Improving the Quality of Service of Failure Detectors with SNMP and Artificial Neural Networks

Raimundo José de Araújo Macêdo, Fábio Ramon Lima e Lima

Laboratório de Sistemas Distribuídos – LaSiD
Departamento de Ciência da Computação
Universidade Federal da Bahia
Campus de Ondina, CEP: 40170-110, Salvador-BA, Brazil
`{macedo,framon}ufba.br`

Abstract. *A failure detector is an important building block for fault-tolerant distributed systems: mechanisms such as distributed consensus and group communication rely on the information provided by failure detectors in order to make progress and terminate. As such, erroneous information provided by the failure detection may delay decision-making or lead the upper-layer mechanism to take incorrect decisions (e.g., excluding correct processes from a group membership). In this paper we explore the use of artificial neural networks to improve the quality of service (QoS) of failure detectors in settings where message transmission delays and system loads can vary over time, such as the Internet. The patterns used to feed the neural network were obtained by using Simple Network Management Protocol (SNMP) agents. Our detector was fully implemented and tested over a set of LINUX networked workstations, and we have run several experiments to evaluate our approach, comparing it against well-known implementations. The experiments show that our failure detector outperformed existing approaches.*

1. Introduction

Failure detection is an important issue for dependable distributed systems. In particular, in the partial synchronous model of the unreliable failure detectors introduced by Chandra-Toueg [1,2], it was presented for the first time a formal definition of failure detectors and related conditions for solving fundamental problems of fault-tolerant computing in such a model. Due to load variations, both in the communication links and runtime system, a failure detector can be too *slo*w, that is, it may take too much time to suspect a crashed process, and it can make *mistake*s by erroneously suspecting some process that is actually operational. To be useful, however, a failure detector has to be reasonably accurate (i.e., must avoid wrong suspicions) and fast (i.e., must avoid unnecessary delays in suspecting failures). In order to evaluate how fast and accurate a failure detector is, Chen, Toueg, and Aguilera proposed the following QoS metrics, and showed how they can fully qualify the service of failure detectors: ***detection tim***e, which defines the failure detector's speed; ***mistake recurrence time***, which defines the time between two consecutive mistakes; and ***mistake duration***, which defines the time it takes the failure detector to correct a mistake [3].

With the aim of improving the *QoS* of failure detectors, many authors have proposed different techniques to dynamically estimate the timeout values used in failure detection. These techniques are either based on the probabilistic behaviour of the system

[3] or on monitoring data for control message transmission delays [4,5,11], or even a combination of both [7]. To the best of our knowledge, two previous works provided implementations and related *QoS* performance analyses for their failure detectors based on the aforementioned metrics. Chen, Toueg, and Aguillera presented implementations for failure detectors that rely on clock synchronization and a probabilistic behaviour of the system [3]. The main drawback of their solution is that a probabilistic distribution function has to be defined to the environment before an estimation can be established. Bertier, Marin, and Sans [7] extended the failure detector developed by Chen, Toueg, and Aguillera, by introducing a safety margin dynamically calculated according Jacbson's algorithm [6], which resulted in a detector with a better detection time average. None of the existing work has explored the use of neural networks.

