Improved Performance of Computer Networks by Embedded Pattern Detection

Angel Kuri-Morales[1] and Iván Cortés-Arce[2]

[1] Instituto Tecnológico Autónomo de México, ITAM, D.F., México
akuri@itam.mx
[2] Universidad Nacional Autónoma de México, IIMAS, D.F., México
ivan.eduardo.uzzy@gmail.com

Abstract. Computer Networks are usually balanced appealing to personal experience and heuristics, without taking advantage of the behavioral patterns embedded in their operation. In this work we report the application of tools of computational intelligence to find such patterns and take advantage of them to improve the network's performance. The traditional traffic flow for Computer Network is improved by the concatenated use of the following "tools": a) Applying intelligent agents, b) Forecasting the traffic flow of the network via Multi-Layer Perceptrons (MLP) and c) Optimizing the forecasted network's parameters with a genetic algorithm. We discuss the implementation and experimentally show that every consecutive new tool introduced improves the behavior of the network. This incremental improvement can be explained from the characterization of the network's dynamics as a set of emerging patterns in time.

Keywords: Load Balancing, Computer Networks, Intelligent Agents, Neural Networks, Genetic Algorithms.

1 Introduction

Computer networks are frequently balanced appealing to heuristics or, simply, the experience of the administrator of the network [5, ch. 1], [27, ch. 3]. This typical practice suffers from the severe inconvenient of not taking into consideration the behavioral patterns which may be discovered if the appropriate tools are used. To begin with, it is very convenient to consider a dynamic link between the elements of the network. If this is achieved, it is possible to apply machine learning and optimization techniques to improve their performance. This realization is the main motivation to merge, in one scheme, the following tools: statistical protocol adaptation, intelligent agents, MLPs and genetic algorithms (GA). The main purpose is to improve the channel utilization and all related computer network resources. It should also consider the impact of these tools under *steady* and *error regime* conditions (see below). It is true that there are solutions [1], [9] based on well-known algorithms [2], [3], [4]. These,

J.F. Martínez-Trinidad et al. (Eds.): MCPR 2014, LNCS 8495, pp. 21–30, 2014.

however, do not consider congestion problems. The Traffic Engineering Performance Objectives are described in [10]. They are either 1) Traffic oriented or 2) Resource oriented. The key traffic oriented performance objectives include minimization of packet loss, minimization of delay and maximization of throughput. Minimization of packet loss is one of the most important traffic oriented performance objectives. The routing algorithms typically consider link state [3] or distance vector [2] to select routing paths based on static link weights or costs. These, however, do not allow us to use all alternate and available paths leading to the same destination. This fact increases the probability of congestion traffic as mentioned in [1] since the routing algorithm does not change in time as the traffic flow does. In this paper we will prove that the use of the appropriate computational intelligence tools minimizes packet loss and maximizes channel utilization. To achieve this, the behavior of the traffic flow along the full network must be known. Such global knowledge allows us, in principle, to attempt accurate forecasting. If this is achieved then it is possible to establish the best values of the variables involved such that the traffic is maximized. We have selected an implementation of the Bellman-Ford distance vector algorithm [2, 5] as a benchmarking standard with which to compare our algorithm and we have set the minimization of the number of lost packets as the routing metric. In Section 2, the necessary concepts and definitions are introduced; in Section 3 we present a brief explanation of the method applied. In section 4 we present the experimental results. Finally in section 5 we present our conclusions.

2 Preliminaries

Suppose that there is a computer network in which there are N routing devices and L links which correspond to the communication channels. We denote the sources of the traffic by G; d is the bandwidth of the channel; p is the packet size; q_t is the percentage utilization of the channel and t is the number of the sample; λ is the network traffic flow given in packets per second [24]; r_n is the *n-th* adaptive routing device; $g_k(\lambda)$ is the *k-th* input traffic source; l_k is the number of links associated with r_n; $l_k^{(i)}$ is the input traffic flow of channel k; $l_k^{(o)}$ is the output traffic flow of channel k; $l_{g(k)}^{(i)}$ is the input traffic flow of the channel which connects r_k to $g_k(\lambda)$; $l_k^{(i)}, l_k^{(o)} \in l_k$ where i is the input and o is the output; $\sum \rho_n(\omega)^{(i)}$, $\sum \rho_n(\omega)^{(o)}$ are, respectively, the sums of all optimized input and output traffic flows on device r_i; w_0, w_1 are the linear regression coefficients of the forecasted channel's utilization in time. *S(p)* is the single path routing algorithm; $M(p)$ is the adaptive path routing algorithm.

