A Failure Detection approach of Data Center Networks[*]

Kai Shen[†]
School of Software
Shanghai Jiao Tong University
Wallamaloo, New Zealand
knshen@sjtu.edu.cn

Charles Palmer
Palmer Research Laboratories
8600 Datapoint Drive
San Antonio, Texas 78229
cpalmer@prl.com

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings. It is an *alternate* style which produces a *tighter-looking* paper and was designed in response to concerns expressed, by authors, over page-budgets. It complements the document *Author's (Alternate) Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file. A compiled PDF version is available on the web page to help you with the 'look and feel'.

CCS Concepts

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Keywords

ACM proceedings; LaTeX; text tagging

1. INTRODUCTION

Data center. The rest of this paper is organized as follows. Section 2 discusses related work,

[*](Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.
[†]Dr. Trovato insisted his name be first.

WOODSTOCK '97 El Paso, Texas USA
© 2016 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

2. RELATED WORK

In this section, we will present various relevant works including failure detectors, failure detection architecture in distributed systems and failure detection in data center networks.

2.1 Failure Detectors

A failure detector (FD) is widely recognized as an oracle to intelligently suspect failed processes [3]. The monitored process periodically send heartbeat messages to its detector to prove its liveness.

Chen et al. proposed a FD that provides QoS [4]. It estimates the expected arrival time (EAs) of the next heartbeat message according to a slide window storing n most recent arrived messages. EAs determine the deadline that the detector will wait for the next heartbeat before suspecting the monitored process. *Chen FD* can adjust current network condition and set the timeout threshold adaptively by referring to recent heartbeats. Bertier et al. proposed a similar FD [2], whereas it uses a dynamic way to compute error margin of *Chen FD*. Other FDs like ϕ *Accrual FD* [8] and *Satzger FD* [11] can output a continuous accrual failure value other than a binary value at any time which stands for the failure probability of the process.

2.2 Failure Detection Architecture

A failure detection architecture aims to provide the service of monitoring nodes in large scale distributed systems in a scalable way. Roughly speaking, there are two kinds of architectures. The first one is hierarchical architecture [5], all the nodes are partitioned into different groups, each group has a leader node. Within a group, the leader node is responsible for monitoring all the nodes. Leader nodes periodically send node status information of his group to other leader nodes. Hierarchical architecture can reduce the number of heartbeat messages effectively, but it exists *Single Point of Failure* problem, when the leader node crashes, it needs to select a new leader.

Another alternative solution is Gossip-Style architecture [12]. Each node maintains a list containing the *heartbeat counter* for each node in the system. Every T_{gossip} seconds, each node firstly update *heartbeat counter* of itself and then randomly select another node to send its list to. Upon receiving a message, the node will merge two lists and update each node's *heartbeat counter* with the bigger one. If the *heartbeat counter* does not update after T_{fail} seconds, the node will be marked as crashed. If the *heartbeat counter* does not update after $T_{cleanup}$ ($T_{cleanup} \geq T_{fail}$) seconds, it will be removed from the list. This approach has very low bandwidth occupation because it only produces n messages each epoch where n is the size of the system. However, with the increasing of crashing nodes, it needs more time to detect these failures.

2.3 Failure Detection in Networks

Different from the work mentioned above, failure detection in networks should consider network topology and localize failure positions at IP layer. Gill et al. present the first large-scale survey on failures in data center networks [6].

Also, there exists a large body of work on applying statistical approaches to failure localization in networks. Sherlock [1] localizes failures to limited network components by capturing dependencies between components. Shrink [10] leverages Bayesian Network model to diagnose root cause of IP network failures. Herodotou et al. [9] actively inject ping traffic into network and analyze the ping results overlaid on top of the network topology. It will generate a ranked list containing the suspected failed device and links along with their failure score that can best explain the observed ping data.

3. DATA CENTER NETWORK ARCHITECTURE

In this section, we will take a brief look at common data center network architecture.

