PREVENTING NETWORK INSTABILITY CAUSED BY PROPAGATION OF CONTROL PLANE POISON MESSAGES[*]

Xiaojiang Du
Mark A. Shayman
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD

Ronald A. Skoog
Telcordia Technologies
Red Bank, NJ

ABSTRACT

In this paper, we present a framework of fault management for a particular type of failure propagation that we refer to as "poison message failure propagation": Some or all of the network elements have a software or protocol 'bug' which is activated on receipt of a certain network control/management message (the poison message). This activated 'bug' will cause the node to fail with some probability. If the network control or management is such that this message is persistently passed among the network nodes, and if the node failure probability is sufficiently high, large-scale instability can result. In order to mitigate this problem, we propose a combination of passive diagnosis and active diagnosis. Passive diagnosis includes protocol analysis of messages received and sent by failed nodes, correlation of messages among multiple failed nodes and analysis of the pattern of failure propagation. This is combined with active diagnosis in which filters are dynamically configured to block suspect protocols or message types. OPNET simulations show the effectiveness of passive diagnosis. Message filtering is formulated as a sequential decision problem, and a heuristic policy is proposed for this problem.

1. INTRODUCTION

There have been a number of incidents where commercial data and telecommunication networks have collapsed due to their entering an unstable mode of operation. The events were caused by unintentional triggers activating underlying system defects (e.g., software 'bugs', design flaws, etc.) that create the propagation mechanism for instability. These system defects are generally not known to the network providers, and new defects are constantly introduced. More importantly, these points of vulnerability can be easily triggered through malicious attack.

The goal of this research is to provide a fault management framework that can protect networks from unstable behavior when the trigger mechanism and underlying defect causing instability are unknown. There are several failure propagation mechanisms that can cause network instability [2]. This paper presents a framework to deal with one of those mechanisms -- what we call the 'poison message' failure propagation mechanism. This mechanism has resulted in large-scale failures in both telecommunication networks and IP networks. A telecommunications example is described below. We are also aware of an incident in which malformed OSPF packets functioned as poison messages and caused failure of the routers in an entire routing area for an Internet Service Provider.

1.1 A TELECOMMUNICATION EXAMPLE

On January 15, 1990 an AT&T network failure occurred [5]. It was caused by a 'poison message' propagation in the network management plane. The AT&T 4ESS switches are signaling points in the SS7 signaling network. The problem began after the 4ESS switch in New York (switch A) took itself out of service during normal recovery from a minor trunk interface problem, announcing this to adjacent 4ESS switches by network management messages. The adjacent switch processors made notations in their programs to indicate that Switch A was temporarily out of service. Upon completion of fault recovery, Switch A announced its recovery to adjacent switches by sending call setup messages. Each adjacent switch then noted recovery of A by an appropriate update in its program. A software flaw leading to network failure caused the adjacent switches to be vulnerable to certain types of disruptions for a few seconds during the period when such updates were being made. The fatal disruption occurred when switch A sent two very closely spaced call setup messages (within an interval of 15 ms) to an adjacent

[*] Research supported by DARPA under contract N66001-00-C-8037, by the Laboratory for Telecommunications Sciences and by ARDA.

switch (switch B). This trigger event caused switch B to execute some bad code and finally took itself out of service. When switch B went out of service, it repeated switch A's actions, disabling other 4ESS switches. A chain reaction was initiated causing a network instability that lasted for hours. During the first 30 minutes of the incident, about 98% of 114 4ESS switches were affected.

1.2 THE POISON MESSAGE FAILURE PROPAGATION PROBLEM

We now describe the generic mechanism of which the AT&T failure is a specific example: A trigger event causes a particular network control or management message (the poison message) to be sent to other network elements. Some or all of the network elements have a software or protocol 'bug' that is activated on receipt of the poison message. This activated 'bug' will cause the node to fail with some probability. If the network control or management is such that this message is persistently passed among the network nodes, and if the node failure probability is sufficiently large, large-scale instability can result.

