Self-adaptive techniques for the load trend evaluation of internal system resources

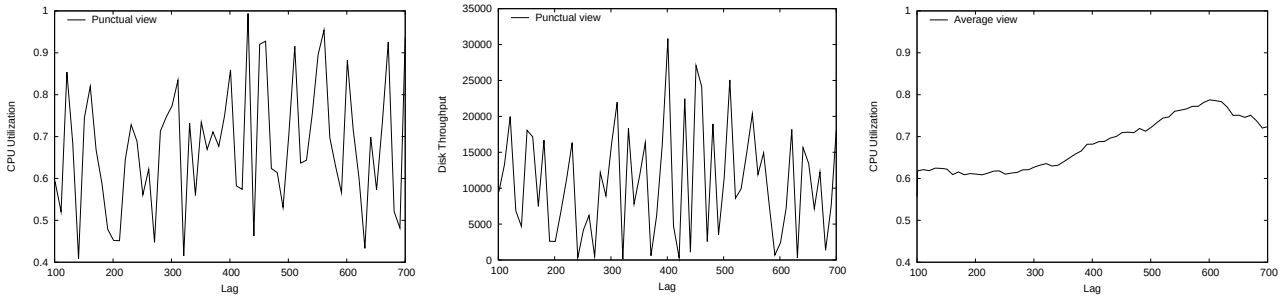
Sara Casolari and Michele Colajanni and Stefania Tosi
Department of Information Engineering
University of Modena and Reggio Emilia
{sara.casolari, michele.colajanni}@unimore.it

Abstract

Modern distributed systems that have to avoid performance degradation and system overload require several runtime management decisions for load balancing and load sharing, overload and admission control, job dispatching and request redirection. As the external workload and the internal resource behavior of the modern system is highly complex and variable, self-adaptive techniques require a stable vision of the system behavior. In this paper we propose a trend model that guarantees a robust interpretation for load-aware decision algorithms. Various experimental results in a Web cluster demonstrate that the proposed models and algorithms guarantee better stability of the load and a reduction of the response time experienced by the users.

1 Introduction

The majority of modern applications are served by distributed systems that must accommodate varying demands for different types of processing within certain time constraints. Behind each user request there are several runtime decision algorithms that are oriented to load balancing and load sharing, overload and admission control [17, 16], job dispatching and redirection even at a geographical scale [7]. These runtime management decisions are taken on the basis of the present and past knowledge of the state of the system resources, that is becoming increasingly complex and variable. Robust self-adaptive techniques seem to be the inevitable answer to manage these critical services. This paper addresses a fundamental issue that is at the basis of any robust *load-aware* decision system that has to satisfy scalability and availability requirements, and has to avoid performance degradation and system overload: how to guarantee an adequate knowledge of the behavior and state of the internal hardware/software resources. The problem is that most runtime management algorithms consider just instantaneous or average values of the resource state coming from measurements (e.g., CPU utilization, disk and network throughput, memory occupancy). In other contexts [8], these resource measures are valid sources to decide where the system is, where the system is going, whether is it necessary or not to activate some runtime management processes. The large majority of these systems are based on some functions that work directly on resource measures [17, 2, 18]. Other papers [12, 9] have shown the utility of the stochastic models based on moving-average and on linear regression to give a more reliable load representation. Moreover, also some low-pass filtering of network throughput samples have been proposed in [19]. We think that the problem with these approaches is that most modern systems are characterized by complex hardware/software architectures and by highly variable workloads that cause high instability of system resource measures. The context of Internet-based system is characterized by distributed systems that are subject to typically non stationary loads, heavy-tailed distributions [3] and flash crowds [14], extreme variability and tendency to become obsolete rather quickly [10]. Hence, simple resource measures are of little help to distinguish overload conditions from transient peaks, to understand load trends and to anticipate future conditions, that are of utmost importance for taking correct runtime decisions and guaranteeing a robust self-adaptive system. In other fields, different studies [4, 11] have faced similar problems following the basic principles of *Qualitative Modeling* based on *trend*, aiming to represent the state and behavior of a physical system retaining only its mainly interesting features, without dealing with punctual and comprehensive overviews of it. Despite that, not only these previous works are all applied to different contexts and do not develop stochastic mod-



(a) Instantaneous values of the CPU utilization **(b) Instantaneous values of the Disk throughput** **(c) Average values of the CPU utilization**

Figure 1. State representations of some system resources.

els for trend definition, but they even handle the problem only in qualitative terms (for example, whether the derivative is increasing, decreasing or oscillating), without integrating them with quantitative measures.