3 Adaptive Process on Computer Network

The adaptive model's performance is incrementally improved through consecutive application of: a) Agents [17, 22], b) MLPs [18] and c) GAs [14]. Along this paper, the results refer to the network illustrated in Fig. 1:

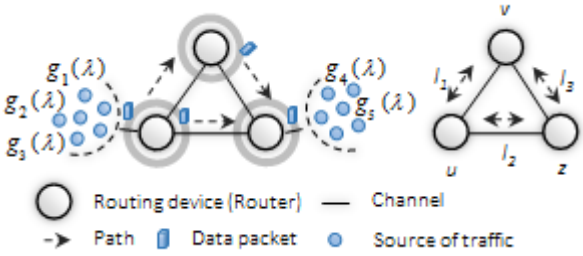

Fig. 1. Schematic architecture of the Computer Network

From Figure 1 we have: $G = \{g_1(\lambda), g_2(\lambda), g_3(\lambda), g_4(\lambda), g_5(\lambda)\}$; $N = \{r_u, r_v, r_z\}$; $L = \{l_1, l_2, l_3\}$; every r_i on the network is fully connected by a link l_k, $r_{u,z}$ are connected to $g_k(\lambda)$ by link $l^{(i)}_{g(k)}$; $r_{u,z}$ are the external nodes connected with $g_k(\lambda)$; r_v is an internal node. For convenience we use $M(p)$ instead of $S(p)$ [2], [5] (since $S(p)$ does not allow the *ad hoc* use of all available paths). $S(p)$, thus, increases the probability of traffic congestion. In our model each r_i is provided with tools such as: a) $A_{(r)}$ [set of intelligent agents (daemons) running in r_i]. $A_{(r)} = \{a_1,...,a_k\}$; $k = |A|$ where a_k is the *k-th* agent in r_i; b) *NN* is a MLP and c) *ega* which is the GA-based [26] optimization algorithm. The use of the tools and adoption of $M(p)$ allows us to consecutively improve the network's performance. In what follows we will describe the different tools applied in every stage.

3.1 Agents

The initial adaptive model implements *wlb*, a simple initial statistical load balancing method which distributes traffic to all routing nodes. When r_i detects traffic, $A_{(r)}$ is started; before this, the agents have not been active. It is here that they get ready to sample the channel's behavior. This they do until they have enough data, at which point an optimization task is started. Each r_i on the network works as an autonomous entity which takes its own decisions derived from the knowledge other agents give. Channel's sampling is described by Equation (1):

$$A(s)_r = \{u_1(t), u_2(t), ..., u_n(t)\} \tag{1}$$

Where $u_i(t) = \{q_1,.., q_n\}$; $u_i(t)$ is resource sampled in r_i,. Once $A_{(r)}$ has sufficiently sampled the channel (as per $s = \{u_1(t), u_2(t),\ldots,u_n(t)\}$) it forecasts the future channel's utilization from linear extrapolation, whose detailed equations are:

$$w_0 \leftarrow w_0 + \alpha \sum_i [u_i - f_w(t_i)]; \quad w_1 \leftarrow w_1 + \alpha \sum_i [u_i - f_w(t_i)] \times t_i$$

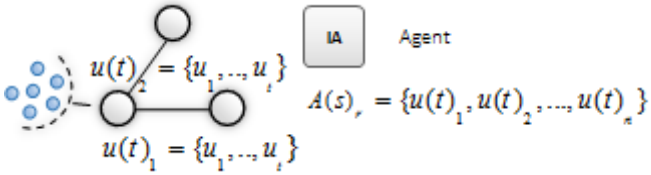

Fig. 2. Example of an Agent's sampling process

In Figure 2 we illustrate how the adaptive model based on linear prediction acts on ri to find a more efficient load balancing via simple statistical load balancing. The next step of the adaptive model is based on MLPs.

