

A Study of Routing Protocol with Byzantine Robustness

Dr. Koshidgewar Bhasker Gangadhar

Asst. Profesor in Computer Science Vai. Dhunda Maharaj Deglurkar College, Degloor Dt. Nanded (M.S)-
INDIA

Mail Id. bhasker149@gmail.com

DOI-

Abstract

We present a routing protocol with Byzantine robustness and detection. The protocol utilizes a topological map and a packet forwarding mechanism with fault detection. The correctness of the protocol is based on authentication of data and control packets. We also present how such authentication can be done efficiently using Message Authentication Codes.

Keywords: *Byzantine robustness, Topological, Fault detection, Message Authentication Codes.*

Introduction

We consider the problem of routing in a highly adversarial environment where the adversary has penetrated in the routing infrastructure, i.e. controls one or more routers. Such adversary can cause network-wide disruption by, for example, injecting false or confusing routing control information into the network. Even if routers utilize a topological map rather than performing topology (or route) discovery, the adversary can cause network-wide disruption by making arbitrary routing decisions or congesting the routers by flooding the network with spurious packets. It can also modify, replay or simply discard packets coming from other routers. Our goal is to mitigate the impact of such adversary. We present a protocol that is able to route packets from source to destination, provided that a non-faulty path exists between them, where a fault can be either malicious or the outcome of an impairment in a link or router. The protocol is efficient, in that it routes over a single path, rather than several, can support links of bandwidth on the order of Gbps at low incremental cost, has low processing requirements on both data and control packets, as it relies on Message Authentication Codes for authentication, and detects faults fast, as faults are detected on a per packet basis, rather than, for example, being detected from a periodic fault detection mechanism. The protocol is directly applicable to existing infrastructure networks of moderate size on the order of 50-100 routers.

It constitutes a framework, however, that can be applied to larger networks or wireless ad hoc ones.

RELATED WORK

Perlman [11] classifies network failures into two types: (i) simple and (ii) Byzantine. A simple failure is a failure whereby some node(s) and/or link(s) become simply inoperative, whereas in a Byzantine failure some node(s) and/or link(s) become faulty, yet they, and possibly the network, continue to operate, but incorrectly. We say that a routing protocol is Byzantine robust if it is capable of delivering any packet from its source to its

destination as long as a non-faulty path exists between them. We also say that a routing protocol has Byzantine detection if faulty network components can be identified. Perlman proposed two types of secure routing protocols. The first type is based on use of (a) a flooding-based routing protocol, (b) reserved buffers, and (c) digital signatures. Flooding-based routing ensures that a packet will traverse every link and hence reach its intended destination, as long as a non-faulty path exists. Reserved buffers, together with digital signatures, ensure that packets will not be dropped because of congestion of a node by excessive trace of packets (which may arise, for instance, in a DoS attack): digital signatures authenticate the source of each packet, and each buffer should be allocated to accommodate a packet from its intended source. Thus, this routing protocol of Perlman is Byzantine robust in the sense defined above. The main deficiencies of this protocol are the excessive communication overhead, due to flooding, the excessive processing overhead, due to digital signatures, and the lack of Byzantine detection.

The second type of secure routing protocol by Perlman is based on use of (a) a link state routing protocol, (b) reserved buffers, and (c) digital signatures. The reserved buffer and digital signatures serve the same purpose as in the first protocol. A basic idea of link-state routing is that each router first discovers its neighbors and the state of their incident links, and then broadcasts this information to all other routers in the network, using link state advertisement (LSA) packets. Unlike the flooding-based routing protocol, all routers now have a topological map, i.e., explicit knowledge of the network topology. If we assume that there will be no more than k failures in the network, then forwarding a packet over $k + 1$ disjoint routes should guarantee successful delivery of the packet. Note, however, that this routing protocol is not Byzantine robust: the existence of a non-faulty path does not imply existence of $k + 1$ disjoint routes. The disadvantage of this protocol is its processing overhead, its communication

overhead (routing each packet over several paths) and the weak Byzantine detection (identification of faulty paths).

Herzberg and Kutten [5] have proposed the combined use of acknowledgements, timeouts and fault announcements, to detect packet forwarding faults and have recognized its potential to detect Byzantine faults. They present one communication optimal and one time optimal protocol as well as protocols that trade-off communication and time optimality. The protocols are presented in an abstract model, a realization of which is our routing protocol of Section 3.

