Detection of Fractal Breakdowns by the Novel Real-time Pattern Detection Model (Enhanced-RTPD + Holder Exponent) for Web Applications

Wilfred W.K. Lin [1], Allan K. Y. Wong[2], Tharam S. Dillon[1], Elizabeth Chang[1]
[1]Digital Ecosystems and Business Intelligence Institute (DEBII)
Curtin University of Technology
e-mail : wilfred.lin@gmail.com
[2] Department of Computing, Hong Kong Polytechnic University, Hong Kong S.A.R.
e-mail: csalwong@comp.polyu.edu.hk

***Abstract*—The M[3]RT-based real-time traffic pattern detector proposed identifies the Internet traffic pattern on the fly. Firstly it determines if a time series aggregate is stationary. Secondly it confirms if the aggregate exhibits short-range dependence (SRD) or long-range dependence (LRD). Thirdly it detects if the smooth system operation has suddenly become irregular and chaotic. This detection is achieved by computing the instantaneous value of the Holder exponent that has a (0,1) range to accommodate different degrees of fractality. When the Holder exponent has wandered outside the (0,1) region, fractal breakdown has occurred. The capability of detecting such breakdowns by a real-time application enables it to avoid sudden failure. The Intel's VTune Performance Analyzer indicates the proposed model can be deployed in real time effectively. This feature is of importance to the reliability improvement of web applications which run on the Internet.**

***Index Terms*—real-time traffic pattern detection (RTPD), fractal breakdown, Holder exponent, Long Range Dependence**

I. INTRODUCTION

The Internet is an important element of the supporting infrastructure for web Applications. Applications running on the Internet are naturally object-based and in this way they produce speedup from the intrinsic parallelism provided by the distributed networked hardware. Being object-based the collaborating logical objects interact in a client/server relationship by message passing in an asynchronous and end-to-end manner. This mode of communication is prone to errors and retransmission because passing a message from one end to another means traversing links of varying quality. It is a known fact that it is difficult to harness the end-to-end roundtrip time (RTT) over a TCP channel. As a result measures are needed to overcome this difficulty especially for time-critical applications that have execution deadlines to meet. The length of the service roundtrip time is associated with the channel's error probability ρ for retransmissions. Thus, the *average number of trials* (ANT) for a successful transmission with ρ is $ANT = \sum_{j=1}^{\infty} j[\rho^{j-1}(1-\rho)] \approx \lim_{j\to\infty} 1/(1-\rho)$. In fact, ρ encapsulates different faults and errors. Off-line analyses of pre-collected Internet traffic traces (i.e. inter-arrival-time (IAT) time series) have empirically confirmed that different traffic patterns increase ρ to various degrees [1,2]. However, there is little experience published for real-time traffic pattern detection (RTPD) for on-line applications. With RTPD capability incorporated a real-time application can detect the traffic pattern changes and nullify their ill effects on performance [3]. In reality, the Internet traffic pattern changes suddenly, for example, from LRD (*long-range dependence*) to SRD (*short-range dependence*) or vice versa [4]. To fill the vacuum of not having any RTPD techniques in the field, the *enhanced real-time traffic pattern detector* (E-RTPD) [5] was proposed. The E-RTPD has the following capabilities:

a) If a traffic trace (in the light of a time series aggregate) is stationary, then it differentiates short-range dependence (SRD) (e.g. Markovian traffic) from long-range dependence (LRD) (e.g. self-similar and heavy-tailed traffic).
b) It confirms if the LRD traffic is heavy-tailed or self-similar (i.e. non-differentiable). The element in the E-RTPD that identifies self-similarity is the *self-similarity* (S^2) *filter* [6].
c) It shows if the fractal dimension (or fractality) of the self-similar traffic changes. If a sizeable time series is arbitrarily taken, then different data points

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

may have different fractality. If this happens, then the times series is multifractal.