2. The Neural Network based Adaptive Failure Detector (NN-AFD)

We consider the model of *partial synchrony* proposed by Chandra and Toueg in [1], which defines that, among other things, in every execution, there are bounds on process speeds and on message transmission times, but these bounds are not known and they hold only after some unknown time (the Global Stabilization Time – GST). It is assumed that processes will be sending heartbeat messages with a time interval, named *HP* (*heartbeat* period), between the emissions of two consecutive *heartbeats*. To monitor a process *q*, a process *p* uses an estimated value, named *TO* (timeout), that tells *p* how much time it has to wait for the *heartbeat* from *q*. We propose a failure detector that is adaptive with respect to the current system and communication loads, and we use a neural network to achieve the required adaptation. We use a special kind of Neural Network, called *Feedforward Multilayer Perceptron* (MLP), and have modelled it with four layers: an input layer with six neurons corresponding the collected data from the MIB - Management Information Base [9]; two middle ones, one layer with nine and the other with four neurons; and an output layer with a single neuron, the estimated timeout for the arrival of the next *heartbeat* message. Adaptation of NN-AFD is achieved in two phases, namely the training and estimation phases. In the first phase, the Neural Network is trained through the *backpropagation* algorithm [13] to associate a given system load with the timeout value for the arrival of the corresponding *heartbeat* message. The system load is obtained by SNMP agents over the local MIB variables *IfInUcastPkts, ifOutUcastPkts, ifOutQLen, udpInDatagrams, udpOutDatagrams,* and *udpNoPorts.* In the second phase, the neural network is queried to estimate the arrival time of the next *heartbeat* for a particular pattern of the MIB. In both phases, the same time interval between heartbeat messages is used.

3. The QoS Analysis of NN-AFD

To allow a comparative analysis of the NN-AFD performance, we also implemented in our system the Bertier-Marin-Sens detector [7], which has been shown to perform better than the Chen-Toueg-Aguilera detector [3,10]. Bertier-Marin-Sens detector is a combination of Chen-Toueg-Aguilera detector and the Jacobson's estimation (used in the protocol TCP to estimate the delay after which a node retransmits its last message [6]). Thus, to calculate the estimation for the next *heartbeat* arrival time, named $\tau_{(k+1)}$, the Bertier-Marin Sens' estimation is calculated by adding the Chen-Toueg-Aguilera's estimation, $EA_{(k+1)}$, to a safety margin given by the Jacobson's estimation, $\alpha_{(k+1)}$. That is $\tau_{(k+1)} = \alpha_{(k+1)} + EA_{(k+1)}$.

We have implemented two versions of our failure detector. The first one is purely based on the neural network. The second version combines the pure NN-AFD with the Bertier-Marin-Sens detector, switching between both detectors depending on the network load pattern variation. To evaluate our detectors, we carried out two kinds of experiments: one with the ordinary network load and another one with an extra load introduced randomly by a third process, named the *overloader*. Thus, for every minute, during a period of 10 seconds, randomly chosen, the *overloader* continuously transmitted messages. In the remaining seconds of this particular minute, the *overloader* transmits only 30 messages per second. Experiments were performed at the Distributed Systems Laboratory (LaSiD) over three Pentium-III 800 Mhz hosts over an Ethernet 10/100 Base-T network.

The dynamic estimation of the Bertier-Marin-Sens detector was parameterised, in all experiments, as given in [7]. Due to space limitations, below we only discuss the results concerning the experiments with communication overload (a complete analysis can be found elsewhere [12]). These experiments were carried out for 5 rounds. In each round, after sending 1000 *heartbeats* as required by the adaptation phase of the Bertier-Marin-Sens detector, a process in a given host, say *p*, sent *heatbeats* for about 10 minutes, with the frequency of 1 message per second (*HP* = 1 second), and the process in another host tried to detect failure of *p* by running in parallel the three detectors: the Pure NN-AFD, the combined NN-AFD, and the Bertier-Marin-Sens detector.

The data collected in each round include the arrival time for the *heartbeat* messages, according to local clock, the estimation time for each detector, the time that a detector starts suspecting *p*, and the time that a detector corrects a mistake (i.e., stops suspecting *p* as a message from *p* arrives). We then calculated for each round, the number of false detections, the average detection time, and the average mistake duration. The table below shows the results. The mistake duration, the detection time, and the number of false detections are the average for the five rounds.