We propose a mathematical model for the runtime evaluation of the behavioral *trend* of the internal resources load state, from a qualitative and also quantitative point of view. In this paper, we apply the proposed self-adaptive behavioral trend models to a classical request dispatching problem in the context of a locally distributed multi-tier Web system. Our experimental results show that load-aware dispatching decisions based on trend guarantee better stability of the load among the servers, and a reduction of the response time.

The paper is organized as following. Section 2 introduces the problem definition. Section 3 presents the self-adaptive qualitative and quantitative trend models. Section 4 discusses the experimental results. Section 5 concludes the paper with some final remarks.

2 Problem definition

The state of the internal system resources for load-aware decision systems is traditionally obtained through a periodic collection of measures from server monitors. Moreover, the observed measures of the internal resources are characterized by some statistical properties, such as noise, short time dependency [1] and non stationary effects, that make the observed values completely unreliable to characterize the system state with respect to the resource capacities. To give a qualitative motivation of the difficulties in capturing any clear message from a sequence of resource measures, in Figures 1 we report an example related to the back-end server of a multi-tier Web system subject to a realistic non-stationary workload. In these figures, we consider different metrics and representations:

- two resource metrics: CPU utilization (Figure 1(a)), and disk throughput as blocks/second (Figure 1(b));
- two load representations: instantaneous values (Figures 1(a) and 1(b)) and average values (Figures 1(c)).

Figures 1(a) and 1(b) share the common trait that the view of a resource that is obtained from system monitors is extremely variable. On the other hand, Figure 1(c) shows that a simple average measure reduces the variability of the resource state, but at the price of an unacceptable delay in the state representation [1]. The variability of the instantaneous view and the delay of the average view are high to the extent that any load-aware decision based on these values may be completely wrong. For example, for a decision system that should work on the CPU utilization measures similar to those in Figure 1(a) it would be impossible to judge when a server is under- or over-loaded. On the other hand, an average view of the resource measures mitigates the oscillations, but a self-adaptive algorithm would be significantly delayed for taking any correct decision.

In general, the traditional models for load representation are able to capture the information about the actual load state of a resource but anyone is able to adapt the load representation to the sudden changes of the resource behavior. Let us consider the i-th load state, S_i. A traditional load representation model describes S_i through a linear combination of the n past data:

$$S_i = \sum_{j=0}^{n-1} q_j x_{i-j}; \qquad \sum_{j=0}^{n-1} q_j = 1 \tag{1}$$

where $x_i, \ldots, x_{i-(n-1)}$ are the last n monitored data weighted by the q_j coefficients. Many different load representation models based on an instantaneous view

exist but in our paper we consider models based on the instantaneous view of the monitored measures and on the moving average of the last n data. In particular, if we consider the model based on the instantaneous measures, the load state S_i is equal to:

$$S_i = x_i \tag{2}$$

while, using the model based on the moving average, S_i is equal to:

$$S_i = \gamma x_i + (1-\gamma)S_{i-1}; \quad \gamma = \frac{2}{(1+n)} \tag{3}$$

where γ is the smoothing factor [15].

In literature there are many popular stochastic models that are able to reduce the limits of the considered models, such as auto-regressive integrated moving average models (ARIMA) and auto-regressive fractionally integrated moving average (ARFIMA) models [12]. These models are able to reproduce precisely the behavior of the data set, but at the price of an excessive computational cost and frequent updates of their parameters that are inadequate to be used at runtime in a context characterized by high variability. Hence, to support load-aware decision algorithms it is important to define a new load representation that is able to limit the drawbacks of the existing models in the specific context of Internet-based distributed systems.

3 Trend models

A robust load representation for supporting the load-aware decision algorithms should combine the linear model simplicity, the AR and ARIMA qualities of reproducing the stochastic pattern of the data set and the simple average model ability of smoothing some noise components.

Our idea is that a valid load representation model in a non stationary context should not consider just the actual values computed through the traditional model, but it should also be able to estimate the *behavioral trend* of the resource load. The behavioral trend gives a fundamental self-adaptive geometric interpretation of the load behavior that adapts itself to the non stationarity of the load and that can be utilized to evaluate whether the load state of a resource is increasing, decreasing, oscillating or stabilizing. Consequently, it is possible to generate a new load representation based on this geometric interpretation.