The E-RTPD uses the Hurst (H) effect (i.e. H_{ss} [8]) as the yardstick to differentiate LRD for $0.5 < H < 1$ from SRD for $0 < H < 0.5$ for SRD. For a time series aggregate of size m in a stochastic process X, namely $X^m = \{X_l^m : l \geq 1\}$, the *autocorrelation function* (ACF) (*correleogram*) is $r^m(l) = \sum_{l=1}^{N} r^m$, where r^m is the autocorrelation of X^m and l identifies the aggregate level. The ACF of LRD traffic is non-summable (i.e. $\sum_{l=1}^{N \to \infty} r^m \approx \infty$), but it is summable for SRD (i.e. $\sum_{l=1}^{N \to \infty} r^m < \infty$). The previous E-RTPD is incomplete because subsequent experience shows that it lacks the capability to differentiate multifractality from a fractal breakdown. It is possible for a smooth distribution/function to have different degrees of fractality at local data points (i.e. it is multifractal). For some functions, however, fractality may breakdown at singular data points, which represent irregular regions. Fractal breakdown identification pinpoints where the system has gone into chaotic operation and may fail suddenly because of this. Its detection helps a system to invoke the necessary measures to avoid failure. Our literature search shows that the Holder exponent which has a (0,1) range is useful as a fractal breakdown indicator [7,8]. Therefore, the novel RTPD model proposed in this paper is an improved E-RTPD version called the *versatile* RTPD (V-RTPD), which is basically the "E-RTPD + Holder exponent" combination. The V-RTPD has a short execution time and is therefore suitable for real-time applications.

The basic capability to identify self-similarity (also called self-affinity) is to compute its *dimension* (D). If an object is geometrically, recursively split into similar pieces, at the K^{th} iteration step the total measure of the object is "*product of the number of similar pieces and* O^D". The parameter O is the splitting *resolution* or *reduction*. The *Cantor Set* is a simple example in which a line segment of interval [0,1] is drawn 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 steps continuously. The K^{th} iteration produces 2^K similar line segments of length $s = (1/3)^K$ each. The *Cantor Set's* self-similarity dimension is defined by the formula $D_s = 2^K * (1/3)^K$ or $D_s = [(K \log(2))/(K \log(3))] \approx 0.63$. An object is fractal if its D value is non-integer. Different non-converging dimension definitions exist, and the *Cantor Set* provides only a conceptual basis. A stochastic process $X(t)$ is stationary, self-similar and fractal provided that its two finite-dimensional distributions, $X(at)$ and $a^H X(t)$ are identical for $a > 0$. Therefore, the following expression holds, $\{X(at_1), X(at_2) \ldots X(at_n)\} \equiv \{a^H X(t_1), \ldots a^H X(at_n)\}$; $\equiv$ means equality and H (Hurst exponent) is the scaling factor.

II. RELATED WORK

The E-RTPD core is derived from the traditional R/S (*rescaled adjusted statistics*) approach for *non-real-time* (i.e. "post-mortem") applications. It is basically a real-time "$M^3RT + R/S + filtration$" package. The M^3RT element is a *micro Convergence Algorithm* (CA) approach [9], which enhances the traditional R/S approach (i.e. E-R/S) for real-time application. The *filtration* process invokes the right filter to identify the exact traffic pattern; for example, the *modified QQ-plot filter* identifies heavy-tailed distributions. The E-R/S calculates the Hurst (H) value on the fly. The traditional R/S is defined by $R/S = \frac{\max\{W_i : i = 1,2,\ldots,k\} - \min\{W_i : i = 1,2,\ldots,k\}}{\sqrt{\text{var}(X)}}$. The parameter W_i is computed as $W_i = \sum_{m=1}^{i} (X_m - \overline{X})$ for $i = 1,2,\ldots k$, where $\overline{X}$ is the mean of $\overline{X} = 1/k \sum_{i=1}^{k} X_i$. The best value for *k* is usually found by trial and error, and this becomes the drawback because R/S accuracy and speed depends on *k*. The R/S ratio is the rescaled range of the stochastic process X over a time interval *k*, $\{X_i : i = 1,2,\ldots k\}$. In the E/RS the $\overline{X}$ value is computed by M^3RT in a real-time manner. The useful R/S feature is the log-log of $R/S \approx (k/2)^H$, which yields the H value. The CA operation is derived from the *Central Limit Theorem*, and is summarized by equations: (2.1) and (2.2). 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 therefore depends on the interval for collecting the *F* samples. It was confirmed previously that M_i has the

