2009 International Conference on Computational Science and Engineering

A Survey on Fault-tolerance in Distributed Network Systems

Naixue Xiong[1,2]
[1]*College of Computer Science, Wuhan Univ. of Science and Engineering, China*
Email: nxiong@cs.gsu.edu

Yan Yang
Center of Asian and Pacific Studies, Seikei Univserity, Tokyo, Japan
Email: yanyang818@gmail.com

Ming Cao
Surface Ship Development Dept. China Ship Development and Design Center,430064, China
Email:caomingwhu@yahoo.com.cn

Jing He[2]
[2]*Dept. of Computer Science, Georgia State University, Atlanta, GA, 30303, USA*
Email jhe9@student.gsu.edu

Lei Shu
Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland
Email:lei.shu@deri.org

Abstract

In this paper, we give a survey on fault tolerant issue in distributed systems. More specially speaking, we talk about one important and basic component called failure detection, which is to detect the failure of the process quickly and accurately. Thus, a good failure detection method will avoid the further system lost due to process crash. This survey provides the related research results and also explored the future directions about failure detection, and it is a good reference for researcher on this topic.

Keywords: *Failure detector, Fault-tolerance, Network Systems, Quality-of-service*

1. Introduction

Fault-tolerance is particularly prominent to distributed network systems in general, especially important in large-scale environment. Normally, distributed system users want the system to remain operational in spite of technical failures, even if some of the participants of these network systems have crashed. With a large number of participants and long running time, the probability that the hosts crash during the execution is inevitable, regardless of the physical reliability of each individual host. Thus, an effective system must be designed and executed in such a way that the system can tolerate seamlessly a reasonable number of host failures, and the occurrence of a reasonable number of host failures is acceptable.

Failure detection and process monitoring are the basic components of most techniques for fault-tolerance (tolerating failures) in distributed network systems, such as ISIS, Ensemble, Relacs, Transis, and Air Traffic Control Systems. Now how to design failure detectors over local networks is a rather well-known issue, but it is still far from being a solved problem with large-scale systems. Because a large-scale distributed system has lots of difficulties, which need to be addressed if we simulate them as a wired network environment: such as the potentially very large number of monitored processes, the higher probability of message loss, the ever-changing topology of the system, and the high unpredictability of message delays. All the above prominent factors fail to be addressed by the traditional solutions. To effective communication in large-scale distributed network systems and because of its importance [15-20], it is highly desirable for failure detectors to be executed as a common generic service shared among distributed applications (similar to IP address lookup) (e.g., [15, 18-19]) rather than as redundant ad hoc implementations (e.g., [21]). If such generic service can be achieved, it is very easy to apply failure detectors in any kinds of applications to ensure the requirement of fault tolerance. In spite of many ground-breaking advances made on failure detection, such a service still remains at a distant horizon [22].

The design of dependable Failure Detectors (FDs) is a hard task, mainly because of the indefinable statistic behavior of communication delays. Furthermore, asynchronous (i.e., no bound on the process execution speed or message-passing delay) distributed network systems make it impossible to determine precisely whether a remote process has failed or has just been very slow [23]. Failure detectors can be seen as one oracle per process. An oracle provides a list of processes that it currently suspects to have crashed. The unreliable FD [23] can make mistakes by erroneously suspecting correct processes or trusting crashed processes. Many fault-tolerant algorithms have been proposed [23-26] based on unreliable FDs. It is utmost important to ensure acceptable quality-of-service (QoS) of FD to properly tune its parameters for the most desirable QoS to be provided, because the QoS of FD greatly influences the QoS that upper layers may provide.

978-0-7695-3823-5/09 $26.00 © 2009 IEEE
DOI 10.1109/CSE.2009.497

However, there are few papers about comparing and implementing of these detectors [25].

2. Performance Metrics of Failure Detector

To be useful, a failure detector has to be reasonably fast and accurate. However, there are two aspects to consider the performance metrics. As we know, the paper [23] is the first time to discuss the performance metrics of failure detector qualitatively, i.e., from completeness and accuracy. Roughly speaking, completeness requires that a failure detector eventually suspects every process that actually crashes, while accuracy restricts the mistakes that a failure detector can make. The authors defined two completeness and four accuracy properties, which gives rise to eight classes of failure detectors, and consider the problem of solving Consensus using failure detectors from each class.

