MATEC Web of Conferences **42**, 07002 (2016)
DOI: 10.1051/matecconf/20164207002
© Owned by the authors, published by EDP Sciences, 2016

Reliable Channels for Systems in the Presence of Byzantine Faults

Mehmet Karaata[1], Ali Hamdan[2]

[1]*Department of Computer Engineering, Kuwait University, P.O. Box 5969, Safat 13060 Kuwait*
[2]*Department of Computer Engineering, Kuwait University, P.O. Box 5969, Safat 13060 Kuwait*

Abstract. Consider a distributed system that delivers each message from a process to its destination if the message transmission does not experience any faults and only delivers those sent by a non-faulty system process. Such a system is referred to as a reliable message passing system. A reliable message passing system requires a reliable channel, a communication channel between a pair of processes that always detects a fault in message transmission and each detected fault is an actual fault, to be implemented. In this paper, we first identify the necessary conditions to detect some restricted form of Byzantine Faults in a message passing system where n disjoint paths exist between each pair of endpoints. We consider Byzantine Faults (BF) whose e_ect is limited to the modification of a message metadata, omission faults, and message replay. We then present a protocol implementing a reliable channel in message passing systems in the presence of n — 1 Byzantine Faults using n disjoint paths between each pair of communication endpoints where the paths with faults are not known apriori. The proposed protocol detects Byzantine Faults, where each detected fault, an actual fault, authenticates message origins, identifies faulty paths and classifies faults in the presence of multiple messages sent by various system processes.

1 Introduction

Communication reliability and security are essential properties in computer networks. Two important issues in communication reliability and security are secrecy and authenticity; this work is concerned with the latter. A communication that provides secrecy ensures that the message contents cannot be discovered by intermediate processes and channels. On the other hand, an *authenticated network* transmits each message sent and only those sent by the source process to the destination process intact. A channel with communication authenticity, also referred to as an *authenticated channel*, is implemented by recovering from lost and altered messages, and detecting spurious messages, i.e., messages generated in the system by sources other than the expected/intended source process. Whereas, *a reliable channel* transmits only those messages sent by the source process to the destination process intact.

A reliable channel ensures the following two properties. *Completeness:* whenever a node observes the effects of a faulty message, the system eventually generates evidence against the faulty message. *Accuracy:* it never generates valid evidence against a correctly delivered message.

Distributed systems are subject to variety of faults including *Byzantine Faults* (BF) [1]. A Byzantine Fault is the most general form of faults where a system component may exhibit an arbitrary behaviour such as corruption of its state or program, sending arbitrary messages, or modifying system messages [2]. Therefore, many security attacks including censorship, misrouting, data corruption, software defects, and virus attacks can be considered as Byzantine Faults.

Byzantine Fault Detection, is a fundamental problem in achieving system availability, safety, and security. Byzantine Fault detection refers to the ability of a system to monitor and identify faulty system behaviour, rather than masking the effect of the fault. Byzantine fault detection is important since upon detection of the fault, the systemcan take appropriate actions.

Known approaches to deal with Byzantine Faults include service replication [3], cryptographic techniques, message authentication codes [3], network coding and error detecting/recovering code techniques [4], and signed digests [5]. For instance, [4] shows that Byzantine modification detection capability can be added to a multicast scheme based on random linear block network coding, with modest additional computational and communication overhead, by incorporating a simple polynomial hash/check value in each packet. With this approach, a sink node can detect Byzantine modifications with high probability, provided that these modifications have not been designed with knowledge of the random coding combinations present in all other packets obtained at the sink: the only essential condition is the adversary's incomplete knowledge of the random network code seen by the sink.

Existing schemes that detect or tolerate Byzantine Faults perturbing message content and variables deal with end-to-end communication [6, 7, 8] but fall short of tolerating/detecting faults in the form of spurious, and replay messages, and perturbation of message parameters during message routing. When message parameters are perturbed and spurious and replay messages are sent, additional fault types are introduced including routing the message to an unintended destination, mixing up of shares of *different messages*, *replacement messages* or *false positives*, a non-faulty message detected as a faulty message. *Replacement messages* are messages whose parameters are altered in such a way that it is regarded as another message. Cryptographic schemes, schemes based on network coding, message digests and hashing techniques available in the literature do not deal with these types of faults. These techniques also make strong assumptions such as perfect cryptography and cause delays due to the ciphering/coding at the sending end and deciphering/decoding at the receiving end. Furthermore, most authenticated channel implementations rely on computational bounds on the adversaries and an authentication step to exchange a key.