Herzberg and Kutten leave the construction of a routing protocol from their fault detection protocols an open problem (We use the term routing protocol in its broad sense, to refer to the function of delivering packets from source to destination). For example they do not consider issues such as authentication mechanisms, reserved buffers and sequence numbers. Our routing protocol could use any such fault detection protocols. However, we note that both communication and time optimal fault detection protocols have the same worst case fault detection time in our routing protocol, triggered by an adversary that is dropping destination ACKs rather than packets. Since the source cannot distinguish whether the packet or the ACK was dropped, from a routing protocol perspective the same packet may have to be retransmitted. We therefore argue that time complexity should be addressed at the routing protocol level rather than with the fault detection mechanism alone. The importance of considering the routing protocol is also argued by the fact that Herzberg and Kutten imply that fault announcements should have global significance. In our routing protocol fault announcements should only be relevant to the source as faulty sources can trigger fault announcements for non-faulty links. Bradley et al. [3] propose a protocol for detecting and avoiding routers that are dropping or misrouting packets. It is based on (a) link state routing, and (b) the “conservation of flow” principle. They assume that there is at least one good neighbor to an adversarial router that may drop packets. The conservation of flow can be tested if we let the routers count the number of bytes which enter and leave their interfaces and they announce this information periodically. When a router detects that a neighbor router is misbehaving, it invalidates the link to this router. Protection against traditional DoS in which misbehaving routers congest the network is not considered. The fault detection mechanism is triggered periodically, which results in delayed detection of faults. Awerbuch et al. [2] propose a protocol that detects packet forwarding faults and attempts to route around them. The protocol uses a

probing technique and acknowledgements not only by the destination but also by intermediate nodes. The additional overhead of several ACK’s for every packet is not justified in the paper. Protection against traditional DoS in which misbehaving routers congest the network is also not considered. An interesting feature of this protocol is that the fault detection mechanism is enabled only if throughput is sufficiently low. Marti et al. [7] propose a protocol for detecting and avoiding routers that are dropping or modifying packets in a mobile ad hoc network (MANET). It is based on the Dynamic Source Routing (DSR) protocol. In a MANET, neighbor routers share the communication medium, therefore routers pirating in promiscuous mode can verify whether their neighbor routers drop or modify packets. In [7], detection of such misbehavior is followed by an announcement of the misbehaving router. This protocol is vulnerable, among other things, to collusion and false misbehavior reports.

Other work in secure routing [4, 10, 8, 6, 13, 14] is about protecting topology (route) discovery, that although important is not the focus of this paper.

The Protocol

In this section we present a routing protocol with Byzantine robustness and detection. Byzantine robustness means that the protocol routes packets from source to destination as long as a non-faulty path exists. Byzantine detection means that the protocol identifies faulty links. We first give a definition of what constitutes a faulty component and then justify this definition.

A faulty node is a node that:

² does not follow our protocol, or

² can be impersonated by another node.

The first part of the definition captures a node that is controlled by an adversary or executes buggy code. The second part of the definition is not obvious: we associate the notion of faulty with that of malicious or harmful but in this case, the behavior of the faulty node does involve any malice. The faulty node can only be impersonated if, for example, its keys have been compromised. We cannot guarantee communication with a faulty node like this.

A faulty link is a link that:

² drops packets or

² is incident to a faulty node.

The first part of the definition is about links that have an impaired underlying communication system.

Regarding the second part of the definition, we need to observe that a link that is incident to a faulty node can only route packets either from or to this node. If the faulty node has crashed, for example, then packets cannot be routed in either direction of the

link. If the faulty node is a subverted one, then we would also like to avoid routing through this node, therefore identifying its incident links as faulty is equivalent from a routing robustness perspective to identifying this node as faulty. For performance reasons we would have liked to be able to identify faulty routers. However, we cannot tell with certainty whether a link or the downstream router is faulty, although we do not preclude certain cases where this can happen. Another reason is that a faulty router can invalidate its incident link without provision from the protocol. Therefore, if a link is detected to be faulty by our protocol, then one or more of the following statements are true: ² The upstream router is faulty. ² The underlying physical communication system is faulty. ² The downstream router is faulty. The protocol can be seen as a combination of several components, each of which is important for the protocol's correctness. These components are:

1. source routing,
2. destination acknowledgements,
3. timeouts,
4. fault announcements,
5. authentication,
6. reserved buffers,
7. sequence numbers, and
8. FIFO scheduling.