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

fastest convergence for F=14 [9]. The 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 started running. In the E-R/S, M_i replaces $\overline{X}$ to yield $W_i = \sum_{m=1}^{i}(X_m - M_i)$. This replacement makes the E-R/S more suitable for real-time applications for the number of data items (e.g. IAT) to calculate W_i is predictable (i.e. $F = 14$). In real-life applications $\overline{X} = 1/k \sum_{i=1}^{k} X_i$ needs much longer computation than M_i for two reasons: a) k is usually larger than F, and b) the IAT among X_i could be so large that the product of "k and average IAT" is a significant time delay. In an E-RTPD implementation the E-/RS, M^3RT and traffic filters are running in parallel. The E-RTPD execution time depends on the E-R/S module, which has the longest execution. The *Intel's VTune Performance Analyzer* [10] records from the Java E-RTPD prototype the following intrinsic average execution times in clock cycles for the different components: 981 for E-R/S, 250 for M^3RT, 520 for the modified QQ-plot filter, and 1455 for the S^2 *filter*. Although the S^2 *filter* quickly detects self-similar traffic on the fly, it cannot differentiate multifractality from fractal breakdown.

$$M_i = \frac{M_{i-1} + \sum_{j=1}^{j=F-1} m_j^i}{F} \text{.....(2.1);}$$

$$M_0 = m_{j=0}^{i=1} \text{..... .(2.2); } i \geq 1$$

2.1. The *Self-Similarity Filter*

LRD traffic can be heavy-tailed or self-similar. The *self-similarity* (S^2) *filter* separates the two. 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 for at least the FSNDPP (*Fractal-Shot-Noise-Driven Poisson Process*) is not heavy-tailed. The S^2 *filter* is based on the "*asymptotically second-order self-similarity*" concept, or simply called *statistical* $2^{nd}OSS$ or $S2^{nd}OSS$. For an aggregate $X^m = \{X_l^m : l \geq 1\}$ $S2^{nd}OSS$ for $m \to \infty$ means that its *autocorrelation function* (ACF), namely $r^m(l)$ (for X^m), is proportional to $l^{-(2-2H)}$. $S2^{nd}OSS$ is LRD for its ACF is non-summable. The condition of "$r^m(l) \propto l^{-(2-2H)}$ for $m \to \infty$" is mathematically the *slowly decaying variance property*. That is, the variance of the mean of sample size m decays more slowly than m. This is represented by the expression, $Var(X^m) \propto m^{-\beta}$. For a stationary $2^{nd}OSS$ process X and $0.5 < H < 1$ the value of $\beta = 2 - 2H$ should apply. Equations (2.3) and (2.4) summarize the $S2^{nd}OSS$ property and they hold for the weaker condition in equation (2.5). The *slowly decaying variance* property is clear if a log-log plot is produced for equation (2.3). As shown by equation (2.6), $\log(Var(X))$ 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. The key for the S^2 *filter* operation is to choose a sufficiently large m, which is a multiple (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 (e.g. β for aggregate 2 is available at $Ag = 2$ in Figure 2).

$$Var(X^m) = \frac{1}{m^{(2-2H)}} Var(X) \text{.....(2.3)}$$

$$r^m(l) = r(k) \text{....(2.4)} \quad \lim_{m \to \infty} r^m(l) = r(k) \text{....(2.5)}$$

$$\log(Var(X^m)) = \log(Var(X)) - \beta \log(m) \text{....(2.6)}$$

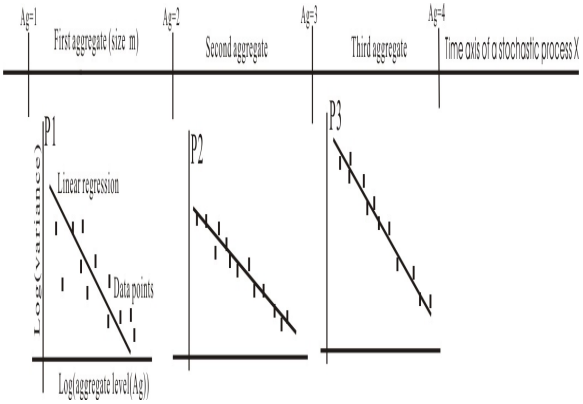

Figure 2. The "continuous *aggregate based* (CAB)" approach

The process in the S^2 *filter* to calculate β is the "*continuous aggregate based* (CAB)" method. The CAB evaluates if an aggregate is stationary by