In this paper, we consider a system where processes communicate via message-passing over disjoint paths in the presence of Byzantine Faults. We first identify the necessary conditions to detect the presence of Byzantine Faults in the system using messages exchanged via disjoint paths between communicating endpoints. We then present a protocol to implement *reliable channels* in message passing systems in the presence of *n-1* Byzantine Faults using *n* disjoint paths between each pair of communication endpoints where the paths with faults are not known. The proposed algorithm deals with a large variety of Byzantine Faults including *spurious messages*, message omission and replay, and modification/corruption of message meta-data (parameters). The protocol allows multiple messages to be sent simultaneously in the system where multiple faults in the meta-data of the messages may occur introducing additional types of Byzantine Faults such as replacement messages that are not possible otherwise. The algorithm also performs sender authentication where the destination process detects changes in the sender id and messages sent by an adversary (imposter) process claiming the id of another process. It also identifies faulty paths and the fault types allowing various recovery methods to be adapted. The algorithm does not use cryptographic or error correcting/recovering codes. Therefore, there is no message latency due to network coding or cryptographic encodings, and there is no need to agree on the coding/decoding scheme be- forehand leading to timely discovery of faults affecting message transmission.

The proposed algorithm can be used for reliable message passing and fault detection in computer networks containing multiple disjoint paths between communication endpoints, in particular, in P2P networks whose overlay network topologies often contain many disjoint paths between endpoints. The protocol has the overhead of sending *n* copies of the same message. However, the overhead can be reduced if the number of faults reduces. Using the proposed protocol, a perfectly reliable message passing can be implemented where the message is delivered if there is no fault detected, and discarded otherwise.

In addition, the proposed protocol can be combined with network coding techniques when the protocol is modified to send *n* shares of a message using network coding instead of sending *n* copies of the same message. In this case, network coding can be used to detect/tolerate faults in message content while the proposed algorithm detects various other forms of Byzantine Faults during the routing of messages accurately.

The paper is organized as follows. In Section 2, we present the background of the proposed work and the related work available in the literature. Section 3 defines the necessary conditions required for the detection of Byzantine Faults. We conclude the paper in Section 4 with some final remarks.

2 Background and Related Work

A survey of information theoretic research in the area of secure network communication in the presence of Byzantine adversaries is given in [9]. Two important issues in secure network communication are secrecy and authenticity. In the network coding context, the problem of ensuring secrecy in the presence of a wire tap adversary has been considered in [10] and [11]. The problem of correcting adversarial errors to build authenticated channels has been studied in [12] and [13].

The concept of failure detection was introduced by Chandra and Toueg [14]. These were defined for crash failures, but Malkhi and Reiter [15] later extended them to special classes of Byzantine failures. In a more general manner, Doudou et al. [16] have introduced muteness failure detectors dealing with nodes that prematurely stop sending messages. Ho et. al [4] present a Byzantine modification detection based on random network coding techniques. Kihlstrom et al. [17] have introduced several classes of failure detectors that expose detectable Byzantine failures. However, they consider classes of algorithms in which all messages are broadcast, and in which processes know when to expect messages from other processes. State machine replication [18] is a classical technique for masking a limited number of Byzantine Faults. BFT techniques, e.g. [3], are based on this idea. Although these techniques perfectly protect the system from Byzantine failures, they are usually not intended to detect the faulty nodes, and they are inherently expensive and not scalable. The BAR model [19] extends the BFT approach to tolerate selfish behaviour of rational nodes while providing a mechanism for detecting certain application specific misbehaviour. Alvisi et al. [20] introduced a technique that monitors quorum systems and raises an alarm if the failure assumptions are about to be violated. This technique is probabilistic and cannot identify which nodes are faulty. Intrusion detection systems [21] can detect certain types of protocol violations; however the heuristics used in IDS tend to produce either false positives, false negatives, or both. Reputation systems such as EigenTrust [22] can be used against Byzantine failures, but they cannot prevent a coalition of malicious nodes from denouncing a correct node. Finally, trusted computing platforms like TCG/Palladium can detect failures that involve software modifications, but force users to exclusively run certified software.