Of special interest is the weakest class of failure detectors considered in [23]. Informally, a failure detector is in W if it satisfies the following two properties:

Completeness. There is a time after which every process that crashes is permanently suspected by some correct process.

Accuracy. There is a time after which some correct process is never suspected by any correct process.

A set of metrics have been proposed by Chen et al. in [35] to quantify the QoS of a FD: how fast it detects actual failures and how well it avoids false detections. Note that speed is with respect to processes that crash, while accuracy is with respect to processes that do not crash. There are three main metrics: Detection Time, Mistake Rate, and Query Accuracy Probability.

Definition 1 (*Detection time: TD*). The detection time is the time that elapses from the crash of q until p begins to suspect q permanently.

Definition 2 (*Mistake Rate: MR*) This is a random variable that represents the number of mistakes that failure detector makes in a unit time, i.e., it represents how frequent failure detector makes mistakes.

Definition 3 (*Query Accuracy Probability: QAP*) This is a probability that, when queried at a random time, the FD at q indicates correctly that process p is up [27, 35].

Except these three metrics, there are *Mistake recurrence time (TMR), Mistake duration (TM) and Good period duration*. The output of the failure detector at *q* at time t is either *S* or *T*, which means that q suspects or trusts *p* at time *t*, respectively. A transition occurs when the output of the failure detector at q changes: An *S*-transition occurs when the output at *q* changes from *T* to *S*; a *T*-transition occurs when the output at *q* changes from *S* to *T*. We assume that there are only a finite number of transitions during any finite time interval.

Mistake recurrence time (TMR): this measures the time between two consecutive mistakes. More precisely, TMR is a random variable representing the time that elapses from an S-transition to the next one.

Mistake duration (TM): this measures the time it takes the failure detector to correct a mistake. More precisely, TM is a random variable representing the time that elapses from an S-transition to the next T-transition.

Good period duration (TG): this measures the length of a good period. More precisely, TG is a random variable representing the time that elapses from a T-transition to the next S-transition.

With these performance metrics, we would like to introduce the recent related work and their characteristics in the following section.

3. Related Failure Detectors and System Model

The long-term and final target of failure detectors is to define and implement a generic failure detection service for large scale network systems, and to provide failure detection service as a generic network-service, like Domain Name Service (DNS), Network Information System (NIS), Network File System (NFS), Send mail, etc. The network service consists of two parts: failure detection and information propagation. The former monitors processes, nodes, etc., and also detects their failures. This detection corresponds closely to traditional failure detectors.

The traditional failure detectors are based on a simple interaction system model. In this model, monitored processes can only send heartbeat messages and failure detections use constant timeout to either trust or suspect the monitored processes. Several recent adaptive failure detectors in [27, 36] could adjust a proper timeout based on both application requirements and network conditions. One of the major difficulties in building such a service is that applications with completely different requirements and running simultaneously have to be effectively adjusted by the service to meet their requirements. Furthermore, many distributed applications can greatly benefit from providing different levels of failure detection to trigger different reactions (see [31, 33]). For example, an application can take a precautionary action when confidence in a suspicion reaches a given level (a certain lower level); while take more drastic action once the confidence raises up (much higher level) [37].

Except for the above related works, there were some other failure detection mechanisms. For example, Falai and Bondavalli [38] introduced a lot of experiments performed on Wide Area Network to assess and fairly compare QoS, which is provided by a large family of FDs. They described choices for estimators and safety margins used to build several FDs. Compared with [38], Xiong et al. [35] considered comparing all kinds of adaptive FD schemes in different experimental environments.

Nunes et al. [39] analyzed and evaluated the QoS of an FD based on timeout for the different combinations of safety margins and communication delay predictors. Based on the results, in order to revise the QoS, the authors

presented the advice that one must consider the relation between the pair predictor/margin, instead of each one separately. However, we think it is hard to find such proper pairs.

Fabio et al. [40] adapt FDs to load fluctuations of communication network by artificial neural networks and Simple Network Management Protocol (SNMP). The training patterns, which were used to feed the neural network, were obtained by using SNMP agents over Management Information Base variables. The output of such neural network is an estimated value of the arrival time for the FD to receive the next heartbeat message from a remote process. This scheme improves the QoS of the FD, while the training of neural network is more complex on achieving the same goal as in this paper.