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

checking its Gaussian property or "Gaussianity" [11] 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 of independent stationary increments. The S^2 *filter* follows the CAB procedure 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 [12]. 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 2 works with the aggregates $X^m_{Ag=l}$ in a 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 (2.6), 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 confirms 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 aggregate that combines P2 and P3 is multifractal in nature. The D value may change over time due to various factors, for example, the ON/OFF situations in the network [4]. A changing D or H is a sign of possible non-linearity in the stochastic process being examined.

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-shape 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 $skewness = 0$.

2.3. Some Previous Experimental Results

These experiments were conducted on the stable Aglets mobile agent platform, which is designed for Internet applications (e.g. web Applications). The Aglets makes the experimental results scalable for the open Internet. The setup for the experiments is shown in Figure 3, in which the driver and server are both aglets (agile applets). 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 [5] 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 Web Applications 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. Table 1 summarizes seven of the many different simulations conducted. The self-similar traces used in the experiments are generated by the Kramer's tool [13]. The useful information deduced from the Table 1 is 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. For $\psi > 0.4$ traffic self-similarity scales differently as shown by Figure 5 and Figure 6. Our analysis indicates that this is possibly a sign of multifractal traffic or the beginning of non-linear scaling. Both Figure 5 and 6 work with the threshold of $Th_{R^2} = 0.9$.
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 $\psi \le 0.4$ the scaling is basically the same (i.e. a *monofractal* sign). The β value in every case (row) in Table 1 is the average of several aggregates for the same stochastic process X.
d) The kurtosis and skewness vary with different self-similar traces. Nevertheless they indeed indicate the presence of a bell curve.

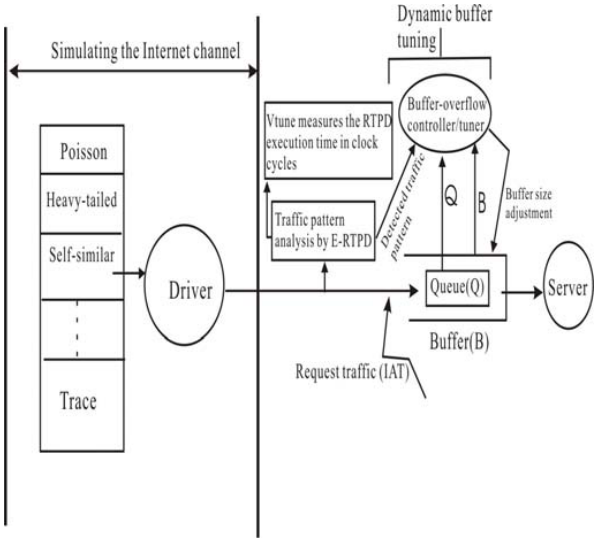

Figure 3. Setup for the S^2 *filter* experiments

β	$H = (1 - \beta/2)$	R^2 (coefficient of determination)	loading
0.6583	0.671	0.956 (95.6%)	0.1 (10
0.6809	0.660	0.975 (97.5%)	0.2
0.6425	0.679	0.977 (97.7%)	0.3
0.6473	0.677	0.972 (97.2%)	0.4
0.4685	0.766	0.959 (95.9%)	0.5
0.3762	0.812	0.885 (88.5%) (less than Th_{R^2})	0.6 (re
0.1978	0.901	0.605 (60.5%)	0.7 (re