Our task is to design a fault management framework that can identify the message type, or at least the protocol, carrying the poison message, and block the propagation of the poison message until the network is stabilized. We propose using both passive diagnosis and active diagnosis to identify and block the corresponding protocol or message type.

2. THE PROBLEM FEATURES

This problem has several differences from traditional network fault management problems [3]. In our problem, the failure itself propagates, and propagation occurs through messages associated with particular control plane or management plane protocols. It is also different from worms or viruses in that worms and viruses propagate at the application layer. A protocol may have a characteristic pattern of propagation. For example, OSPF uses flooding so a poison message carried by OSPF is passed to all neighbors. In contrast, RSVP path messages follow shortest paths so a poison message carried by RSVP is passed to a sequence of routers along such a path. Consequently, we expect pattern recognition techniques to be useful in helping to infer the protocol responsible for carrying the poison message. We make the following two assumptions: 1). There is a central controller and a central observer in the network. I.e., we use centralized network management. 2). The recent communication history (messages exchanged) of each node in a communication network can be recorded.

There are several ways to record the communication history. Here we assume we can put a link box at each link of the network. The link box can be used to record messages exchanged recently in the link. The link box can also be configured to block certain message types or all messages belonging to a protocol. We refer to the blocking as message or protocol filtering. Filtering may be used to isolate the responsible protocol. E.g., when failures are observed, one could configure the link box to block all suspect protocols in the control/management plane. Then the protocols can be turned on one by one. This method should be able to identify the responsible protocol, but the cost may be very large in terms of degraded control of the network. On the other hand, one could isolate the responsible protocol by blocking one protocol at a time and observing whether failure propagation was halted. The filter settings may or may not be chosen to be the same throughout the network. We suggest combining both passive diagnosis and active diagnosis to find out the poison message.

3. PASSIVE DIAGNOSIS

Passive diagnosis includes the following methods.

a. Finite State Machine Method: This is a distributed method used at a single failed node. As we know, all communication protocols can be modeled as finite state machines [1]. At the beginning of the failure propagation, only a small number of nodes fail. Then as the poison message propagates through the network, the number of failed nodes increases. The neighbor of a failed node will retrieve messages belonging to the failed node. From the message sequence for each protocol, we can determine what state a protocol was in immediately prior to failure by checking the FSM model. We can also find out whether those messages match (are consistent with) the FSM. If there are one or more mismatches between the messages and the FSM, that probably means there is something wrong in the protocol.

b. Correlating Messages: Event correlation is an important technique in fault management. We store recent exchanged messages before a node fails. Then we analyze the stored messages from multiple failed nodes. If multiple nodes are failed by the same poison message, there must be some common features in the stored messages. We need to take advantage of this. One can compare the stored messages (recently exchanged) from those failed nodes. If for a protocol, there are no common received messages among the failed nodes, then we can probably rule out this protocol. I.e., this protocol is not responsible for the poison message. On the other hard, if all failed nodes have the same final message in one protocol, we can use Bayes'

Rule to calculate the probability of the final message being the poison one. The details are given in section 6.

c. Using Node Failure Pattern: Different protocols have different failure propagation patterns. One way to exploit the node failure pattern is to use a neural network classifier. The neural network is trained via simulation. A simulation testbed can be set up for a communication network. The testbed has the same topology and protocol configuration as the real network. Then for each message type used in the network, the poison message failure is simulated. And the simulation is run for the probability of a node failure taking on different values. After the neural network is trained, it is applied using the node failure sequence as input, and a pattern match score is the output. Results of this method will be reported elsewhere.

When a poison message failure occurs, anything could happen in the failed node. We classify the failure into four cases:
1) The failure causes one (or more) mismatches between messages and the corresponding FSM. This can be detected by FSM method.
2) The failure does not cause any mismatch between messages and the corresponding FSM, but it leads the final state of the protocol to be in an "Error" state. This can also be detected by FSM method.
3) The last message in a protocol is the poison message. I.e., there are no more messages exchanged in the poison protocol between the time the node received the poison message and when it failed. This case can be dealt with by correlating messages among multiple failed nodes.
4) None of the above. – This case can be dealt with using the node failure pattern method.