	Pure NN-AFD	**Bertier-Marin-Sens**	**Combined NN-AFD**
Number of false detections average	14,7	17	16,33
Mistake duration average (*ms*)	26,05	31,85	31,87
Detection Time average (*ms*)	1010,03	1032,17	1010,45

When we consider load variations, our experiments clearly show that the pure NN-AFD performed better than the detector of Bertier-Marin-Sens for all metrics. Indeed, when compared with Bertier-Marin-Sens detector, this result contradicts the common intuition that shorter detection times, which favour faster recovery procedures for fault-tolerant computing, leads to a less accurate detector. Still considering the experiment with load variation, the combined NN-AFD outperformed Bertier-Marin-Sens detector, being only slightly worse in the mistake duration average (in fact, practically the same average). As expected, NN-AFD outperformed the combined NN-AFD. Whereas these results validate our techniques, they also suggest that our method to switch between the detectors (in the combined version), can be improved to take advantage of the NN-AFD performance during load variations.

4. Concluding Remarks

We explored the use of artificial neural networks in order to improve the quality of service of failure detectors. In our work, the training patterns used to feed the neural network were obtained by using SNMP agents. The output of our neural network is an estimation for the arrival time for the failure detector to receive the next *heartbeat* from a remote process. Our detectors were fully implemented and tested over a set of LINUX networked workstations. In order to analyse the efficiency of our approach, we have run a series of experiments where network and system loads were varied randomly, and we measured several *QoS* parameters according to the metrics introduced in [3], comparing our detector against well-known implementations. The experiments show that our adaptive solution outperformed existing approaches. In the next step of our work, we are going to incorporate the failure detector implemented into a group membership service and a distributed system management tool under development at LaSiD/UFBA.

5. References

[1] Chandra, T. and Toueg, S., Unreliable Failure Detectors for Reliable Distributed Systems Journal of the ACM, 43(2):225-267, March 1996.

[2] Chandra, T., Hadzilacos, V. and Toueg S., The Weakest Failure Detector for Solving Consensus. Journal of the ACM, 43(4):685 – 722, July 1996.

[3] Chen, W., Toueg, S., Aguilera, M. On the quality of service of failure detectors. Proc. of the First Int. Conf. on Dependable Systems and Networks (DSN 2000).

[4] Macêdo R. Failure detection in asynchronous distributed systems. Proceedings of II Workshop on Tests and Fault-Tolerance, Curitiba, Brazil. July/2000. PP. 76-81.

[5] Nunes, R. C., Jansch-Pôrto, I. A lightweight interface to predict communication delays using time series. Proceedings of Latin American Symposium on Dependable Computing (LDAC 2003). LNCS 2847, pages 254-263.

[6] Jacobson, V. Congestion avoidance and control. Proc. of the ACM Symp. Comm., Architectures and Protocol, Palo Alto, ACM Press(1988) 314–329.

[7] Bertier, M., Marin, O., Sens, P. Implementation and performance evaluation of an adaptable failure detector. Proc. of Int. Conf. on Dep. Systems and Networks, 2002.

[8] Case, J., Fedor, M., Schoffstall, M., Davin, J. Connected: an Internet encyclopaedia: A simple network management. Protocol.
http://deese.univ lemans.fr:8003/connected/RFC/1157/index.html.

[9] McCloghrie, K., Rose, M. Connected: an internet encyclopaedia: Management information base for network management of TCP/IP-based internets: MIB-II. http://deese.univ-lemans.fr:8003/connected/RFC/1213/index.html.

[10] Devianov, B., Toueg, S. Failure detector service for dependable computing. Proceedings of the First Int. Conf. on Dependable Systems and Networks, pp. 2000

[11] Sotoma, I., Madeira, E. Adaptation – algorithms to adaptive fault monitoring and their implementation on CORBA. Proc. of the 3rd Int. Symp. on Dist. Objects and Appl., 2001.

[12] Macêdo, R, and Lima, F. Dynamically Adapting Failure Detectors to Loads Fluctuating Using SNMP and Artificial Neural Nets. Distributed Systems Laboratory (LaSiD), Federal University of Bahia (to be published).

[13] Haykin, S. Neural networks: a comprehensive foundation. 1st ed. New York, Macmillan, 1994.