Table 1. S^2 filter'(log(variance) versus log (aggregate level) to find β

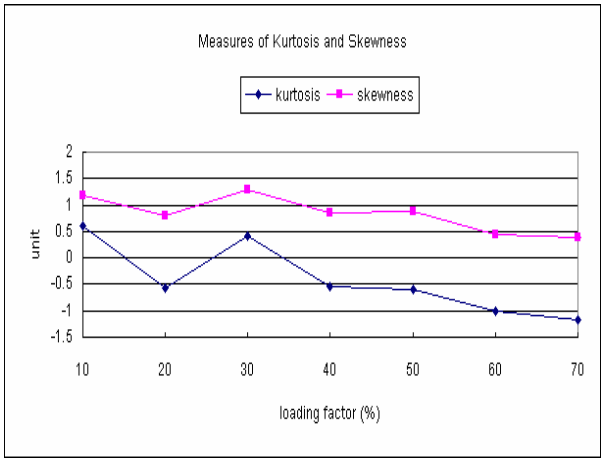

Figure 4. Kurtosis and skewness measurements

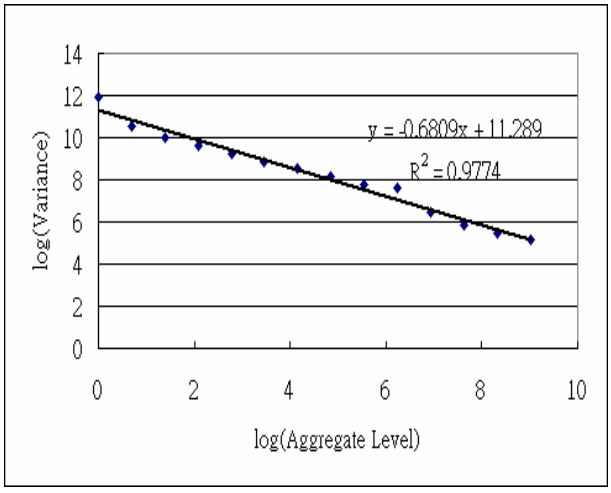

Figure 5. S^2 filter yields slope = -0.6809($\beta = 0.6809$), R^2= 97.74% for $\psi = 0.2$

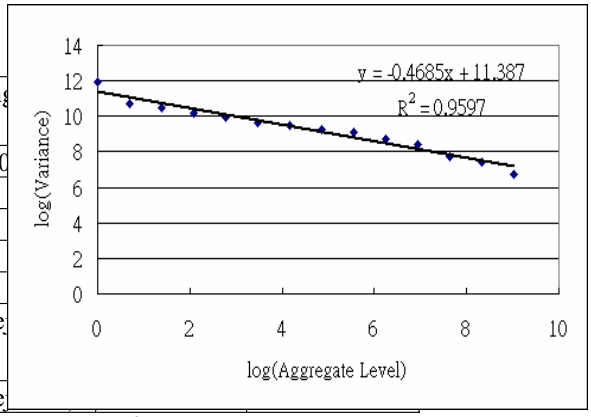

Figure 6. S^2 filter yields slope = -0.4685($\beta = 0.4685$), R^2= 95.97% for $\psi = 0.5$

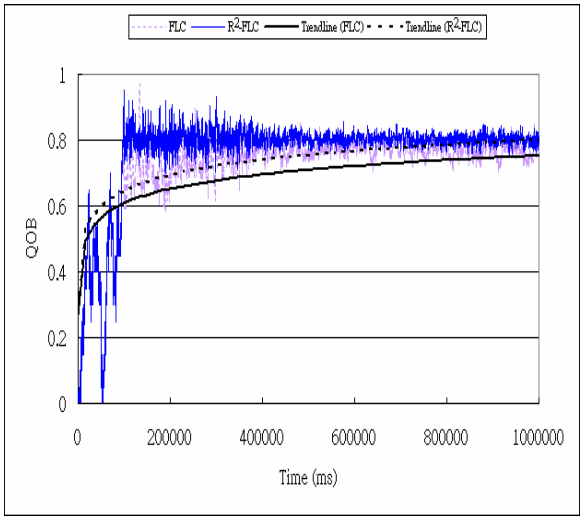

Figure 7. Faster convergence of the FLC+ S^2 filter (R2-FLC) than the FLC working alone