The output of passive diagnosis will be a probability distribution that indicates for each protocol (or message type) an estimated probability that it is responsible for the poison message. In our simulation experiments, we have focused on correlating messages across multiple failed nodes and using Bayes' Rule to generate the probability distribution. The details are given in section 6.

4. ACTIVE DIAGNOSIS

From passive diagnosis we obtain a probability distribution vector over the possible poison protocols (or message types). Each component of this vector corresponds to a particular protocol (or message type) and gives the current estimate of the probability that it is the one carrying the poison message. Message filtering will be used for further failure identification. There are different costs associated with turning off different protocols or message types. We formulate it as a sequential decision problem.

- At each stage, the state consists of a probability distribution vector for each protocol potentially carrying the poison message, and the recent history of the node failures.
- Based on the current state, a decision (action) is made as to how to configure filters.
- When new node failures are observed, the state is updated based on the current state, action and new observation.
- Actions are chosen according to a policy that is computed off-line based on optimizing an objective function.

. Because the probability of two protocols having poison messages at the same time is very small, we assume there is only one protocol carrying the poison message when such failure occurs. We proposed a heuristic policy for the sequential decision problem: To block the single protocol (or message type) with the smallest ratio E[C]/p at each decision time step, where the E[C] is the expected cost (in terms of network performance) associated with blocking the protocol, and p is the current probability that the protocol is poison [4]. Derivation of more sophisticated filtering policies is the subject of ongoing research.

5. THE POISON MESSAGE FAILURE EXAMPLES

The poison message failures could happen in many protocols and scenarios. We construct three scenarios where Label Distribution Protocol (LDP), OSPF and BGP are the responsible protocols carrying the poison messages. In the LDP case, for some reasons (e.g. software bug) a LDP label request message can cause a receiving node to fail with some probability. Consider in an MPLS network, several dynamic Label Switched Paths (LSPs) are set up. The term "dynamic path" means that the path from ingress router to egress router is not fixed, but is found dynamically by a routing protocol. If any link or router in the path fails, the ingress router will try to find another path to the egress router.

An ingress router R_0 wants to set up an LSP to the egress router R_e. The Label Request message is sent to next router R_1. With some probability P router R_1 fails because of the poison message. If R_1 does not fail, then it will send a Label Request message to next router R_2. And with some probability R_2 will fail, and so on. If any router along the LSP fails, R_0 will try to find another path to R_e, and that may cause some other nodes to fail. Later a failed router reboots and it may be failed again by the poison message. If there are enough dynamic LSPs in the network, and if P is large enough, sustained network instability can result.

In the OSPF scenario, the poison message is a Link State Advertisement (LSA) message with one field having an unusual value. When a router receives the poison LSA message, it fails with some probability. If it fails, its neighbors will find the failure and send LSA messages to other routers. That may cause some other routers to fail leading to a chain reaction of failures. After a failed router reboots, it may receive the poison LSA messages again, and its failure may be repeated. This also causes an unstable network.

The third example involves BGP. The poison message is a normal BGP update message. In the BGP case, the failure propagation is similar to the OSPF case. A router fails because of a BGP poison message; later other BGP speakers will discover the failure and send out update messages. That could cause other BGP speakers to fail leading to failure propagation among the BGP speakers throughout the network.

6. SIMULATION RESULTS

In order to validate passive diagnosis, we have implemented an OPNET testbed to simulate an MPLS network in which poison messages can be carried by BGP, LDP, or OSPF. Different probabilities of a poison message failing a router have been tested. For each fixed probability, three simulation runs were performed. The testbed has 14 routers of which 5 are Label Edge Routers and 9 are (non-edge) Label Switching Routers. We use numbers 1,2,…,14 to denote each router in the simulation.

