | | | |
|---|---|---|---|---|---|
| | | NL | NS | PS | PL |
| QOB 0.7 | ML | - | - | - | - |
| 0.75 | SL | - | - | - | - |
| 0.8 | L | - | X | X | + |
| 0.85 | G | - | X | X | + |
| 0.9 | SG | + | + | + | + |
| | MG | + | + | + | + |

**Table 1. The FLC(6x4) design example**

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

**Figure 3. The set of fuzzy rules predefined for the FLC (6x4) design inTable 4**

## 4. Preliminary Results

Different combinations of FLC designs and fuzzy rules were experimented. The outcome confirms that the FLC approach is indeed more efficacious and produces smoother control than the A-PID model. The two inherent A-PID shortcomings are totally eliminated by the FLC fuzzy logic. Figure 4 compares the performance profiles between the FLC(6x4) and a lone A-PID. The trace for this plot was sampled for the logical channel that connected the LabTrobe University site in Australia and our laboratory in the Hong Kong PolyU. This setup is a generic one for our research, and in each test case the "*Buffer controller*" will be instantiated with the wanted control model. The plot shows that the buffer length being tuned by the FLC no longer lingers at high values because the unused buffer space is released immediately. The FLC maintains the given $\Delta$ consistently so that the queue length never to gets dangerously close to the buffer length.

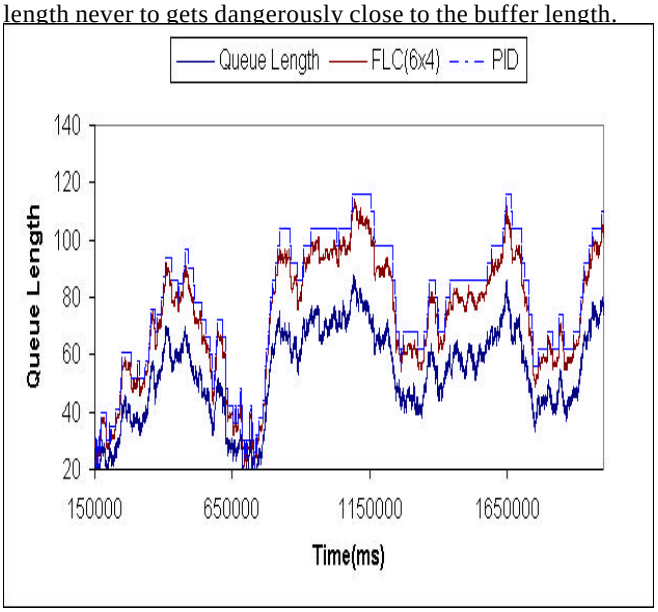


**Figure 4. Comparing the FLC(6x4) (Table 1) and A-PID performance profiles**

The timing analysis of the two controllers was carried out with the *Intel's VTune Performance Analyzer* [18], which measures the execution time by number of clock cycles. The result shows that the execution times for the controllers are comparable. The FLC needs 255 T/clock cycles for execution on average, while the A-PID requires 205T. Measurement of clock cycles is neutral to the hardware because the measurement can be converted into the physical execution time of the chosen platform. If a platform operates at 933 MHz, then the physical times for 205T and 255T are $2.2*10^{-7}$ and $2.73*10^{-7}$ seconds respectively. These figures represent the operating limits of the controllers in real-time applications. It means that the duration of the error to be remedied must be longer than the limit for control/remedy to be correct.