Fetzer et al. presented an adaptive failure detection protocol in [30]. This scheme enjoys the nice property of relying as much as possible on application messages to perform this monitoring. Differently from previous process of crash detection protocols, this protocol uses control messages only when no application messages are sent by the monitoring process to the observed process. These measurement results demonstrate that the number of wrong suspicions can be reduced by requiring each process to keep track of the maximum round trip delay between executions.

Hayashibara et al. [34] presented that a generic failure detection service outputted a value based on a continuous normalized scale. Simple speaking, this value showed the degree of confidence in the judgment if the corresponding process had crashed. It was then left behind to each application process to set a suspicion threshold in terms of its own quality-of service requirements. Furthermore, even within the scope of a single distributed application, it was often desirable to trigger different reactions based on different degrees of suspicion. The key advantage of this approach [34] was that it decoupled the failure detection service from running applications. This scheme allowed it to scale well with respect to the number of simultaneously running applications and/or triggered actions within each application.

For improving the QoS of FD, lots of adaptive FDs have been presented [17, 30, 39-40], such as Chen FD [27], Bertier FD [28], and the φ FD [34]. In [27], Chen et al. presented several implementations depending on clock synchronization and a probabilistic behavior of the system. The implementations used arrival times sampled in the recent past to compute an estimation of the arrival time of the next heartbeat. The timeout was set based on this estimation and a constant safety margin, and it was recomputed for every interval. This scheme provided a good estimation for the next arrival time. Furthermore, this paper assumed that the communication history was driven by uncorrelated samples with an ergodic stationary action, and message delays followed some probabilistic distribution. However, this scheme used a constant safety margin because the authors estimated that the model presented a probabilistic behavior [28]. Therefore, Bertier FD [28] presented an optimization of safety margin for Chen FD. It used a different estimation function, which combined Chen's estimation and Jacobson's estimation of the round-trip time (RTT). Bertier FD performed as a very aggressive FD [34], because this scheme was primarily designed to be used over wired local area networks (LANs). It means that in the environments messages were seldom lost. Hayashibara et al. [34] proposed a method using a probabilistic analysis of network traffic, it was similar as in Chen FD, and it assumed that the inter-arrival time followed a normal distribution. Furthermore, φ FD computed a value φ with a scale that changed dynamically to match recent network conditions. This FD, differently from the other FDs, outputted a suspicion level on a continuous scale, which was instead of binary nature (suspect or trust). These above three FDs dynamically predicted new timeout values in terms of observed communication delays to revise the performance of the protocols. The self-tuned FDs, presented in [41] and [36], used the statistics of the previously obtained communication delays to continuously adjust its timeout. I.e., they assumed a weak past dependence on communication history.

For the former failure detection schemes, all of them can not actively adapt their parameters by themselves to satisfy the requirement of users. To the question, Xiong et al. [35] presented a self-tuning failure detection based on [27]. In this paper, lots of experimental results demonstrate that the scheme is effective. It is sure that this idea also can apply into other failure detection to achieve self-tuning requirement. Firstly, this paper presented an optimization to improve the adaptation of [27], which significantly enhanced QoS, especially in the aggressive range and when the network was unstable. Secondly, they addressed the problem of most adaptive schemes, namely their requirement for a large window of samples. They studied a scheme that was designed to use a fixed and very limited amount of memory for each monitored–monitoring link. The experimental results over several kinds of networks (Cluster, WiFi, wired LAN, WAN) demonstrated the properties of the existing adaptive FDs and that the optimization is acceptable and reasonable. Furthermore, the extensive experimental results demonstrated what was the effect of memory size on the overall QoS of every adaptive FD.

Xiong et al. [42] observed from lots of experimental statistical results, and found it was not a good assumption that the φ failure detection [34] uses the normal distribution to estimate the arrival time of the coming heartbeat, especially in large scale distributed networks or unstable networks. Thus, here this paper developed an optimization over φ failure detection based on exponential distribution, called exponential distribution failure detection. This significantly improved the QoS, especially in the design of

real systems. Lots of experiments have been carried out based on several kinds of networks (Cluster, WiFi, Wired local area network, and Wide area network). The experimental results had demonstrated the properties of the existing adaptive FDs, and demonstrated that the presented exponential distribution FD outperforms the existing FDs in the aggressive range.