6.1 BGP SIMULATIONS

The BGP simulation is used to validate message correlation method and node failure pattern method. As we mentioned in Section 3, one possibility of the poison message failure is that the final message in a protocol is the poison message. If the node fails shortly after receiving the poison message, this is likely to be the case. However, just because multiple failed nodes have the same last message in a particular protocol does not guarantee that this is the culprit protocol and message type. For certain protocols, a large proportion of the messages exchanged may be of a particular type. Consequently, when a node fails it is likely that this message type will be the final one observed in that protocol prior to node failure even if it is not the cause of the node failure. We address this issue by using Bayes' Rule to get the posterior probability of the poison message given that several nodes have the same type of final message.

First we need the prior distribution of the final messages. To compute the prior distribution, we run simulations where nodes fail randomly and determine the relative frequency of the final message in each protocol. This relative frequency gives the probability of that being the final message in the protocol. There are 4 kinds of BGP messages in the OPNET simulator. The BGP message types and **prior distributions** are: BGP open message, 0.34%; BGP update message, 78.3%; BGP keep alive message, 21.2%; and BGP notification message, 0.16%. There are 11 kinds of LDP messages in the OPNET simulator. The often used message types and prior distributions are: hello message, 17.4%; initialization message, 17.4%; keep alive message, 34.8%; label mapping message, 13.0%; label request message, 13.0%; label release message 4.35%. There are 5 kinds of OSPF messages in the OPNET simulator. The OSPF message types and prior distributions are: OSPF hello message, 45.9%; OSPF database description message, 5.86%; OSPF request message, 3.53%; OSPF update message, 33.1% and OSPF acknowledge message, 9.70%.

In the BGP scenarios, the poison message is a normal BGP message. For each simulation, the poison message is either an update message, keep alive message or open message. We change the probability of a BGP message failing a node in different simulation runs. The following probability (P) values have been tested: 0.03, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00. For a node that fails, there is a small random delay between its receipt of the poison message and failure. Results from a typical simulation run with BGP keep alive message being the poison message are given in Table 1. Because there is a small random delay between the time nodes receive the poison message and when they fail, not all the failed nodes have the same final message in BGP. We use number (1,2,..) to denote message types in all tables. The mapping is: BGP: open=1, update=2, keep=3; OSPF: hello=1, request=2, update=4, ack=5; LDP: init=2, keepalive=3, mapping=4, request=5, hello=6, release=7.

Failed nodes	12	7	4	11	5
Failed time (sec.)	137.0	137.01	137.02	137.03	200.1
BGP last message	3	3	3	2	3
LDP last message	No	No	No	No	6
OSPF last message	1	1	4	1	1

Failed nodes	9	6	13	14	3
Failed time (sec.)	200.2	200.3	242.1	242.1	242.2
BGP last message	3	3	3	3	3
LDP last message	No	No	No	No	No
OSPF last message	1	4	1	1	1

Table 1. BGP Simulation Data (P = 0.4)

From Table 1 we can see that some of the failed nodes do not have LDP messages. So we can rule out LDP for sure--i.e., LDP is not the protocol carrying the poison message.

Now we need to consider BGP and OSPF. p(keep) is used to denote the prior probability of {keep alive message is the final message in a failed node}. Assume the distribution of final messages at different nodes is independent and identical. Then we can use the multinomial distribution to get the probability of N nodes having certain final message distribution. Considering node failure time in Table 1, we can regard the first batch of failed nodes as the four nodes: 12,7,4,11. Applying the multinomial distribution to those four nodes in BGP, we have: P(keep, keep, keep, update)

$= 4!/(3!1!)*$ p(keep)^3 p(update) $= 0.0298 \equiv P_a$ (1)

We use data from other simulations in which the BGP keep alive message is the poison message to compute the following conditional probabilities:

P(keep alive message is the final message | BGP keep alive message is poison) = 0.9375.