A related problem, Byzantine consensus in asynchronous message-passing systems, has been shown to require at least 3f +1 processes to be solvable in several system models (e.g., with failure detectors, partial synchrony or randomization) [23]. Recently a couple of solutions to implement Byzantine fault-tolerant state-machine replication using only 2f + 1 replicas have appeared [24]. This reduction from 3f + 1 to 2f +1 is possible with a hybrid system model, i.e., by extending the system model with trusted/trustworthy components that constrain the power of faulty processes to have certain behaviours.

Our algorithm differs from previous work in using disjoint paths to detect Byzantine Faults between two nodes (sender and destination). The proposed algorithm deals with a large variety of Byzantine Faults including spurious messages, message omission and replay, and modification/corruption of message metadata (parameters) that are not dealt with by existing schemes. Unlike most work on Byzantine Faults, this work focuses on detecting Byzantine Faults rather than tolerating them, therefore, we have weaker set of assumptions, i.e. it is only enough to have one non-faulty path for detection. The same algorithm can be extended to tolerate faults but with stronger assumption. The algorithm uses disjoint paths to detect faults between two nodes only unlike other work [25] that find all faulty nodes. This allows to have more than n=3 faulty nodes in the network. The proposed algorithm deals with a large class of faults including message meta-data and content alteration, omission faults, spurious messages and message replay. Unlike cryptographic coding techniques, the proposed algorithm does not require high computational power.

3 Properties of BFD Protocol

In this section, we first define the properties that any system that detects Byzantine Faults needs to satisfy. We provide detection conditions that are necessary and sufficient to detect Byzantine Faults in the system and then prove that they are necessary and sufficient.

A system that detects a BF satisfies the following liveness and safety properties following [14], [17], and [26].

3.1. Eventual Strong Completeness (Liveness)

Every message sent by a non-faulty node and perturbed (but not omitted) by an intermediate node or a link is eventually received by the destination. In addition, Every non-faulty node that sends a message perturbed or omitted by an intermediate node or a link eventually discovers that its message has experienced a fault prior to reaching its destination..

3.2 Eventual Strong Accuracy (Safety)

No non-faulty node receives a perturbed message and wrongfully considers it as a non-perturbed message. In addition, no non-faulty node wrongfully discovers that a message has been perturbed.

4 Concluding Remarks

We presented a new approach using disjoint paths to detect Byzantine Faults (BFs) in computer networks to implement reliable channels and showed the necessary conditions to detect Byzantine Faults. It is an open problem to investigate ways to use disjoint paths to detect other forms of Byzantine Faults. In addition, investigation of various ways to combine the proposed algorithm with the existing ones such as finite state duplication, and cryptographic and network coding techniques to deal with Byzantine Faults is an open problem.

References

1. L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," ACM Trans. Program. Lang. Syst., vol. 4, no. 3, pp. 382-401,1982.
2. D. Dolev, "The byzantine generals strike again," Stanford University, 1982.
3. M. Castro and B. Liskov, \Practical byzantine fault tolerance and proactive recovery," ACM Trans. Comput. Syst., vol. 20, no. 4, pp. 398-461, 2002.
4. T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks with random network coding," Information Theory, IEEE Transactions on, vol. 54, no. 6, pp. 2798-2803, June.2008.
5. K. P. Kihlstrom, L. E. Moser, and P. M. Melliarsmith, "The secure ring protocols for securing group communication," in Proceedings of Hawaii International Conference on System Sciences, 1998, pp. 317-326.
6. T. Ho, B. Leong, R. Koetter, M. Madard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in Proceedings of the 2004 IEEE International Symposium on Information Theory (ISIT), June. 2004.
7. F. Greve, P. Sens, L. Arantes, and V. Simon,"A failure detector for wireless networks with unknown membership," in Euro-Par 2011 Parallel Processing. Springer, 2011, pp. 27-38.
8. B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in Proceedings of the 1st ACM workshop on Wireless security. ACM, 2002, pp. 21-30.
9. Y. Desmedt, "Unconditionally private and reliable communication in an untrusted network," in Theory and Practice in Information-Theoretic Security, 2005. IEEE Information Theory Workshop on, Oct 2005, pp. 38-41.

