Detection of Invalid Routing Announcements in RIP Protocol

Dan Pei
Computer Science Department
UCLA
Los Angeles, California 90095
Email: peidan@cs.ucla.edu

Dan Massey
Suite 200, 3811 North Fairfax Drive
USC/Information Sciences Institute
Arlington, Virginia 22203
Email: masseyd@isi.edu

Lixia Zhang
Computer Science Department
UCLA
Los Angeles, California 90095
Email: lixia@cs.ucla.edu

***Abstract*— Traditional routing protocol designs have focused solely on the functionality of the protocols and implicitly assume that all routing update messages received by a router carry valid information. However operational experience suggests that hardware faults, software implementation bugs, operator misconfigurations, let alone malicious attacks can all lead to invalid routing protocol announcements. Although several recent efforts have developed cryptography-based authentication for routing protocols, such enhancements alone are rendered ineffective in the face of faults caused by misconfigurations or hardware/software errors. In this paper we develop a simple routing update validation algorithm for the RIP protocol, RIP with Triangle theorem checking and Probing (RIP-TP). In RIP-TP routers utilize a triangle theorem to identify suspicious new routing announcements, and then use probing messages to verify the correctness of the announcements. We have evaluated the effectiveness of RIP-TP through simulation using various faulty node behaviors, link failure dynamics and network sizes. The results show that, with an overhead as low as about one probing message per received update message in the worst case, RIP-TP can effectively detect 95% or more invalid routing announcements.**

I. Introduction

Up to now routing protocol designs have focused solely on the functionality of the protocols: how to calculate the best paths, how to prevent looping and speed up convergence after topological changes. It is assumed implicitly that all routers operate correctly according to the specification, and all routing messages received by each router carry valid update information. Although several recent efforts have developed cryptography-based authentication for various routing protocols, these enhancements alone are rendered ineffective in the face of faults caused by misconfigurations or hardware/software errors. To make the routing system more resilient to unexpected faults, we believe that routing protocols must be designed with update validation capabilities. As a proof of feasibility, in this paper we develop such an update validation algorithm for the RIP protocol [1].

Even though none of the existing routing protocols have been designed with invalid routing update detection, their design differences create different potentials for fault detection. The various routing protocols currently used in the Internet can be divided into 3 general classes: distance vector protocols (e.g. RIP), link state protocols (e.g. OSPF [2]), and path vector protocols (BGP [3]). In a link state routing protocol, each node learns the state of the entire network topology, whereas in distance vector and path vector protocols each node only has partial connectivity information which is the output of a (potentially faulty) neighbor node's routing decision process. Because of this, link-state protocols are generally considered more promising for detecting faults [4]. Path vector routing protocols provide partial information that could be exploited for fault detection. For example, [5] uses this partial information to detect invalid paths and improve BGP convergence.

Nodes running distance vector protocols only have information regarding the connectivity to its direct neighbor nodes, thus they are considered least capable of fault detection. A router announces to its directly connected neighbor nodes its shortest distance to all destinations. Unlike link-state protocols where a node computes its shortest path to the destination based on the network topology, a node running a distance-vector protocol computes its shortest path based on distance updates from its neighbors; it has no direct information regarding the network topology beyond the immediate neighbors. [4] argues that distance vector protocols are poor candidates for fault detection because a node running the distance vector protocol has no way to verify the validity of any distance information regarding remote connectivity. One real world example happened to the distance vector protocol used in the early ARPANET [6]. Due to a rare memory fault, an east coast router advertised a zero cost route to UCLA, and other routers learned and believed this(invalid) route. The traffic to UCLA was then sent along the false route until it reached the faulty router and this router simply dropped the UCLA traffic.

In this paper, we present a novel approach, named ***RIP-TP***(**RIP** with ***T****riangle theorem* checking and **P**robing), to detect invalid routing announcements in the RIP protocol. *RIP-TP* can be implemented either in routers or on dedicated detection devices auxiliary to routers. *RIP-TP* is compatible with current RIP standard [1], and incrementally deployable. In particular, a router can detect faulty updates by implementing *RIP-TP* even when no other routers have deployed *RIP-TP*. We have evaluated the effectiveness of *RIP-TP* through simulation using various faulty node behaviors, link failure dynamics, and network sizes. The results show that, with an overhead as low

as about one probing message per received RIP update message in the worst case, *RIP-TP* can effectively detect 95% or more invalid RIP announcements. Had RIP-TP been implemented as part of the ARPANET routing protocol, the above mentioned invalid routing announcement due to memory fault could have been detected before black holing the traffic to UCLA.

The remainder of the paper is organized as follows. Section II presents the triangle theorem. Section III presents the probing message technique used to verify distance information and discusses some of our design choices. Section IV provides the simulation results. Section V reviews the related work and we conclude the paper in Section VI.