We define the *behavioral trend* as a function $T(X_n, m)$ that takes into account :

- the n data set values: $X_n = (x_{i-(n-1)}, \ldots, x_i)$;
- the m values of the observed data set X_n used for the behavioral trend evaluation, $m \leq n$.

and gives a geometric interpretation of the selected data. Hence, we can write:

$$T(X_n, m) = b_k \tag{4}$$

where $b_k \in \mathcal{B} = \{b_1, \ldots, b_K\}$ denotes the behavior trend associated to the m selected data. The behavior b_k may have a qualitative or a quantitative representation.

3.1 Qualitative trend model

The qualitative trend associates a sequence of m values to a specific pattern. We consider at least $m = 2$ selected points, x_i and $x_{i-\lfloor\frac{n}{m}\rfloor}$, where $\frac{n}{m}$ is the selection frequency of the points in the data set X_n. These values allow us to obtain three trend patterns $\mathcal{B}_{m=2} = \{increasing, decreasing, stable\}$, that is,

- *increasing* is the trend that is associated to an increment between the last two selected values, $x_i > x_{i-\lfloor\frac{n}{m}\rfloor}$;
- *stable* represents the trend characterized by $x_i = x_{i-\lfloor\frac{n}{m}\rfloor}$;
- *decreasing* is associated to consecutive values that have a descending tendency, $x_i < x_{i-\lfloor\frac{n}{m}\rfloor}$;

In general, if we consider m selected points, we are able to define 3^{m-1} different trend patterns. In this paper for $m > 2$ we consider five classes: $\mathcal{B}_{m>2} = \{increasing, unstable+, stable, unstable-, decreasing\}$, because preliminary evaluations demonstrated that a higher number of classes is useless. The considered five classes are defined as following:

- *increasing* characterizes a sequence of $m-1$ increasing behavioral trends, $x_{i-(j-1)\lfloor\frac{n}{m}\rfloor} > x_{i-j\lfloor\frac{n}{m}\rfloor}$ for $1 \leq j \leq m$;
- *unstable+* describes a sequence of oscillating behavioral trends that has the last one in an *increasing* mode, $x_i > x_{i-\lfloor\frac{n}{m}\rfloor}$;
- *stable* represents a stable condition, $x_{i-(j-1)\lfloor\frac{n}{m}\rfloor} = x_{i-j\lfloor\frac{n}{m}\rfloor}$ for $1 \leq j \leq m$;
- *unstable-* represents a sequence of oscillating behavioral trends where the last one is in a *decreasing* mode, $x_i < x_{i-\lfloor\frac{n}{m}\rfloor}$;
- *decreasing* characterizes a sequence of $m-1$ increasing behavioral trends, $x_{i-(j-1)\lfloor\frac{n}{m}\rfloor} < x_{i-j\lfloor\frac{n}{m}\rfloor}$ for $1 \leq j \leq m$;

Using only the qualitative trend to represent the i-th load state of a resource, we have:

$$S_i = b_k \tag{5}$$

3.2 Quantitative trend model

It is possible to pass from a qualitative representation of the behavioral trend to a quantitative view by evaluating the gradient of the segment that is obtained when $m = 2$ or by some linear combinations of the past values when $m > 2$. Let us consider this latter instance.

Between every pair of the m consecutive selected points in the vector X_n, we compute the *trend coefficient* α_j, with $0 \leq j \leq m-1$, of the line that divides the consecutive points $x_{i-j|\frac{n}{m}|}$ and $x_{i-(j+1)|\frac{n}{m}|}$.

$$\alpha_j = \frac{x_{i-j|\frac{n}{m}|} - x_{i-(j+1)|\frac{n}{m}|}}{|\frac{n}{m}|}; \quad 0 \leq j \leq m-1 \quad i < m \tag{6}$$