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

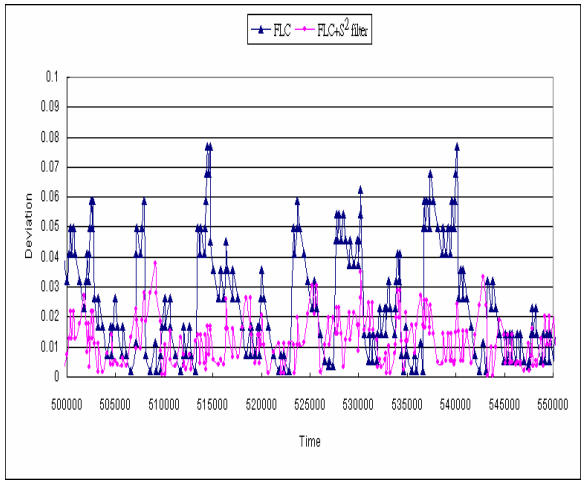

Figure 8. Less MD deviation by FLC+ S^2 than the FLC alone

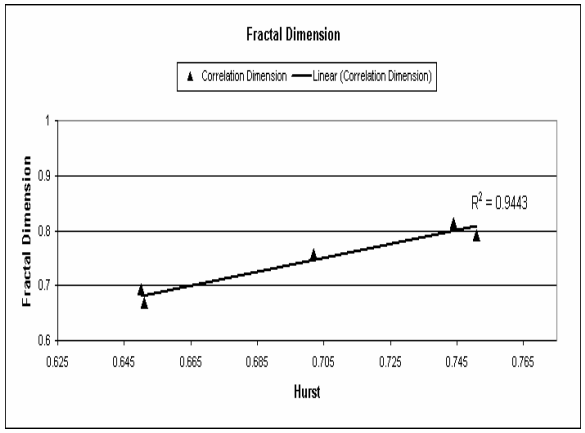

Figure 9. D/H correlation for Table 1

III. THE VERSATILE RTPD MODEL

The versatile RTPD (V-RTPD) model is basically the "E-RTPD + Holder filter". The Holder filter enables the V-RTPD to differentiate multifractality from fractal breakdown. The local degree of fractality of a data point in a time series can have any Holder exponent (HE) value in (0,1) range. If the exponent is outside the range, fractal breakdown has occurred. In reality the stationary process may has gone into a chaotic and unstable condition. If this lasts too long, the system may fail. The local HE for point t of function $f(t)$ in the continuous domain that involves an infinite time series can be expressed by equation (3.1). This equation, however, is not suitable for real-time applications because in these situations the HE has to be computed from a small aggregate X^m on the fly (i.e. for a small m).

$$HE(t) = \lim_{h \to 0} \frac{\log(|f(t+h) - f(t)|)}{\log(|h|)} \text{.....(3.1)}$$

In order to calculate the Holder exponent in a real-time manner a group of researchers [8] had proposed a new discrete technique. The aim is to calculate the corresponding HE for the data point x_i with a small aggregate X^m. The aggregate size m is the window of the sampled data points, x_0, $x_1, \ldots x_i, x_{i+1}, \ldots x_{m-1}$. The calculation is controlled by another parameter λ with a range (0,1). Previous experience indicates that $m = 20$ is sufficient to yield a reasonable HE_i. The discrete technique can be summarized in three sequential steps represented by the equations (3.2), (3.3) and (3.4), which can be generalized as equation (3.5). The criteria for applying these equations, which should together form a *discrete tool for detecting fractal breakdown* (DT-DFB) on the fly, are as follows:

a) In the first step the l integer must satisfy the $0 \le i + l \le m$ condition, where m is the aggregate size for data sampling on the fly. For each integer $l \ne 0$ the calculation of $R_{i,l}$ goes from $-s$ to $+s$ about the chosen x_i data point for $s = m/2$.

b) In the second step k represents the absolute integer distance $|s|$ about x_i, $1 \le k \le l$. The aim is to find the minimum value between the R_{i-l} and R_{i+l}.

c) The third step computes the weighted average of the $h_{i,k}$ approximations.