II. The Triangle Theorem For RIP

A network running the Routing Information Protocol (RIP) consists of routers (nodes) and links between them. Each link is assigned a weight and each router maintains a routing table with an entry for each destination. Router R's routing table entry for destination i consists of $[Dist(R,i), Nexthop(R,i)]$: the shortest distance to i and next hop to reach i, respectively. To construct the routing table, neighboring routers exchange their distances to destinations using RIP update messages periodically (every 30 seconds).

For presentation simplicity, in this paper we use the router ID to denote a destination network that is directly connected to the router. But generally speaking, a router ID is not the identifier for the networks attached to it; the latter are represented by IP address prefixes. This simplification allows a clear and concise description of the concepts; the actual implementation is easily adapted to use actual network prefixes. This paper also assumes that each link has a cost of 1. In other words, the hop-count is used as the distance metric, a common practice in networks using RIP as their routing protocol [1].

A. Triangle Theorem Checking

In a network running a shortest path routing protocol, when the routing protocol has reached a stable state, a *triangle theorem* holds for any set of 3 nodes. The triangle theorem states that the distance between one pair of the 3 nodes must be equal or less than the sum of the distances of the other two pairs. More precisely, for any 3 nodes a, b, c, it must be the case that $dist(a,c) \leq dist(a,b) + dist(b,c)$. Note this theorem follows directly from the definition of a shortest path from a to c.

In the standard RIP implementation [1], router R believes any new distance information learned from an update ($Dist(A,i)$). In contrast, RIP-TP applies the triangle theorem check against this new distance information. Suppose node R receives a new RIP update from A including $Dist(A,i)$. Router R first checks whether this route meets the triangle requirement, as illustrated in Figure 1(a). The triangle theorem states that we must have $Dist(R,i) \leq Dist(R,A) + Dist(A,i)$. If this condition is not met, the new update causes a violation of the triangle theorem. A triangle theorem violation indicates an inconsistency in the distance information, but does not identify which of the 3 distances caused the inconsistency. Furthermore, a triangle theorem violation may be due to temporary delay or loss in update message propagation, or as a result of a faulty distance being injected. The first step in verification is to identify between which two nodes lies the potentially invalid distance value, followed by a probing message for validation, as described in the following Section.

In the above example shown in Figure 1(a), if $Dist(R,i) \leq Dist(R,A) + Dist(A,i)$ does not hold true, then either $Dist(R,i)$ or $Dist(A,i)$ is incorrect[1]. Instead of blindly accepting the new update $Dist(A,i)$, as the case in the standard RIP implementation, RIP-TP marks $Dist(A,i)$ as *potentially invalid* and proceeds with a second triangle theorem check, as illustrated in Figure 1(b). Let $Nexthop(R,i) = B$, node R checks whether $Dist(B,i) \leq Dist(B,A) + Dist(A,i)$. Although R does not know $Dist(B,A)$ directly, since both A and B are neighbors of R, R can conclude that $Dist(B,A) \leq 2$.

Based on the result from the two triangle theorem checks, we can classify each new distance information $Dist(A,i)$ from a RIP update into the following 3 classes:

- A distance that passes check-1 is accepted without further checking.
- A distance that fails only check-1 is *potentially invalid*.
- A distance that fails both check-1 and check-2 is *probably invalid*.

B. Generalizing the Triangle Theorem

The triangle theorem can be easily extended to other types of routing metrics. Check-1 ($Dist(R,i) \leq Dist(R,A) + Dist(A,i)$) and check-2($Dist(B,i) \leq Dist(B,A) + Dist(A,i)$) should hold true not only for hop-count metric, but also for any metric which are always larger than zero. Furthermore, the triangle theorem can be extended to other distance vector based routing protocols, such as the Distributed Bellman Ford Protocol [7], in which each router keeps the routes learned from all the neighbors. Therefore, node B in check-2 can be R's any neighbor other than A, not necessarily just node $Nexthop(R,i)$ as in RIP.

III. Verification through Probing Messages

By checking the triangle theorem as described in the previous section, a router can detect distances reported by new update messages that are either *potentially invalid* or *probably invalid*. Since the triangle theorem violation can be due to either transient state after legitimate route changes or invalid distances, further verification is needed. In this section we introduce a simple low overhead technique to verify the new distance.

We are particularly concerned with the *probably invalid* distances that failed both triangle theorem checks. We would also like to check the *potentially invalid* distances that failed only the first check. However, because each RIP update message can list a neighbor's distances for up to 25 destinations[2], in the

[1]Note since R received the update from A, R and A are neighbors and $Dist(R,A)$ is 1.

[2]As defined in [1], each RIP update message can contain up to 25 destinations. If a network has more than 25 destinations, each node sends multiple update messages to cover its complete routing table.