A data center is a facility used to house computer systems and associated components, it usually contains hundreds of thousands devices and links. Modern data centers use hierarchical architecture. Figure 1 depicts a conventional data center network architecture [6, 7, 9]. At the bottom layer are racks of servers, typically there are 20 to 40 servers in each rack, and each server in the same rack connects to a Top of Rack (ToR) Switch via a 1 Gbps link. Each ToR then connects to a primary and a backup Aggregation Switch (AS) for redundancy. ASs are connected to Access Routers (ARs) forming a complete bipartite graph. Finally, ARs are connected to Core Routers (CRs).

To limit overheads and to isolate different services, servers are partitioned into virtual LANs (VLANs). All links in the data center use Ethernet as link layer protocol and physical connections are a mix of copper and fiber cables. In addition to the devices in Figure 1, there may exist other devices like load balancers (LBs) and firewalls.

Modern data center usually employs *Equal Cost MultiPath* (ECMP) routing within the data center for fault tolerance, which means that there are multiple paths from source to destination. Thus in this work, we will apply ECMP routing to the simulated network by enabling ECMP routing configurations in NS-3.

4. APPROACH OVERVIEW

In this section, we will introduce the whole design of our failure detection approach.

Different network device should have different quality of service (QoS). Gill et.al [6] have shown the diversity of different devices' failure characteristics. Unfortunately, none of the work in Section 2 treats these devices differently. We firstly divide network devices into two categories.

- **Imperetative Device.** This category includes servers, ToRs and links between ToRs and servers. Typically servers run various user applications and store user data. The crash of servers may cause user-perceived faults. The crash of a ToR will make servers under it unreachable. Also Gill et.al declare that the downtime of ToRs may be very long. Thus the failure detection for these devices must be fast enough besides the accuracy requirement.

- **Non-imperetative Device.** It includes other routers and switches. A single failure of a link or a router does not affect the functionality of a data center because of path redundancy. So administrators care more about how effective the detect service is (The ability to identify actually happening failures).