Hayashibara in [22] proposed a basic concept of κ FD, which had the formal properties of accrual FD. This scheme allowed for gradual settings between a conservative behavior and an aggressive one. Here, they developed this idea and especially further discussed service performance of κ failure detector in a variety of network environments.

Failure detections have been provided in many papers as an independent service (e.g., [15, 18-19, 23]). While, there are several important issues, which should be addressed before an effective generic service can be really executed.

(1) A failure detection service must adapt to dynamic network conditions and application requirements. Several solutions were proposed recently to address this issue specifically [17, 27, 28, 30].

(2) A failure detection service must adapt to diverse application requirements. A few schemes (e.g., [27]) have been made to adapt the parameters of a failure detector service to match the requirements, while they are designed to support a single class of requirements. Cosquer et al. [29] identified the problem. Their scheme is wonderful, while it remains inflexible because they do not express the Boolean nature of failure detection.

Hayashibara et al. [34] have pointed this out recently. They proposed a φ failure detector to deal with this point. However, the QoS of failure detector is not enough for an effective generic service.

(3) A failure detection service must propose a good QoS for users. However, so far as I know, many schemes try to express this point. While none of the solutions resolved it perfectly.

As mentioned above, lots of problems remain in implementing a generic failure detection service. Also, to the best of our knowledge, there is no work about self-tuning the parameters of failure detector to satisfy requirement of users. As we know, it is still an open problem in fault tolerant research area. Therefore, it is very necessary to address this question, i.e., it can adjust the parameters of failure detector to satisfy requirement of users by itself.

4. Scalability and Reliability for Fault-tolerant network system

For the information delivery applications in network environment, scalability and reliability are two of the most important issues.

Network reliability has two aspects: one is the availability of end to end functionality for customers, the other is the ability to experience failures or systematic attacks, without impacting customers or operations. Network reliability is not cheap or free, and is not only about component reliability, reliable piece or part, fault tolerance hardware or fast recovery.

Network scalability is about the ability to handle growing amount of work, and the capability of a system to increase total throughput under an increased load when resources are added.

To tackle the above two issues at the same time, many schemes are proposed in wired/ wireless network environment.

For wired network environment,

(1) Tree-based hierarchies of subgroups: In different schemes, the special nodes in the subgroups can be group members or not and they can be either individual hosts or routers. Different names of the special nodes are used in different schemes, e.g. the proxies in [1] and the Reliable Multicast proxies (RMXs) in [2].

(2) Client-Server scheme [3]: Group membership services are provided by dedicated membership servers. Different levels of membership servers function as multicast routers to route messages to particular groups according to the membership information maintained in these membership servers.

For wireless network environment:

(1) Mobile IP Bidirectional Tunneling (MIP-BT): A bidirectional tunnel is built between the mobile host (MH) and its Home Agent (HA).

(2) Mobile IP Remote Subscription (MIP-RS): each MH always r-subscribes to its desired multicast group when it enters a foreign network.

(3) Mobile Multicast (MoM) scheme [4]: It uses the Designated Multicast Service Provider (DMSP) to avoid duplicated messages being tunneled to the same Foreign Agent (FA).

(4) The Multicast Agent (MA) scheme [5]: An MA is a multicast router that provides multicast services to mobile group members in multiple foreign networks.

(5) The MobiCast scheme [6]: It adopts a hierarchical mobility management approach to isolate the mobility of the MHs from the main multicast delivery tree. Each foreign domain has a Domain Foreign Agent (DFA).

(6) The multicast by Multicast Agent (MMA) [7]: In this scheme, the Multicast Agent (MA) and the Multicast Forwarder (MF) are introduced.

(7) The Host-View Membership Protocol (HVMP) [8]: a two-tier framework for reliably delivering multicast messages to MHs is presented.

(8) The Reliable Multicast (RelM) scheme [9]: a three-tier framework is proposed to deal with the problems in the HVMP scheme.

(9) The Reliable Multicast Protocol (RMP) [10]: the three-tier framework of MHs, Mobile Support Stations (MSSs) and Coordinators is proposed.