10. N. Cai and R. Yeung, "Secure network coding," in Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on, 2002, pp.323.
11. Y. Wei, Z. Yu, and Y. Guan, "Efficient weakly secure network coding schemes against wiretapping attacks," in Network Coding (NetCod), 2010 IEEE International Symposium on, June 2010, pp. 1-6.
12. Z. Zhang, "Linear network error correction codes in packet networks," Information Theory, IEEE Transactions on, vol. 54, no. 1, pp. 209-218, Jan. 2008.
13. S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Mdard, "Resilient network coding in the presence of byzantine adversaries," in Proceedings 26th Annual IEEE Conference on Computer Commun., INFOCOM, 2007, pp. 616-624.
14. T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," J.ACM, vol. 43, no. 2, pp. 225-267, 1996.
15. D. Malkhi and M. Reiter, "Unreliable intrusion detection in distributed computations," in CSFW'97: Proceedings of the 10th IEEE workshop on Computer Security Foundations. Washington, DC, USA: IEEE Computer Society, 1997, p. 116.
16. A. Doudou, B. Garbinato, R. Guerraoui, and A. Schiper, "Muteness failure detectors: Specification and implementation," in EDCC-3: Proceedings of the Third European Dependable Computing Conference on Dependable Computing. London, UK: Springer-Verlag, 1999, pp. 71-87.
17. K. P. Kihlstrom, L. E. Moser, and P. M. MelliarSmith, "Byzantine fault detectors for solving consensus," The Computer Journal, vol. 46, p. 2003, 2003.
18. F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," ACM Computing Surveys, vol. 22, pp. 299-319, 1990.
19. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth, \Bar fault tolerance for cooperative services," in 20th Symposium on Operating Systems Principles (SOSP). ACM, 2005, pp. 45-58.
20. L. Alvisi, D. Malkhi, E. Pierce, and M. Reiter, "Fault detection for byzantine quorum systems," Parallel and Distributed Systems, IEEE Transactions on, vol. 12, no. 9, pp. 996-1007, sep. 2001.
21. D. Denning, "An intrusion-detection model," Software Engineering, IEEE Transactions on, vol. SE-13, no. 2, pp. 222-232, Feb. 1987.
22. S. D. Kamvar, M. T. Schlosser, and H. Garciamolina, "The eigentrust algorithm for reputation management in p2p networks," in Proceedings of the 12th International World Wide Web Conference (WWW 2003), 2003.
23. M. Merideth, A. Iyengar, T. Mikalsen, S. Tai, I. Rouvellou, and P. Narasimhan, "Thema: Byzantine-fault-tolerant middleware for webservice applications," in Reliable Distributed Systems, 2005. SRDS 2005. 24th IEEE Symposium on, Oct 2005, pp. 131-140.
24. M. Correia, G. S. Veronese, and L. C. Lung, "Asynchronous byzantine consensus with 2f+1 processes," in SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing. New York, NY, USA: ACM, 2010, pp. 475-480.
25. G. Liang and N. H. Vaidya, "Byzantine broadcast in point-to-point networks using local linear coding," in Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing, ser. PODC '12. New York, NY, USA: ACM, 2012, pp. 319-328. [Online]. Available: http://doi.acm.org/10.1145/2332432.2332492
26. A. Haeberlen, P. Kouznetsov, and P. Druschel, "The case for byzantine fault detection," in Proceedings of the 2nd Conference on Hot Topics in System Dependability - Volume 2, ser. HOTDEP'06. Berkeley, CA, USA: USENIX Association, 2006, pp. 5{5. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251014.1251019