P(update message is the final message | BGP keep alive message is poison) = 0.0625.

Next we want to get the probability

P(keep, keep, keep, update | BGP keep alive message is poison message) $\equiv P_b$

Here we make a simplifying assumption: Given BGP keep alive message is poison, the distribution of final messages at different nodes is independent. (The reasonableness of this assumption requires further investigation.) Then by the multinomial distribution, we have:

$P_b =$ 4!/(3!1!)*P(keep alive |BGP keep alive message is poison) ^{3} *P(update |BGP keep alive message is poison)=0.206 (2)

Next consider the prior probability for each message type to be the poison message. Since we have no prior knowledge which message is the poison one, we use a uniform distribution over all message types for all protocols. I.e., the prior probability:

P(BGP keep alive message is poison)

= P(OSPF hello message is poison) = …= ∂ (3)

Then by Bayes' Rule,

P(BGP keep alive message is poison | keep, keep, keep, update) * P(keep, keep, keep, update)

= P(keep, keep, keep, update | BGP keep alive message is poison)*P(BGP keep alive is poison)

Combining (1), (2), (3), we have

$P_1 \equiv$ P (BGP keep alive message is poison | keep, keep, keep, update) $= \partial * P_b / P_a = 6.903 * \partial$ (4)

Similarly, we can get the posterior probability for other message types. The results are as follows: BGP update message: $P_2 = 0.031 * \partial$. OSPF hello message: $P_3 = 3.295 * \partial$, OSPF update message: $P_4 = 0.366 * \partial$.

Since we assume there is only one kind of poison message, $\sum_i P_i = 1$. If we only suspect the above four message types, then $P_1 + P_2 + P_3 + P_4 = 1$. We have the following posterior probabilities: $P_1 =$ 0.65, $P_2 =$ 0.003, $P_3 =$ 0.31, $P_4 =$ 0.037.

Of course we can include more message types in the calculation. Actually other probabilities are even smaller than P_2 , and P_1, P_4 do not change much. The results show that if BGP keep alive message is the poison message, data from only 4 failed nodes can generate a good probability distribution. But in some cases one needs to collect more information (i.e., wait for more failed nodes) to get a good probability distribution, as is the case discussed next when BGP open message is poison. We also ran simulations where the poison message is BGP open message or update message. After similar calculation, we have the following results.

Poison Message	BGP open	BGP update	BGP update
# of failed nodes	4	5	10
BGP open	**0.987**	0.000	0.000
BGP keep alive	1.1×10^{-10}	0.002	0.008
BGP update	0.012	**0.497**	**0.577**
OSPF hello	0.001	0.478	0.392
OSPF update	8.3×10^{-8}	0.023	0.030

Table 2. Posterior Probability in BGP Simulation

In Table 2, row 2 is the number of failed nodes used in calculation, and data in row 3 through row 7 are the posterior probabilities of different message types. We can see that BGP open message has a very high posterior probability (0.987) when open message is the poison one. This is because the prior probability of having open message as final message is very small (0.34%). The posterior probability also depends on how many failed nodes are included in the calculation. Column 3 indicates that when BGP update message is the poison message and only 5 failed nodes are considered, the probability of OSPF hello message being poison is very close to that of BGP update message so it not clear which message is the poison one. We can wait and more nodes will fail, so more information can be collected and used to calculate the posterior probability. Column 4 shows that when data from ten failed nodes is used, the probability of BGP update message is much larger than that of other message types. The OSPF simulations provide similar results and are omitted here.

6.2 LDP SIMULATIONS

Ten dynamic LSPs are set up in the network. The LSP Recovery Parameters are set to reroute if there is a link or node failure along the LSP. OSPF is used to implement routing and rerouting of LSPs. The LDP simulation is used to validate the FSM method. We modify the OPNET LDP model so that when a Label Request message is received, with some probability there is a FSM change by which the router goes from state "response" to "release", instead of going to state "establish" (which is the normal case). Then after a (relatively large) random time, the router fails. Part of the result from a typical LDP simulation is shown in Table 3.