(10) RingNet hierarchy of proxies [12-14]: The RingNet hierarchy is more general than the ring-based hierarchy in [11] in the sense that each proxy within this hierarchy may have multiple children nodes, while each proxy within the ring-based hierarchy has at most one child node. A Membership-Propagation algorithm proposed to propagate membership information from non-leader to leader and from leader to parent. Moreover a Topology-Maintenance algorithm proposed to maintain the hierarchy due to Member-Join/Leave/Failure/Handoff and Proxy-Failure events.

One commonality of the above mobile multicast protocols is that they employ the tree-based hierarchy for group communication, and they consider only message loss problem due to un-reliable communication links. We also need to consider node failures within the tree-based hierarchy to improve the reliability of the network.

5. Conclusion and future work

In this paper, we first explore the relative failure detection, which is an important issue for supporting dependability in distributed systems, and often is an important performance bottleneck in the event of node failure. In this field, we analyze the existing failure detections, including the four different schemes (Bertier FD, Exponential distribution FD, Kappa FD, Self-tuning FD) to ensure acceptable QoS in unpredictable and dynamic network environments.

In future work, the QoS scalability is researched, which is as interference from heavier network traffic (e.g., a scenario where most of the nodes in the networked system have active FDs), to see whether that will affect detection time, detection accuracy, etc. Furthermore, we would like to research their properties and relation in software engineering applications, and then to find or propose a reasonable FD in fault-tolerant distributed system. To apply the proposed FD into an actual fault-tolerant distributed system, we should design a self-tuning failure detector (SFD) in actual fault-tolerant distributed system. For all the existing FDs so far, the common point is that they all can detect the directly connected processes. However, for some processes that are not directly connected, i.e., they only can communicate each other by some middle processes, all the existing FDs are not applicable. Therefore, an open question arises: how to design an indirect failure detector to make sure any two processes, even they are not connected directly, can detect each other effectively. Furthermore, we could explore the following three aspects.

(1) Design new schemes based on different architectures of failure detection;

(2) Build a pragmatic platform on failure detection;

(3) Carry out the applications of FDs in Ad hoc, Mobile network, or other environment;

9. Acknowledgements

This research has been supported by the US National Science Foundation CAREER Award under Grant No. CCF-0545667. Mr. Lei Shu's work in this paper was supported by the Lion project supported by Science Foundation Ireland under grant No. SFI/08/CE/I1380 (Lion-2), and by the European project CONET (Cooperating Objects NETwork of excellence) under grant No. 224053.

12. References

[1] A.P. Markopoulou and F.A. Tobagi, “Hierarchical Reliable Multicast: Performance Analysis and Placement of Proxies,” Proc. Int’l Conf. Parallel Processing (ICPP ‘00), pp. 271-278, Aug. 2000.

[2] Y. Chawathe, S. McCanne, and E.A. Brewer, “RMX: Reliable Multicast for Heterogeneous Networks,” Proc. IEEE INFOCOM ’00, vol. 2, pp. 795-804, Mar. 2000

[3] T. Anker, G. V. Chockler, D. Dolev, and I. Keidar, “Scalable Group Membership Services for Novel Applications,” Proc. DIMACS 1998, vol. 45, pp. 23-42, 1998

[4] T.G. Harrison, C.L. Williamson, W.L. Mackrell, and R.B. Bunt, “Mobile Multicast (MoM) Protocol: Multicast Support for Mobile Hosts,” Proc. ACM MobiCom 1997, pp. 151-160. Sept. 1997

[5] Y. Wang and W. Chen, “Supporting IP Multicast for Mobile Hosts,” ACM Mobile Networks and Applications, vol. 6, no. 1, pp. 57-66, 2001

[6] C.L. Tan and S. Pink, “MobiCast: A Multicast Scheme for wireless Networks,” ACM Mobile Networks and Applications, vol. 5, no. 4, pp. 259-271, Dec. 2000

[7] H.S. Shin, Y.J. Suh, and D.H. Kwon, “Multicast Routing Protocol by Multicast Agent in Mobile Networks,” Proc. IEEE Int’l Conf. Parallel Processing (ICPP ‘00), pp. 271-278, Aug. 2000