Overview of Operation

When the source router receives a packet, it first appends a source route to the packet. The source should therefore be able to obtain such a route. This can be ensured if sources are manually configured with a topological map of the network or if routers perform topology discovery by "link state routing", for example. The source, then, appends a new sequence number to the packet, larger than any of the sequence numbers that this source has used before. The source also appends an authentication tag to the packet. The purpose of this tag is to authenticate the packet to every downstream router. Finally the source sets a timeout to receive either a destination ACK or a fault announcement (FA) from a downstream router, for this packet, and forwards the packet to the first such router. When an intermediate router receives a data packet, it verifies the authenticity of the packet and makes sure that its sequence number is larger than the sequence numbers that this router has previously seen from the packet's source. If any of the checks fails, then the packet is dropped. If the checks succeed, the packet is scheduled for transmission in the appropriate outgoing link. When the packet is transmitted, the router sets a timeout to receive either an ACK or an FA for this packet. When the destination receives a data packet it verifies the authenticity of the packet and makes sure that its

sequence number is larger than the sequence numbers that it has previously seen from the packet's source. If any of the checks fails, then the packet is dropped. If the checks succeed, it delivers the packet (either to a higher level protocol or an attached user) and schedules an ACK for transmission along the reverse of the path that the packet traversed. The ACK rejects the sequence number of the packet. The destination also appends an authentication tag to the ACK whose purpose is to authenticate it to all upstream routers. If the timeout at an intermediate router res, it schedules For transmission to the upstream path an FA for the first downstream link. The FA rejects the sequence number of the packet and also bears an authentication tag, for authentication to the upstream routers. When an intermediate router receives an ACK, it verifies its authenticity and that a timeout is pending for the corresponding data packet. If the ACK is not authentic or a timeout is not pending, it drops the ACK. Otherwise it cancels the timeout and further forwards the ACK. When an intermediate router receives an FA, it verifies its authenticity, it verifies that a timeout is pending for the corresponding data packet and that the link reported in the FA is the first downstream to the node the generated it. If the FA is not authentic, a timeout is not pending, or the link is not the downstream to the router reporting it, then it drops the FA. Otherwise, it cancels the timeout and further forwards the FA. If the timeout at the source ...res, then it deletes the first downstream link from its topological map. It then calculates a new path to the destination in this new map and reprocesses the "failed" packet as if it were a new packet. If the source receives an ACK, then it assumes successful delivery of the packet. If the source receives an authentic FA, then it deletes the link in the FA, provided that this is the downstream link of the router that generated the FA. It then calculates a new path to the destination in this new map and reprocesses the "failed" packet as if it were a new packet.

Elaborating on Individual Components

We assume that each link has one a-priori reserved buffer for every source router in the network. This ensures that packets are never dropped because of congestion. If we allowed packets to be dropped because of lack of available buffer space, then FAs would also be pertinent to congested, rather than inherently faulty, links. Reserved buffers also protect against traditional DoS in which malicious sources flood the network with spurious packets. Reserved buffers are claimed through authentication and sequence numbers. Authentication ensures that the reserved buffer is allocated to its intended source. Monotonically increasing non- wrapping sequence numbers

safeguard against replay. In a replay attack, an intermediate router stores authentic packets and introduces them at a later time into the network in order to “take out” new packets. In our protocol, new packets have larger sequence numbers and priority is given to packets with larger sequence numbers. ACKs provide feedback on whether a packet was successfully delivered. Timeouts detect delivery failures. The combined use of source routing, ACKs, authentication, reserved buffers, FIFO scheduling, monotonically increasing non-wrapping sequence numbers and a timeout at the source only, is sufficient to identify whether a path is faulty. Timeouts at every intermediate router, in combination with FAs, provide feedback on faulty links as well. This is helpful for network management purposes and also aids the route selection process. Timeouts are set as the worst-case round trip time to the destination. With source routing the worst-case round trip time to the destination is known to the source and every intermediate router. We also note that FAs are only relevant to the source of a data packet and are not to be interpreted by intermediate routers. The reason is that faulty sources can trigger FAs for non-faulty links, by routing packets simultaneously in overlapping routes. Another requirement for the correctness of the protocol is that the packet processing speed (which mainly consists of verifying and generating authentication tags) should bind the link bandwidth. This is further discussed in the following section.