Failed nodes	1	2	3	5	1
Failed time (sec.)	185.4	187.6	188.3	189.2	305.4
BGP last message	2	1	2	2	2
LDP last message	5	7	7	3	4
LDP FSM mismatch	yes	yes	yes	yes	yes
OSPF last message	1	1	1	1	5

Table 3. LDP Simulation Data

Since software is implemented according to the corresponding protocol specification, most of the time the messages should match the protocol's FSM. If a FSM change is found in some protocols, it strongly suggests that there is some kind of failure in that protocol. If most of the failed nodes have FSM mismatches in one protocol, then we can assign a large probability to that protocol. Consequently, in the LDP simulation above, we can conclude with a high degree of confidence that LDP is the protocol responsible for carrying the poison message. If there is no FSM change in LDP, then we can still use Bayes' Rule (as in 6.1) to get a posterior probability distribution over the possible poison message types. The results are given below.

BGP open	BGP update	OSPF hello	OSPF update	LDP request	LDP mapping	LDP release
0.034	0.038	0.170	0.008	0.094	0.023	**0.632**

Table 4. Probability Distribution in LDP Simulation

In the LDP simulation, there is a relatively large random delay between a node receiving a poison message and failing. This explains why many LDP final messages are not the poison message (label request message), and why the largest probability is not the poison message but instead is LDP label release message. In this case, we did not locate the exact message type; however, we still get a large probability for LDP -- the responsible protocol. Actually, in some situations, by passive diagnosis we can only locate the responsible protocol instead of the exact message type. This may still be satisfactory since if we find the responsible protocol, we can block the protocol and hence stop failure propagation. Then we will have enough time to use additional diagnostic techniques to find out the exact poison message.

7. SUMMARY

We have discussed a particular failure propagation mechanism--poison message failure propagation--and provided a framework to identify the responsible protocol or message type. We have proposed passive diagnosis, which includes the FSM method applied at individual failed nodes, correlating protocol events across multiple failed nodes and using node failure pattern. Bayes Rule is then used to generate a probability distribution over the possible message types or protocols. If passive diagnosis cannot solve the problem by itself, it can be augmented by protocol or message type filtering, which is formulated as a sequential decision problem. We implemented an OPNET testbed where BGP, LDP and OSPF can carry poison messages. Our simulations demonstrate the effectiveness of passive diagnosis. I.e., passive diagnosis can either find the poison message or provide a good probability distribution that can be used to determine the initial filter configurations for active diagnosis. This is an interesting and challenging problem. Our next tasks include: (1) Implementation of a neural network classifier for node failure pattern recognition, and (2) Implementation of a heuristic policy and rollout algorithm for the sequential decision problem.

REFERENCE

[1] A. Bouloutas, et al, "Fault identification using a finite state machine model with unreliable partially observed data sequences," *IEEE Tran. Communications*, Vol.: 41 Issue: 7, pp1074–1083, July 1993.
[2] R. Skoog et al., "Network management and control mechanisms to prevent maliciously induced network instability," *Network Operations and Management Symposium*, Florence, Italy, April 2002, to appear.
[3] H. Li and J. S. Baras, "A framework for supporting intelligent fault and performance management for communication networks", *Technical Report, CSHCN TR 2001-13*, University of Maryland, 2001.
[4] M.A. Shayman and E. Fernandez-Gaucherand, "Fault management in communication networks: Test scheduling with a risk-sensitive criterion and precedence constraints," *Proceedings of the IEEE Conference on Decision and Control,* Sidney, Australia, December 2000.
[5] D. J. Houck, K. S. Meier-Hellstern, and R. A. Skoog, "Failure and congestion propagation through signaling controls". *In Proc. 14th Intl. Teletraffic Congress*, Amsterdam: Elsevier, 367–376, 1994.