In order to quantify the degree of variation of the past data values, we consider a weighted linear regression of the m trend coefficients:

$$\bar{\alpha}_i = \sum_{j=0}^{m-1} p_j \alpha_j; \quad \sum_{j=0}^{m-1} p_j = 1 \tag{7}$$

where $\alpha_0, \ldots, \alpha_{(m-1)}$ are the trend coefficients that are weighted by the p_j coefficients. This is the most general formula that can pass from not weighted p_j values to weighted coefficients obtained through some decay distributions. In this paper, we consider a geometric distribution of the weights p that gives more importance to the most recent trend coefficients. The absolute value of the j-th trend coefficient $\mid \alpha_j \mid$ identifies the intensity of the variation between two consecutive measures $x_{i-j|\frac{n}{m}|}$ and $x_{i-(j+1)|\frac{n}{m}|}$. The sign of α_j denotes the direction of the variation: a plus represents an increase between the $x_{i-j|\frac{n}{m}|}$ and $x_{i-(j+1)|\frac{n}{m}|}$ values, while a minus denotes a decrease. A load representation of the i-th load state S_i based on only the quantitative trend model has the following expression:

$$S_i = \bar{\alpha}_i \tag{8}$$

It is possible to extend the quantitative behavioral trend model with the information about the actual load state shown in Equation 1. In this self-adaptive model, that we name *quantitative trend with load representation* model, the i-th load state, S_i, is the result of a linear combination between the quantitative trend, $\bar{\alpha}_i$, and the load representation value, that is:

$$S_i = \bar{\alpha}_i + \sum_{j=0}^{n-1} q_j x_{i-j}; \quad \sum_{j=0}^{n-1} q_j = 1 \tag{9}$$

4 Experimental results

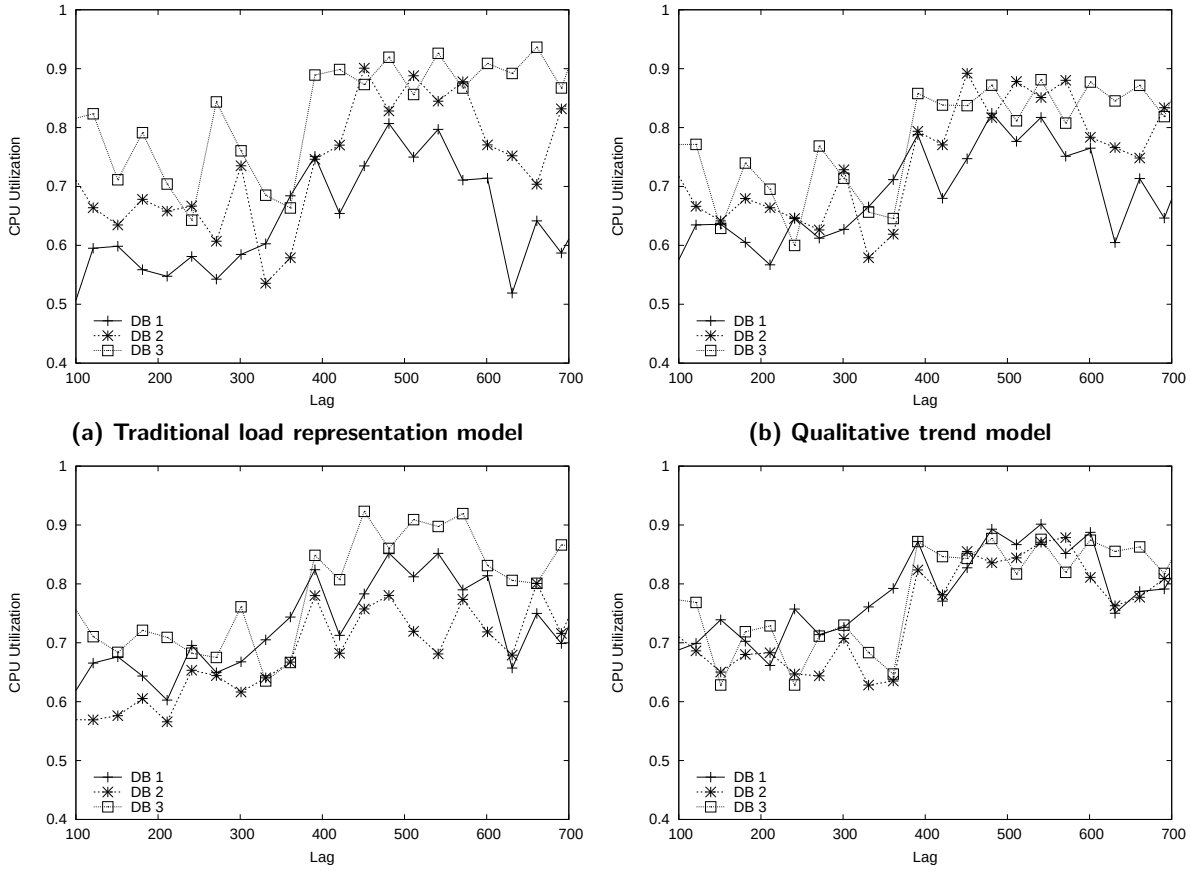
The architecture that we use as a testbed for the experiments is a typical multi-tier Web architecture that is based on the implementation presented in [6]; the application servers are deployed through the Tomcat servlet container, and are connected to MySQL database servers. In our experiments, we exercise the system through realistic traces; each experiment lasts one hour. To manage large numbers of requests (possibly hotspots) among these locally distributed systems the front-end switch requires smart policies that must take fast decisions at runtime. We start from a popular load balancer where dispatching decisions are based on the weighted round robin (WRR) policy [13] where the weights are computed on the bases of the actual load representation. We then apply the trend models of Section 3 and consider that the weights are computed by means of different self-adaptive trend-aware representations: *qualitative trend* models when only the qualitative behavioral trend is used, *quantitative trend* models that consider only the quantitative behavioral trend and the *quantitative trend model with load representation* models that combine the quantitative trend with the moving average model as load representation. In the presented experiments the trend-aware models consider three past points that is, $m = 3$. Moreover, we consider two different traditional models: the first one represents the load state using directly the instantaneous measures, x_i; and the second is based on the moving average of the last $n = 10$ data.