Authentications

Using digital signatures is the most straightforward authentication mechanism. The computational overhead associated with digital signatures is prohibitive however, especially since authentication is required on a per-packet basis. Our protocol employs Message Authentication Codes (MACs), which provide authentication for pairs of routers. Therefore, the source of a data or control packet has to authenticate it to every downstream router separately. The authentication tag bears a special structure to ensure that faulty routers cannot trigger FAs for non-faulty links. Specifically, given a path $\langle s; n_1; \dots; n_i; n_{i+1}; \dots; t \rangle$ the computation of the MAC for node n_i receives as input both the message and the MACs for nodes $n_{i+1}; \dots; t$. MACs are therefore computed sequentially from t to n_1 . If the MAC for each node received as input the message only, then if n_1 , for example, tampered with the MAC of n_{i+1} , then n_{i+1} would have dropped the packet, as not authentic, and n_i would have generated an FA for link $(n_i; n_{i+1})$, which is not faulty. Protocol correctness requires the processing speed to bound link bandwidth; otherwise packets would be dropped

before they are authenticated. Resources would therefore be denied to those packets, and FAs would be triggered for non-faulty links. An important parameter of the protocol is, therefore, the link bandwidth that it can support. If digital signatures were employed, then at most a few Mbps could be supported (at a reasonable cost). MACs allow the protocol to support bandwidth on the order of Gbps. Packet processing mainly consists of verifying the authenticity of the packet and generating either an FA or an ACK. (FAs are generated upon reception of data packets and scheduled for transmission at a later time, if necessary.) If we restrict the maximum permissible length to 10 hops, then at most 11 MAC computations are required. Bandwidth is then calculated by setting the time to compute 11 MACs equal to the transmission time of one packet. A straightforward calculation using performance measurements for the UMAC [1] reveals that a moderate speed PC can support Gbps links. The incremental cost of adding such a processor to each (direction of a) link is small if compared to the cost of a Gbps router.

Future Works

The protocol can be extended to accommodate multiple outstanding packets per route, so as to fill the pipes and increase throughput. Performance would also improve by a multi-tier mechanism, with several classes of packets. The protocol presented in this paper will be used as the top-tier protection mechanism for high priority packets and to detect faulty links. At the lower levels it would be beneficial from a performance perspective to consider protected packets without fault detection (to reduce the overhead of generating FAs) and unprotected packets. Finally, we plan to develop a protocol to isolate faulty routers, by blocking their traffic. This task is non-trivial as the fault detection mechanism identifies faulty links rather than routers.

References

1. <http://www.cs.ucdavis.edu/~rogaway/umac/>
2. B. Awerbuch, D. Holmer, C. Nita-Rotaru, H. Rubens, “An On-Demand Secure Routing Protocol Resilient to Byzantine Failures”, Proc. 2002 ACM Workshop on Wireless Security, Atlanta GA, Sept. 2002.
3. K. Bradley, S. Cheung, N. Puketza, B. Mukherjee and R. Olsson, “Detecting Disruptive Routers: A Distributed Network Monitoring Approach”, Network, Sept./Oct., 1998.
4. R. Hauser, T. Przygienda and G. Tsodik, “Lowering Security Overhead in Link State Routing”, Computer Networks, 31(8):885-894, Apr. 1999.
5. A. Herzberg and S. Kutten, “Early Detection of Message Forwarding Faults”, SIAM J. Comput., vol. 30, no. 4, pp. 1169-1196, 2000.

6. Y. Hu, A. Perrig, D. Johnson, "Ariadne: A Secure On- Demand Routing Protocol for Ad Hoc Networks", Proc. 8th Annual International Conference on Mobile Computing and Networking, Atlanta, GA, Sept. 2002.
7. S. Marti, T. Giuli, K. Lai and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks", Proc. 6th ACM International Conference on Mobile Computing and Networking, Aug. 2000.
8. S. Murphy and M. Badger, "Digital Signature Protection of the OSPF Routing Protocol", Proc. Symp. Network and Distributed System Security, pp. 93-102, 1996.
9. P. Papadimitratos and Z. Haas, "Securing the Internet Routing Infrastructure", Communications, pp. 60-68, Oct. 2002.
10. P. Papadimitratos and Z. Haas, "Secure Routing for Mobile Ad Hoc Networks", Proc. Communication Networks and Distributed Systems Modeling and Simulation Conference, San Antonio, TX, Jan. 2002.
11. R. Perlman, "Network Layer Protocols with Byzantine Robustness", Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Aug. 1998.
12. J. Saltzer, D. Reed and D. Clark, "End-to-End Arguments in System Design", ACM Trans. Computer Systems, vol. 2, iss. 4, Nov. 1984.
13. B. Smith, S. Murthy and J. Garcia-Luna-Aceves, "Securing Distance-Vector Routing Protocols", Proc. Symp. Network and Distributed System Security, San Diego, CA, 1997.
14. B. Smith and J. Garcia-Luna-Aceves, "Securing the Border Gateway Routing Protocol", Proc. Global Internet '96, London, UK, Nov. 1996.