3.2 MLP's

Our aim is to improve on the basic linear prediction by discovering the behavioral patterns in the data of the channel. It is well known [25] that MLPs are universal approximators which, as argued in [18], are accurate regardless of the probability distribution function of the variables under observation. If we want to get the future values to prevent potential network traffic problems, MLPs are a proven reliable choice. Therefore, once enough data have accumulated, we are able to take ad-vantage of the embedded behavioral patterns (uncovered by the MLP) and replace linear forecast with the superior MLP forecast. We denote our BMLP (Backpropagation Multi-Layer Perceptron) model with $NN_{(i)}$. In Figure 3 we illustrate a possible MLP architecture. In our model each adaptive routing device has an associated MLP to perform forecasting. Given samples $s = \{u_1(t), u_2(t),\ldots,u_n(t)\}$ provided by each $A_{(r)}$ in a previous time step, we want to find $U(\tau) = \{u_1(t), u_2(t),\ldots,u_n(\tau)\}$ (described in [13]) such that:

$$u_i(\tau) = f_i(u_i(t), ..., u_n(t)) \quad \text{for } i = 1, .., n \tag{2}$$

$\tau = t + k$ where t represents time and k is time displacement relative to future period to characterize. Forecasted values will be denoted by $u'_i(\tau)$. Such values will have an expected approximation error ε_i such that $u'_i(\tau) = u_i(\tau) \pm \varepsilon_i$

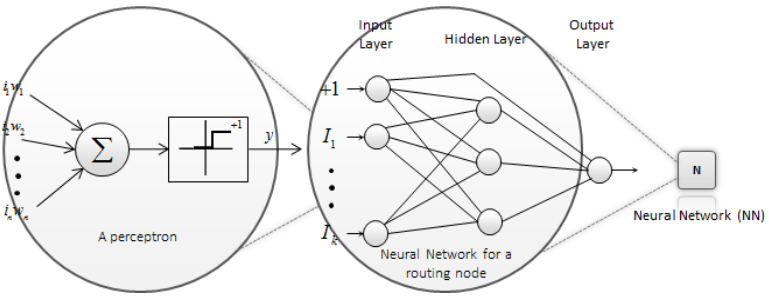

Fig. 3. A Neural Network Model

Taking in consideration all of above we denote the forecasted values with $u'_i(\tau)$ and

$$u'_i(\tau) = NN_{(i)}(u_1(t), ..., u_n(t)) \tag{3}$$

MLP's error is $\varepsilon(NN_{(i)})_t$. Then we can compare it with the previous one $\varepsilon(LR_{(i)})_t$ found by $A_{(r)}$ *without* an MLP. $A_{(r)}$ selects the smaller one for every r_i.

3.3 Genetic Algorithm

EGA [26] gets values from $u'_i(\tau)$ to optimize traffic flow. For convenience, traffic flow is given in mbps (megabits per second). In Fig. 4 is shown a typical genome for this application.

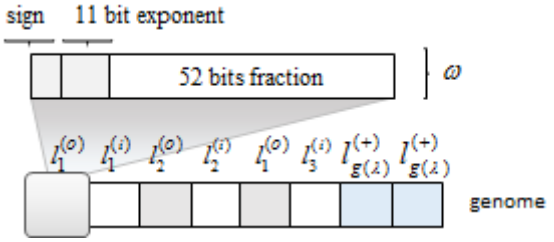

Fig. 4. Representation of the Genome

Henceforth we use the following equation to transform $u'_i(\tau)$ into suitable units.

$$\omega(u'_i(\tau)) = \sum_{i=1}^{\tau} \left(u'_i(\tau) \times r(l_k)\right) \tag{4}$$

Where ω is the traffic flow at time i; r is the bandwidth l_k and (4) allows us to compute forecasted values at time τ. To exemplify, assume that our fitness function (which we want to maximize) is determined by $ega(\omega)$ and has the following form:

$$ega(\omega) = max\left(\sum_{n=1}^{N} \left(\sum \rho_n(\omega)^{(i)} - \sum \rho_n(\omega)^{(o)}\right)\right) \tag{5}$$

In Equation (5) (Fitness function) i and o represent the incoming and outgoing channels. When $ega(\omega)$ has optimized from the forecasted values it supplies them to A(r) which will communicate parameter's values to all the agents.