We evaluate two important performance factors of dispatching: the impact on the response time by considering the *90-percentile* of the response time for an entire Web request, and the level of load balancing among the servers through the *Load Balance Metric* [5] (LBM). Let us define the load state of the server j at the i-th observation of the observation period p as $load_{j,i}$, and $peak\_load_i$ as the highest load in the same observation. The LBM is defined as:

$$LBM = \frac{\sum\limits_{1 \leq i \leq p} peak\_load_i}{(\sum\limits_{1 \leq i \leq p} \sum\limits_{1 \leq j \leq N} load_{j,i})/N} \tag{10}$$

where N is the total number of servers. Note that the value of the LBM can range from 1 to the number of servers ($N = 3$ in the considered system). Smaller values of the LBM indicate a better load balance.

The qualitative effects of the different load representation models into the dispatching algorithms based on the CPU utilization of the three servers can be appreciated from the Figures 2. The servers are highly



(a) Traditional load representation model

(b) Qualitative trend model

(c) Quantitative trend model

(d) Quantitative trend model with load representation

Figure 2. Load balancing in the distributed system.

Table 1. Evaluation of the models.

Support for load-aware decision algorithms	LBM	90-perc. response time
Measure-aware model	2.15	501ms
Moving average model	1.85	450ms
Qualitative trend model	1.40	417ms
Quantitative trend model	1.41	410ms
Quantitative trend model with load representation	1.29	370ms

unbalanced, when the dispatching uses the load representation based on moving average (Figure 2 (a)). On the other hand, the self-adaptive representations based on the behavioral trend bring to a more balanced system, as it is highlighted by the close curves of the CPU utilizations of the three DB servers (Figure 2 (b-d)). This result is important because lasts during the entire experiment.

In Table 1 we report a quantitative summary of the main performance metrics for the considered models. From this table, we can see that the dispatcher based on the *quantitative trend model with load representation* achieves the best load balance in terms of LBM and the best performance in terms of 90-percentile of the response time. The difference between the LBM values increases considerably when the dispatching scheme is based on traditional models or when it is based on the *quantitative trend model with load representation* model. Shifting from the model based on instantaneous measures to the *qualitative trend* model, the LBM difference is about 29% (17% if the comparison starts with the model based on moving average), while if we shift from the measure-aware model to the *quantitative trend model with load representation* model, the increase in LBM values is more than 40% (30% if the comparison starts with the model based on moving average). The 90-percentile of the response time passes from 501ms, when the load-aware decisions are based on the model based on instantaneous values, to 370ms, when the load-aware decision algorithm

uses the *quantitative trend model with load representation* model. This reduction of the 90-percentile of the response time confirms the ability of the qualitative trend to manage a complex distributed system even in the presence of a severe non stationary workload.