[8] A. Acharya and B.R. Badrinath, “A Framework for Delivering Multicast Messages in Networks with Mobile Hosts,” ACM/Kluwer Mobile Networks and Applications, vol. 1, no. 2, pp. 199-219, Oct. 1996

[9] K. Brown and S. Singh, “RelM: Reliable Multicast for Mobile Networks,” Computer Comm., vol. 21, no. 16, pp. 1379-1400, Oct. 1998

[10] G. Anastasi, A. Bartoli, and F. Spadoni, “A Reliable Multicast Protocol for Distributed Mobile Systems: Design and Evaluation,” IEEE Trans. Parallel and Distributed Systems, vol. 12, no. 10, pp. 1009-1022, Oct. 2001

[11] G. Wang, J. Cao, and K.C.C. Chan, “RGB: A Scalable and Reliable Group Membership Protocol in Mobile Internet,” Proc. IEEE Int’l Conf. Parallel Processing (ICPP ’04), pp. 326-333, Aug. 2004

[12] G. Wang, J. Cao, and K.C.C. Chan, “A Reliable Totally-Ordered Group Multicast Protocol for Mobile Internet,” Proc. IEEE Int’l Conf. Parallel Processing (ICPP ‘04), pp. 108-115, Aug. 2004

[13] G. Wang, L. Liao, J. Cao, and K.C.C. Chan, “Key Management for Secure Multicast Using the RingNet Hierarchy,” Proc. Int’l Conf. Computational and Information Sciences (CIS ‘04), pp. 77-84, Dec. 2004

[14] J. Cao, G. Wang, and K. C.C. Chan, "A Fault Tolerant Group Communication Protocol in Large Scale and Highly Dynamic Mobile Next-Generation Networks", IEEE

Transactions on Computers, vol. 56, no. 1, pp. 80 – 94, Jan. 2007

[15] P. Felber, X. Defago, R. Guerraoui, and P. Oser. Failure detectors as first class objects. In Proc. IEEE Intl. Symp. On Distributed Objects and Applications (DOA1999), pages 132-141, Edinburgh, Scotland, Sep. 1999.

[16] J. Dunagan, N. J. A. Harvey, M. B. Jones, D. Kostic, M. Theimer, and A. Wolman. FUSE: Lightweight guaranteed distributed failure notification. In Proc. 6th Symp. on Operating Systems Design and Implementation (OSDI 2004), San Francisco, CA, USA, December 2004.

[17] I. Sotoma and E. R. M. Madeira. ADAPTATION-algorithms to adaptive fault monitoring and their implementation on CORBA. In Proc. 3rd Intl. Symp. on Distributed-Objects and Applications (DOA 2001), pages 219-228, Rome, Italy, September 2001. IEEE Computer Society Press.

[18] P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. von Laszewski. A fault detection service for wide area distributed computations. In Proc. 7th IEEE Symp. on High Performance Distributed Computing, pages 268-278, July 1998.

[19] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In N. Davies, K. Raymond, and J. Seitz, editors, Middleware 1998, pages 55-70, The Lake District, UK, 1998.

[20] M. Wiesmann, P. Urban, and X. Defago. An SNMP based failure detection service. In Proc. 25th IEEE Intl. Symp. on Reliable Distributed Systems (SRDS 2006), pages 365-374, Leeds, UK, October 2006.

[21] N. Sergent, X. Defago, and A. Schiper. Impact of a failure detection mechanism on the performance of consensus. In Proc. 8th IEEE Pacific Rim Symp. on Dependable Computing(PRDC-8), pages 137C145, Seoul, Korea, December 2001.

[22] N. Hayashibara. Accrual failure detectors. Doctoral thesis, Japan Advanced Institute of Science and Technology, June, 2004.

[23] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. Journal of the ACM, 43(2): 225-267, Mar. 1996.

[24] M. K. Aguilera, W. Chen, and S. Toueg. Using the heartbeat failure detector for quiescent reliable communication and consensus in partition-able networks. Theoretical Computer Science, 220(1): 3-30, Jun. 1999.

[25] M. Larrea, A. Fernandez, and S. Arevalo. Optimal implementation of the weakest failure detector for solving consensus. In Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing, pages 334-334, NY, Jul. 2000.

[26] R. Guerraoui, M. Larrea, and A. Schiper. Non-blocking atomic commitment with an unreliable failure detector. Symposium on Reliable Distributed Systems, pages 41–50, 1995.