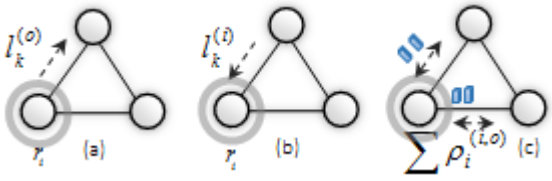

Fig. 5. a) Output Traffic Flow, b) Input Traffic Flow, c) Accumulated Traffic Flow

Figures 5a, 5b, 5c show how the adaptive model optimizes using all available resources from the network. The learning process can be summarized as in Figure 6 which shows the adaptive model with all the stages working in unison.

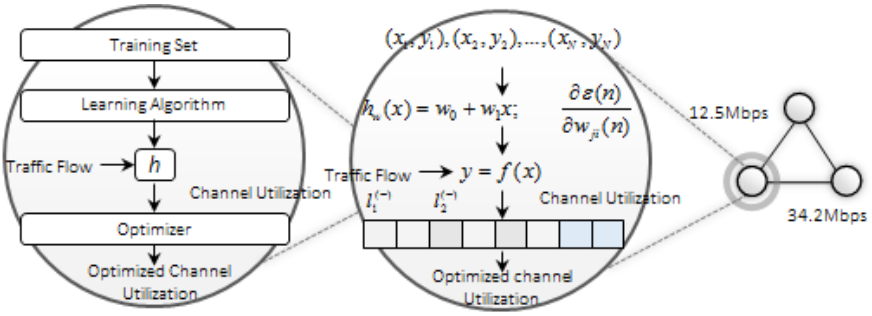

Fig. 6. Adaptive Routing with all stages working in unison

4 Results

We have performed experiments which show that the performance of our model is better than typical ones under similar conditions [2], [3], [5]. We have extensively simulated the behavior of various networks on OMNET++ [23]. The behavior of the system was performed for 25 simulated seconds (corresponding to processing ~57,000 packets; roughly 2,300 packets per second). By *Steady regime* we mean that the network is stable (no errors occur); while the *Error regime* corresponds component errors being considered.

Packet loss may occur from network congestion and overflow at queues in routers and/or switches. It results in wasted bandwidth. Recalling that one of the key traffic oriented performance objectives is the minimization of packet loss, the evident superiority of our method is shown in Table 1.

Table 1. Difference betweeen packets generated and packets arrived

Routing Strategy	Packet Received	Relative Loss	Packet Lost	Relative Efficiency
Distance Vector	39,662	23.56%	17,544	46.75%
Adaptive with Agents	40,257	21.73%	16,949	48.39%
Adaptive with MLP	41,088	19.27	16,118	50.88%
Adaptive with EGA	49,005	0.00%	8,201	100.00%

Notice the outstanding improvement resulting from optimal routing assignment which is achieved from a) The values delivered by the EGA; this attests to b) The successful forecast which the MLPs yield; c) Clearly, were it not for the efficient update of the network's status (via Agents) the MLP-EGA stages would not perform adequately.

If errors (due to assumed malfunction of the components) are included, the behavior of the network varies accordingly. To this effect, exponentially distributed

component failures were induced. Error recovery times were also assumed exponentially distributed, although the parameters of the exponential distributions for both error and recovery times were different for every router. A performance comparison of the network in steady and error regimes is shown in Table 2.

Table 2. Comparative of the Routing Protocol Performance

Routing Strategy	Packet Lost (Stable Regime)	Relative Efficiency (Error regime)
Distance Vector	4,355	21,854
Adaptive with Agents	2,911	5,427
Adaptive with MLP	0	3,101
Adaptive with EGA	0	3,101

The experiments based on the adaptive model display a degradation both when MLPs and/or GAs are included if a relatively large number of network failures is simulated. This is due to the fact that the underlying patterns (which are the basis for adequate forecasting) tend to vanish because of the simulated errors. However, our method automatically folds-back to the simple routing algorithm in such a case. We have also included a graph that shows how load balancing methods are ap-proaching the optimal one with our proposed model. Optimality was calculated by exhaustive enumerative analysis.

Figure 7 illustrates a classification of the number of lost packets.