5 Conclusion

The load-aware decision systems are expected to guaranty robust results even under different workload conditions even critical in terms of variability and peaks of requests. These external characteristics together with the increasing complexity of the internal architecture of distributed systems make management decisions really difficult especially for self-adaptive systems. This paper represents a first contribution in the direction of investigating a new class of representations of the resource states that are based on a trend evaluation of the load. We demonstrate that the proposed self-adaptive trend-aware models are able to improve the traditional load-aware decision systems, such as a front-end dispatcher of a Web cluster.

Acknowledgements

The first author acknowledges the support of the Fondazione Cassa di Risparmio di Modena, Contact 412/0/.8C, March 2008.

References

[1] M. Andreolini, S. Casolari, and M. Colajanni. Models and framework for supporting run-time decisions in web-based systems. *ACM Tran. on the Web*, 2(3), Aug 2008.

[2] J. Bahi, S. Contassot-Vivier, and R. Couturier. Dynamic load balancing and efficient load estimators for asynchronous iterative algorithms. *IEEE Trans. Parallel and Distributed Systems*, 16(4):289–299, Apr. 2006.

[3] P. Barford and M. E. Crovella. Generating representative Web workloads for network and server performance evaluation. In *Proc. of 1st the Joint International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS 1998/Performance 1998)*, Madison, WI, July 1998.

[4] B. Bredeweg and P. Struss. Current topics in qualitative reasoning. *AI Magazine*, 24(4), 2003.

[5] R. B. Bunt, D. L. Eager, G. M. Oster, and C. L. Williamson. Achieving load balance and effective caching in clustered Web servers. In *Proc. of the 4th International Web Caching Workshop*, San Diego, CA, USA, USA, Apr. 1999.

[6] H. W. Cain, R. Rajwar, M. Marden, and M. H. Lipasti. An architectural evaluation of Java TPC-W. In *Proc. of the 7th Intl. Symposium on High-Performance Computer Architecture (HPCA2001)*, Nuovo Leone, Mexico, Jan. 2001.

[7] V. Cardellini, M. Colajanni, and P. Yu. Request redirection algorithms for distributed Web systems. *IEEE Trans. Parallel and Distributed Systems*, 14(5):355–368, May 2003.

[8] X. Chen and J. Heidemann. Flash crowd mitigation via an adaptive admission control based on application-level observations. *IEEE Trans. Internet Technology*, 5(3):532–569, Aug 2005.

[9] L. Cherkasova and P. Phaal. Session based admission control: A mechanism for improving performance of commercial Web sites. In *Proc. of 7th Intl. Workshop on Quality of Service (IWQoS 1999)*, pages 226–235, London, UK, June 1999.

[10] M. Dahlin. Interpreting stale load information. *IEEE Trans. Parallel and Distributed Systems*, 11(10):1033–1047, Oct. 2000.

[11] R. M. P. de Alcantara, G. M. da Nobrega, and P. Salles. Towards the use of qualitative reasoning for supporting information technology management. In *Proceedings of the 20th Workshop on Qualitative Reasoning (QR 2006)*.

[12] P. Dinda and D. O'Hallaron. Host load prediction using linear models. *Cluster Computing*, 3(4):265–280, Dec. 2000.

[13] G. Hunt, G. Goldszmidt, R. King, and R. Mukherjee. Network web switch: A connection router for scalable internet services. In *Proc. of 7th Int. World Wide Web Conf.*, Brisbane, Australia, Apr. 1998.

[14] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites. In *Proc. of 11th Int'l World Wide Web Conference (WWW2002)*, Honolulu, HW, May 2002.

[15] D. J. Lilja. *Measuring computer performance. A practitioner's guide*. Cambridge University Press, 2000.

[16] D. Menascé and J. Kephart. Autonomic computing. *IEEE Internet Computing*, 11(1):18–21, Jan. 2007.

[17] M. Mitzenmacher. How useful is old information. *IEEE Trans. Parallel and Distributed Systems*, 11(1):6–20, Jan. 2000.

[18] G. Pierre and M. Van Steen. Globule: a platform for self replicating Web documents. In *Proc. of the 6th Conference on Protocols for Multimedia systems (PROMS 2001)*, Enschede, The Netherlands, 2001.

[19] A. Sang and S. Li. A predictability analysis of network traffic. In *Proc. of the 19th IEEE Intl. Conference on Computer Communications (INFOCOM 2000)*, Tel Aviv, Israel, 2000.