[27] W. Chen, S. Toueg, and M. K. Aguilera. On the quality of service of failure detectors. IEEE Transaction on Computers, 51(5):561–580, May 2002.

[28] M. Bertier, O. Marin, P. Sens. Implementation and performance evaluation of an adaptable failure detector. In Proc. Intl. Conf. on Dependable Systems and Networks (DSN2002), pages 354-363, Washington DC, USA, Jun. 2002.

[29] F. J. N. Cosquer, L. E. T. Rodrigues, and P. Verissimo. Using tailored failure suspectors to support distributed cooperative applications. In Proc. 7th IASTED Intl. Conf. on Parallel and Distributed Computing and Systems (PDCS 2005), pages 352-356, Washington, DC, USA, October 1995.

[30] C. Fetzer, M. Raynal, and F. Tronel. An adaptive failure detection protocol. In Proc. IEEE the 8th Pacific Rim Symposium on Dependable Computing, pages 146- 153, Seoul, Korea, Dec. 2001.

[31] B. Charron-Bost, X. Defago, and A. Schiper. Broadcasting messages in fault-tolerant distributed systems: the benefit of handling input-triggered and output-triggered suspicions differently. In Proc. 21st IEEE Intl. Symp. on Reliable Distributed Systems (SRDS2002), pages 244–249, Osaka, Japan, Oct. 2002.

[32] X. Defago, A. Schiper, and N. Sergent. Semi-passive replication. In Proc. 17th IEEE Intl. Symp. Reliable Distributed Systems (SRDS1998), pages 43–50, West Lafayette, IN, USA, Oct. 1998.

[33] P. Urban, I. Shnayderman, and A. Schiper. Comparison of failure detectors and group membership: Performance study of two atomic broadcast algorithms. In Proc. IEEE Intl. Conf. on Dependable Systems and Networks (DSN2003), pages 645–654, San Francisco, CA, USA, June 2003.

[34] N. Hayashibara, X. Defago, R. Yared, and T. Katayama. The phi accrual failure detector. In Proc. 23rd IEEE Intl. Symp. on Reliable Distributed Systems (SRDS2004), pages 66-78, Florianpolis, Brazil, Oct. 2004.

[35] N. Xiong, Athanasios V. Vasilakos, Laurence T. Yang, and Lingyang Song, Pan Yi, Rajgopal Kannan, Y. Li, "Comparative Analysis of Quality of Service and Memory Usage for Adaptive Failure Detectors in Healthcare Systems," IEEE Journal on Selected Areas in Communications (JSAC), vol. 27, no. 4, pp. 495-509, May 2009.

[36] P. Felber. The CORBA object group service - a service approach to object groups in CORBA. PhD thesis, D′epartment D'informatique, Lausanne, EPFL, Swizerland, 1998.

[37] M. M¨uller. Performance evaluation of a failure detector using SNMP. Semester project report, Ecole Poly-technique Federale de Lausanne, Lausanne, Switzerland, Feb. 2004.

[38] L. Falai and A. Bondavalli. Experimental evaluation of the QoS of failure detectors on wide area network. In Proc. of the Int. Conf. on Dependable Systems and Networks (DSN'05), Yokohama, Japan, pages 624-633, Jun. 2005.

[39] R. C. Nunes, I. Jansch-Porto. QoS timeout-based self-tuned failure detectors: the effects of the communication delay predictor and the safety margin. In Proc. 2004 International Conference on Dependable Systems and Networks (DSN 2004), pages 753-761, Florence, Italy, June 2004.

[40] L. Fabio, R. Macedo. Adapting failure detectors to communication network load fluctuations using SNMP and artificial neural networks. In Proc. Second Latin-American Symposium on Dependable Computing (LADC 2005), pages 191-205, Salvador, Brazil, Oct. 2005.

[41] R. Macedo. Implementing failure detection through the use of a self-tuned time connectivity indicator. TR, RT008/98, Laboratrio de Sistemas Distribudos - LaSiD, Salvador-Brazil, Aug. 1998.

[42] N. Xiong. Design and Analysis of Quality of Service on Distributed Fault-tolerant Communication Networks. Doctoral thesis, Japan Advanced Institute of Science and Technology, March, 2008.