Several comparisons between the network's behavior under the different options are possible. For reasons of space we are unable to annotate them all. We wish to point out, however, that introducing agents (and, thus, making the global information available to all routers) yields immediate benefits. For example, let us consider node *u* when connected to a 34.2 Mbps channel (l1) and simultaneously connected to a 12.5 Mbps channel (l2). Two alternative methods are considered: a) Statistical (no agents) and b) Agents. With method (a) 2.91 Mbps are transported on l1 and 6.85 Mbps on l2; with method (b) 68.76 Mbps are transported on l1 and 179.91 on l2. This is a remarkable improvement of 23.54:1 for l1 and 26.26:1 for l2 when agents are considered.

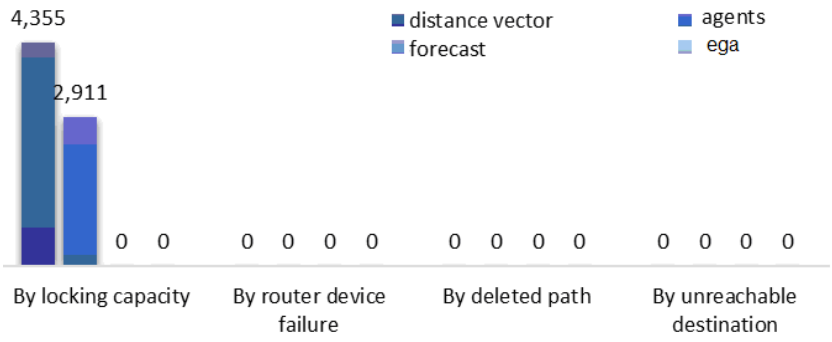

Fig. 7. Routing strategies classifications (Steady Regime)

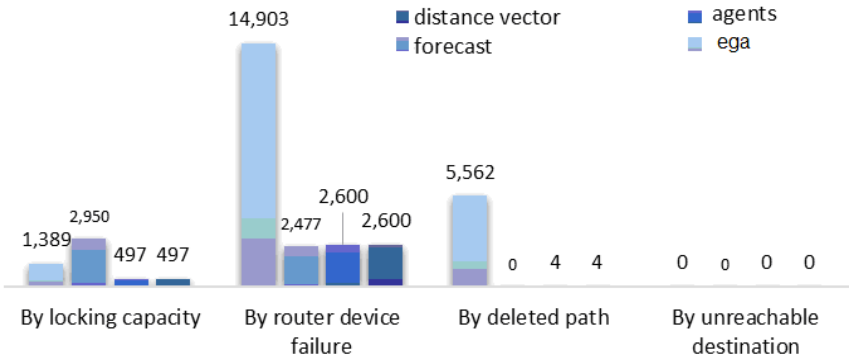

Fig. 8. Routing strategies classification (with errors)

Table 3. Comparative among Load Balancing Methods

Method	34.2Mbps Transported Load (%)	12.5Mbps Transported Load (%)
Theoretical load balancing	0.7323	0.2670
Statistical load balancing	0.7015	0.2984
Intelligent Load balancing	0.7234	0.2765

5 Conclusions

We used a new approach to improve a computer's network resources. We worked with a) Agents, b) Neural Networks, c) Genetic Algorithms. We have shown that our model minimizes the number of packets lost, increasing the network's performance. We have shown that every time that a new component is added to the adaptive model, it further improves the network's performance. On the other hand, when fail-ures are considered, the MLP attempting to forecast from the network's past behavior is unable to uncover (the non-existent) patterns. Regardless of this, the adaptive model still behaves reasonably well because agents are still effective in attempting to globally optimize the system.

We have presented preliminary simulation results based on a rather simple architecture. But the principles are the same regardless of the network's complexity. In fact, more complex architectures are prone to exhibit more intricate patterns and the tools we implemented have been repeatedly shown to be very efficient in detecting patterns in complex environments.

This is a prime example of computational intelligence working in an on-line simulation environment. Given all of the above considerations we think that reliable forecasting and optimization can be improved working, to be sure, with non-chaotic data. A very practical conclusion is that it is possible to improve traffic flow at no software/hardware cost when network balance is based on this approach.