$$R_{i,l} = \frac{\log|x_{i+l} - x_i|}{\log(\frac{|l|}{m})} \text{.....(3.2)}$$

$$h_{i,k} = \min(R_{i,l} : |l| \le k \text{.....(3.3)}$$

$$HE_i = \frac{(1-\lambda)}{(1-\lambda^l)}(h_{i,1} + \lambda h_{i,2} + \lambda^2 h_{i,3} + ... + \lambda^{l-1} h_{i,l}) \text{.....(3.4)}$$

$$HE_i = \frac{(1-\lambda)\sum_{i,q=1}^{q=l}(\lambda^{q-1} h_{i,q})}{(1-\lambda^l)} \text{.....(3.5)}$$

IV. EXPERIMENTAL RESULTS

The implementation of equation (3,5) is called the Holder filter in the V-RTPD package. With a setup similar to Figure 3 our preliminary experimental results show that the detection of fractal breakdowns can indeed by achieved. The experiments that produced Table 1 were repeated, but V-RTPD was used instead of the old E-RTPD. It is confirmed that the sudden drop of the R^2 value in row 6 in Table 1 to less than

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

90% is caused by fractal breakdown. In the old E-RTPD approach the S^2 *filter* suggested a change in fractal dimension and indicated this could be the result of non-linear operation because of the obvious change in the H value. The Holder filter, however, confirmed fractal breakdowns in the time series as shown in Figure 10. The data samples of interest in this example range from the 250th to the 450th. In between the chaotic regions of a rapidly oscillating nature, (e.g. the section from around the 295th data point to the 380th sample) are smooth parts of the time-series distribution. The fractal breakdowns are indicated by the negative Holder exponent values. Figure 11 shows the time analysis of the estimator by the Intel's VTune Timing Analyzer [10]. The V-RTPD is a complete real-time tool because it differentiates multifractality and fractal breakdowns. With the detected knowledge real-time application that has incorporated the V-RTPD can take proactive steps to prevent failure in chaotic and irregular situations.

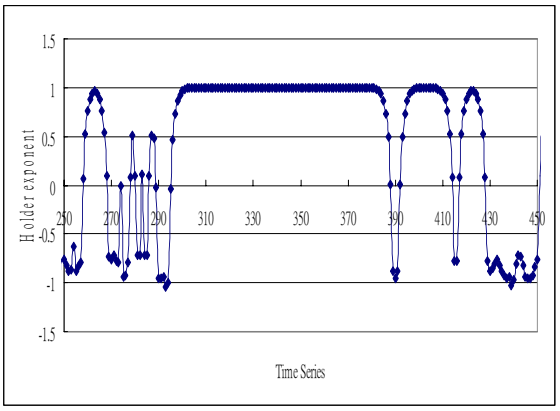

Figure 10 Fractal breakdowns in row 6(loading factor: 60%) of Table 1 detected by DT-DFB

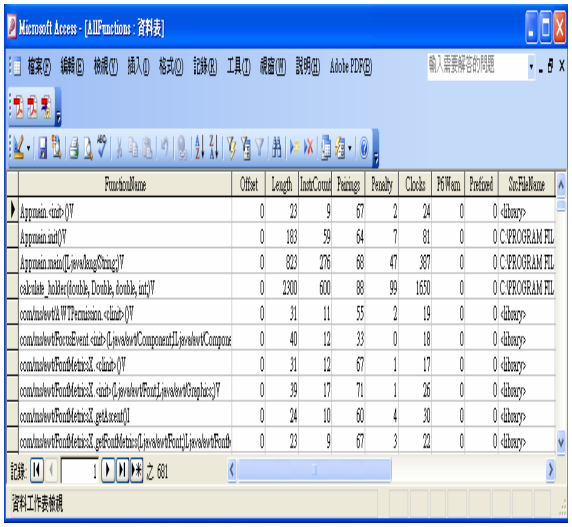

Figure 11 Timing analysis of the detector (needs 1650 clock cycles) by Intel's VTune Timing Analyzer