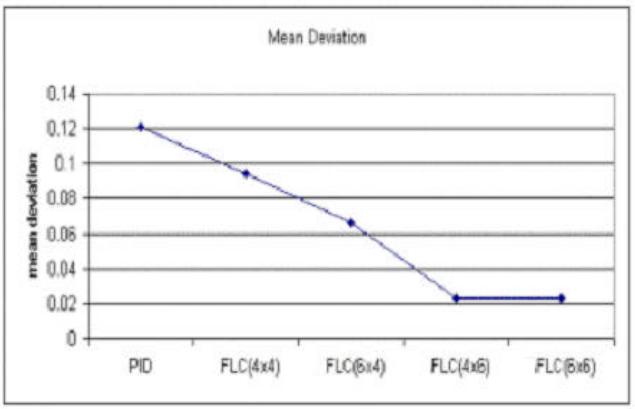


**Figure 5. There is an optimal region for the FLC design process**

Figure 5 is deduced from all the available FLC data empirical, and it indicates that optimal FLC design is possible. It shows the breakpoint (around FLC (4x6)) where the benefits of having more complex FLC design start to wane. The experimental data also shows that the D control element ($dQ/dt$) has a greater effect than the P element (QOB) in general. In order to identify the exact region for optimal FLC design deeper investigation in the future is necessary. This investigation should involve metrics dimensioning, which identifies other useful metrics and their impacts than $dQ/dt$ and QOB.

**5. Conclusion**

The novel FLC is a combination of fuzzy logic and the A-PID model. It is VBL-based because it eliminates buffer overflow at the user (server) level by dynamic buffer tuning. The experimental results confirm that the FLC is more efficacious and smoother than the A-PID. The FLC is intended to be a complementary part of a unified solution by eliminates $s_{User}$. In doing so it prevents the throttling effort dished out by the system from being wasted. This results in less retransmission and shorter channel RTT to provide better timeliness for Internet-based real-time computing. The contribution limit for the FLC is its 255 execution clock cycles on average. The preliminary result shows that optimal FLC design is possible. The next phase in the research is to identify the region for optimal FLC design, and this may involve metrics dimensioning.

**6. References**

[1] L.A. Zadeh, Fuzzy Logic, Neural Networks, and Soft Computing, Communications of the ACM, 37(3), 1994

[2] 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, 2001

[3] J.A. Stankovic et al., Deadline Scheduling for Real-Time Systems, Kluwer Academic Publishers, 1998

[4] J.A. Stankovic et al., Implications of Classical Scheduling Results for Real-Time Systems, IEEE Computer, 1995

[5] Q. Zheng and K.G. Shin, On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks, IEEE Transactions on Communications 42(4), 1994

[6] The 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC' 99, Saint Malo, France, 1999

[7] S.M. Lewandowski, Frameworks for Component-Based Client/Server Computing, ACM Computer Survey, 1998

[8] Tharam S. Dillon, Allan K.Y. Wong and Wilfred W.K. Lin, A Model for Mobile-Agent-Based Distributed Computing over the Internet with Security and Dynamic Load Balancing, Proc. of 3rd the Pacific Rim International Workshop on Multi-Agents (PRIMA2000), Australia, 2000

[9] Allan K.Y. Wong, Tharam S. Dillon, Wilfred W.K. Lin and T.W. Ip, M2RT: A Tool Developed for Predicting the Mean Message Response Time for Internet Channels, Computer Networks, vol. 36, 2001

[10] Allan K.Y. Wong, Wilfred W.K. Lin, May T.W. Ip and Tharam 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 (JPDC), vol. 62, 2002

[11] F. Ren et al., Design of a Fuzzy Controller for Active Queue Management, Computer Communications, vol. 25, 2002

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

[13] J. Semke, J. Mahdavi, and M. Mathis, Automatic TCP Buffer Tuning, Proc. of the ACM SIGCOMM, 1998

[14] Wilfred W.K. Lin and Allan K.Y. Wong, A Novel Adaptive Fuzzy Logic Controller (FLC) to Improve Internet Channel Reliability and Response Timeliness, Proc. of The 8th IEEE Symposium on Computers and Communications (ISCC' 2003), Turkey, 2003

[15] E. Weigle and W.C. Feng, A Comparison of TCP Automatic Tuning Techniques for Distributed Computing, Proc. of the 11th IEEE International Symposium on High Performance Distributed Computing, 2002

[16] 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), 1999

[17] Wilfred W.K. Lin, May T.W. Ip, Dianhui Wang, Alan K.Y. Wong and Tharam S. Dillon, A Neural Network Based Proactive Buffer Control Approach for Better Reliability and Performance for Object-Based Internet Applications, the Proc. of the Parallel Processing Techniques and Applications (PDPTA2001) conference, 2001

[18]IntelVtune, http://developer.intel.com/software/products/vtune/