References

1. Mahlous, A.: Multipath Routing Using Max Flow Algorithms For Internet Traffic: Development and Performance Evaluation of Max Flow Multipath Routing Schemes for the Internet using Max Flow Algorithms. Lambert Academic Publishing (2009)
2. Bellman, R.: On A Routing Problem. RAND Corporation, Santa Monica (1958)
3. Dijkstra, E.: A note on Two Problems in Connexion with Graphs. Numerische Mathmaik 1, 269–271 (1959)
4. Aceves, J.: Loop Free Routing Using Diffusing Computations. IEEE/ACM Transactions on Networking 1(1) (1993)
5. Kurose, J., Ross, K.: Computer Networking: A Top Down Approach, pp. 374–387. Addison Wesley (2010)
6. Awdche, D., Malcom, J., O'Dell, M., MacManus, J.: Requirements for Traffic Engineering Over MPLS. RFC 2702. Internet Society (1999)
7. Akami Technologies, Internet Bottlenecks: The Case for Edge Delivery Services (2000), http://www.cse.cuhk.edu.hk/~cslui/CSC5480/akamai-bottlenecks.pdf
8. Sinha, R., Papadopoulus, C., Heidenmann, J.: Internet Packet Size Distributions: Some Observations. University of Southern California (2005)
9. Ahmad, I., Ghafoor, A., Mehrotra, K., Mohan, C., Ranka, S.: Performance modeling of load balancing algorithms using neural networks. L.C. Smith College of Engineering and Computer Science. Syracuse University (1994)
10. Awdche, D., Malcom, J., O'Dell, M., MacManus, J.: Requirements for Traffic Engineering Over MPLS. RFC 2702. Internet Society (1999)
11. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1996)
12. Kuri-Morales, Á.F.: Pattern Recognition via Vasconcelos' Genetic Algorithm. In: Sanfeliu, A., Martínez Trinidad, J.F., Carrasco Ochoa, J.A. (eds.) CIARP 2004. LNCS, vol. 3287, pp. 328–335. Springer, Heidelberg (2004)
13. Kuri-Morales, A.: Application of a Method Based on Computational Intelligence for the Optimization of Resources Determined from Multivariable Phenomena. In: Batyrshin, I., Mendoza, M.G. (eds.) MICAI 2012, Part II. LNCS, vol. 7630, pp. 292–303. Springer, Heidelberg (2013)
14. Kuri-Morales, A.: A Universal Eclectic Genetic Algorithm for Constrained Optimization. In: Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT 1998 (1998)
15. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley (1989)
16. Anand, V., Spears, W.: A Study of Crossover Operators in Genetic Programming. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1991. LNCS, vol. 542, pp. 409–418. Springer, Heidelberg (1991)
17. Russell, S., Norving, P., Artificial Intelligence: A Modern Approach. Prentice Hall (2010)
18. Haykin, S.: Neural Networks.: A comprehensive Foundation. Prentice Hall (1999)
19. Stallings, W.: Queuing Analysis Notes (2013), http://www.computersciencestudent.com/
20. Kleinrock, L.: On modeling and Analysis of Computer Networks. Proceedings of the IEEE 81(8) (August 1993)
21. Hertz, J., Krogh, A., Palmer, R.: Introduction to the Theory of Neural Computation. A Lecture Notes Editorial Board (1990)

22. Wooldridge, M., Jennings, N.: Intelligent Agents: theory and practice. The Knowledge Engineering Review 10(2), 115–152 (1995)
23. Vargas, A.: OMNeT++ User Manual Version 4.3.1. OpenSim Ldt. (2011)
24. Sinha, R., Papadopoulos, C., Heidemann, J.: Internet packet size distributions: Some observations. USC/Information Sciences Inst (2007), http://netweb.usc.edu/~rsinha/pkt-sizes
25. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(5), 359–366 (1989)
26. Kuri-Morales, A., Aldana-Bobadilla, E.: The Best Genetic Algorithm Part I. In: Castro, F., Gelbukh, A., González, M. (eds.) MICAI 2013, Part II. LNCS, vol. 8266, pp. 1–15. Springer, Heidelberg (2013)
27. Graziani, R., Johnson, A.: Routing Protocols and Concepts, CCNA exploration companion guide. Cisco Press (2007)