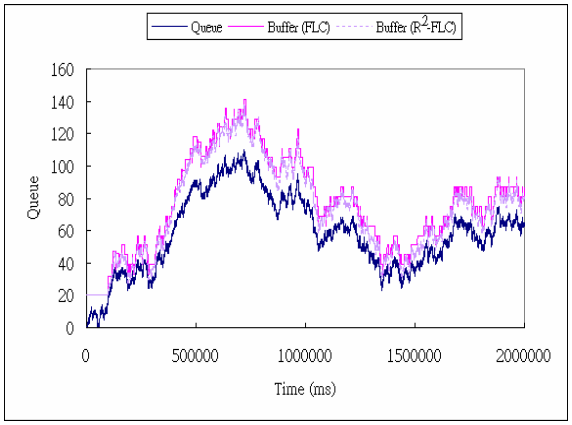

Figure 12.The performance of the R2-FLc and the FLC under simulation

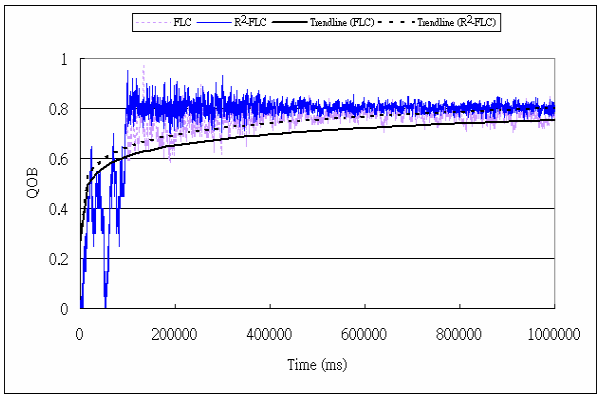

Figure 13. Faster Convergence of the R2-FLC over the FLC to the reference value

V. CONCLUSION

The novel versatile real-time traffic pattern detector (V-RTPD) is proposed for identifying the Internet traffic pattern for the web Applications on the fly. It detects if the smooth system operation has suddenly become irregular and chaotic. This detection is achieved by computing the instantaneous value of the Holder exponent that has a (0,1) range to accommodate different degrees of fractality. A smooth performance distribution such as a time series may embed a varying fractality at different times due to the system dynamics. If the Holder exponent has wandered outside the (0,1) region, fractal breakdown has occurred. The capability of detecting such breakdowns by a real-time application enables it to avoid a sudden failure. The next step in the research is to perfect the detection sensitivity and accuracy by enabling V-RTPD to determine the range (or size m) and the weighted regression coefficient but without causing any unnecessary and significant latency in the process.

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

VI. REFERENCES

[1] B. Braden et al., Recommendation on Queue Management and Congestion Avoidance in the Internet, RFC2309, April 1998

[2] 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, Sept. 2002, 1433-1453

[3] 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, IEEE Transactions on Systems, Man and Cybernetics, Part C, 36(3), May 2006, pp.397 -410

[4] 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

[5] Wilfred W. K. Lin, Richard S.L. Wu Allan K. Y. Wong, and Tharam S. Dillon, A Novel Real-Time Traffic Pattern Detector for Internet Applications, Proc. of the Australasian Telecommunication Networks and Applications Conference, Sydney, Australia (ATNAC'04), Dec 2004, 224-227

[6] Wilfred W. K. Lin, Allan K. Y. Wong, Richard S.L. Wu, and Tharam S. Dillon, 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, ICETE'05, Reading UK, October 2005, vol. 1, 94 – 101

[7] K.Daoudi, A.B. Frakt and A.S. Willsky, Multiscale Autoregressive Models and Wavelets, IEEE Transactions on Information Theory, 45(3), April 1999, 828-845

[8] M. Shereshevsky2003, J. Crowell, B. Cukic, V. Handikota and Y. Liu, Software Aging and Multfractality of Memory Resources, Proc. of the 2003 IEEE International Conference on Dependable Systems and Networks (DSN'03), 2003

[9] Allan K.Y. Wong and Joseph H.C. Wong, A Convergence Algorithm for Enhancing the Performance of Distributed Applications Running on Sizeable Networks, The International Journal of Computer Systems, Science & Engineering, 16(4), July 2001, 229-236

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

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

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

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

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