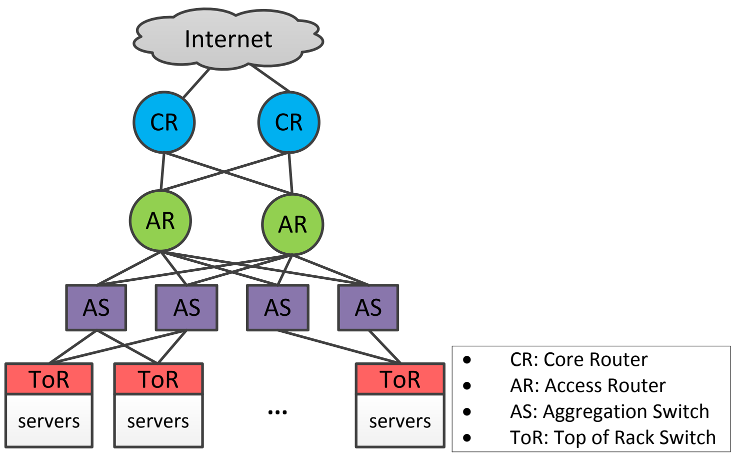

Figure 1: Conventional Data Center Network Topology

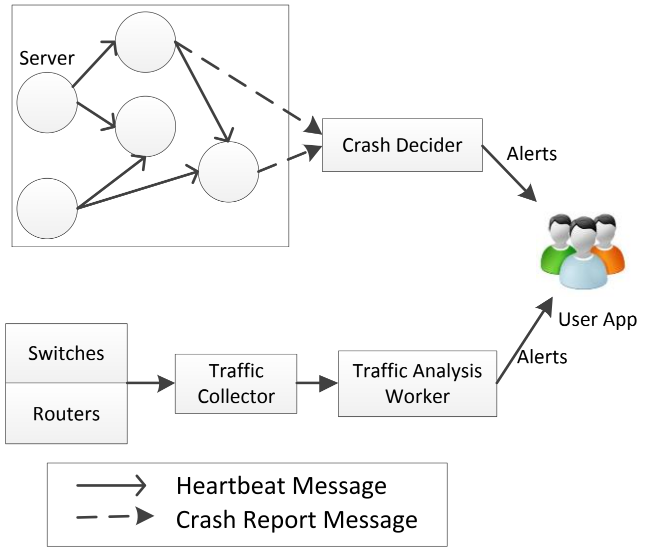

Figure 2: Failure Detection System Architecture

Figure 2 illustrates the entire architecture of our failure detection system. It mainly contains two modules. The servers failure detection system is autonomous and combines centralized structure with decentralized structure. A server can both play the role of a detector and a monitored object. Each server is monitored by multiple FD processes installed on different servers for redundancy. A server can also monitor a group of servers. Once the FD outputs the crash of a server, it will send a crash report message to **Crash Decider**. **Crash Decider** will determine the final status of this server.

For the detection of non-imperetative devices, **Traffic Collector** periodically collects traffic data of routers and switches and send telemetry data to **Traffic Analysis Worker**. **Traffic Analysis Worker** finally generate failure alerts by performing analysis on these data.

5. FAILURE DETECTION APPROACH

In this section, we will discuss the detailed design of different modules of the failure detection system respectively.

5.1 Failure Detection for servers

Failure Detection for servers is responsible for reporting the crash of servers effectively and efficiently. Our detection model is based on the assumption that once the server crashes, it never recovers by itself. The QoS of failure detection for servers are as follows:

- **Efficiency.** It measures how fast the system can report the crash of the server after it crashes. The system is more efficient if it takes less time to detect a failure.
- **Completeness.** It measures whether the system can detect all the crashed servers. The system is more complete if it rules out fewer true crashed ones.
- **Soundness.** In some scenarios, the behavior of the network may vary. For example, message loss and message delay may happen sometimes. The detection system should be sound enough to distinguish between such condition and a real crash.

5.1.1 k-detector

To meet the requirements mentioned above, this paper proposes a detector model named k-detector model.

Rationale.

5.1.2 Monitoring Rule Inference

5.2 Failure Detection for ToRs

5.3 Failure Detection for Links

6. EVALUATION

7. CONCLUSIONS

This paper proposes a failure detector model for data center networks.

8. ACKNOWLEDGMENTS

This work is partially supported by the National High-Technology Research and Development Program of China (Grant No.2014AA01A301).

9. ADDITIONAL AUTHORS

10. REFERENCES

[1] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 13–24. ACM, 2007.

[2] M. Bertier, O. Marin, and P. Sens. Implementation and performance evaluation of an adaptable failure detector. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pages 354–363. IEEE, 2002.

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

[4] W. Chen, S. Toueg, and M. K. Aguilera. On the quality of service of failure detectors. *IEEE Transactions on computers*, 51(5):561–580, 2002.

[5] P. Felber, X. Défago, R. Guerraoui, and P. Oser. Failure detectors as first class objects. In *Distributed Objects and Applications, 1999. Proceedings of the International Symposium on*, pages 132–141. IEEE, 1999.

[6] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 350–361. ACM, 2011.

[7] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: a scalable and flexible data center network. In *ACM SIGCOMM computer communication review*, volume 39, pages 51–62. ACM, 2009.

[8] N. Hayashibara, X. Defago, R. Yared, and T. Katayama. The φ accrual failure detector. In *Reliable Distributed Systems, 2004. Proceedings of the 23rd IEEE International Symposium on*, pages 66–78. IEEE, 2004.

[9] H. Herodotou, B. Ding, S. Balakrishnan, G. Outhred, and P. Fitter. Scalable near real-time failure localization of data center networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1689–1698. ACM, 2014.

[10] S. Kandula, D. Katabi, and J.-P. Vasseur. Shrink: A tool for failure diagnosis in ip networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 173–178. ACM, 2005.

[11] B. Satzger, A. Pietzowski, W. Trumler, and T. Ungerer. A new adaptive accrual failure detector for dependable distributed systems. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 551–555. ACM, 2007.

[12] R. Van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Middlewareąŕ98*, pages 55–70. Springer, 1998.