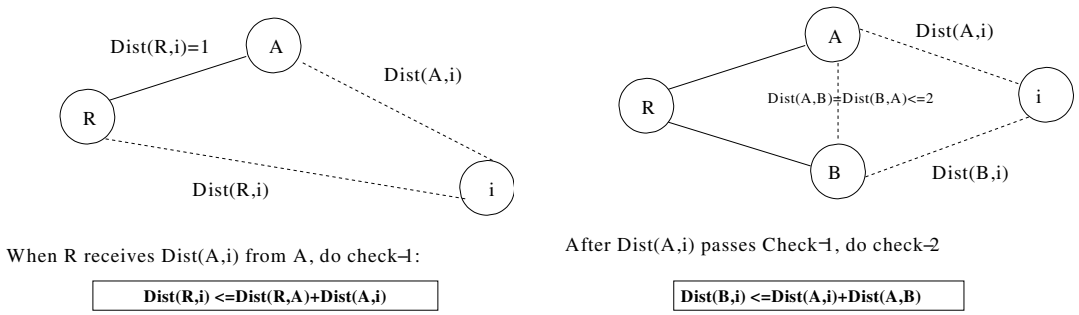


Fig. 1. Triangle Theorem in RIP

worst case all the distances could be all *potentially invalid* or *probably invalid*. To limit the number of verification overhead, each node is allowed to send at most C probing messages per update received, where C is calculated in the following way:

$$C = max(C_{min}, min(0.5 * X, C_{max}))$$

Where X is the number of *potentially invalid* distances , and C_{min} and C_{max} are configured control parameters. We used $C_{min} = 2$ and $C_{max} = 5$ in this study. In selecting up to C distances to verify, a router gives first priority to the distances that were classified as *probably invalid*. The probing messages are implemented by utilizing the existing UDP [8]and ICMP [9] protocols, as described below.

A. Sending Probing Messages

To verify a questionable distance $Dist(A, i)$, node R sends a UDP packet as a probing message, which is destinated to i with TTL set to $Dist(A, i) + 1$, and an UDP port number not used by any nodes (e.g. port number 33434 as used in traceroute software [10]). R also starts a timer $(6*(Dist(A, i)+ 1) * Hop\_Delay)$ where Hop_Delay is the average delay over a hop, either configured or obtained through R's own measurement. If a probing message arrives at the destination, the destination will send back to R an ICMP "destination unreachable" message with the reason of "unreachable port". This ICMP message from node i is considered an acknowledgment("ACK") which confirms the correctness of $Dist(A, i)$. Otherwise, if the actual distance from R to i is greater than the reported $Dist(A, i)$ value, the probing message will be dropped due to TTL exhaustion, and the node which drops the probing message will send an ICMP "time exceeded" message back to R; or if the probing is dropped by a node because it has no reachability to node i, it will send back to R an ICMP "destination unreachable" message; or if the probing message or the ICMP report is lost or delayed due to congestion or other network faults, the timer will expire. All the above 3 events are considered a negative acknowledgment("NACK"), *i.e.*, the verification is considered failed. If the Hop_Delay is 10ms, because the maximal TTL value is set to 15 in RIP, the timer value should not exceed 0.9 second. Therefore the delay caused by waiting for ICMP report or timeout before making a decision on accepting $Dist(A, i)$ is acceptable, given that RIP standard [1] already specifies a damping time of 1 to 5 seconds before consecutive triggered update messages are sent out.

B. Reacting To the Verification Results

A new distance that passes the verification test is installed in R's routing table, together with a flag bit to indicate this distance has been validated. Thus RIP-TP introduces only one bit of storage overhead per destination. A distance that failed the verification test is discarded. Note that a valid distance will be dropped if the probing or ICMP report message is lost or delayed beyond the timeout value. However because neighbor nodes re-announce their shortest distances to all destinations every 30 seconds [1], any incorrectly rejected distance will be retried after receiving future update messages.

Recall that we set an upper bound of C on the number of distances to verify. If the total number of *potentially invalid* or *probably invalid* distances in an update U exceeds C, not all of them can be checked at the time of processing U. Although our design gives first priority to checking *probably invalid* paths, the number of *probably invalid* paths alone may exceed C. To handle this scenario, we introduce two additional optimizations.

First, we discard the entire update message U if the total number of distances that received "NACK" exceeds:

$$Thresh\_Drop = max(Thresh_{min}, min(0.5 * max(C, S), Thresh_{max}))$$

Where S is the number of *probably invalid* distances, and $Thresh_{min}$ and $Thresh_{max}$ are configured control parameters, which are set to 2 and 5 in this study, respectively. $Thresh_{min}$ is necessary because we do not want to drop the entire update message if the number of invalid distances is small. If $Thresh\_Drop$ is exceeded, the router discards the entire update message (including all the valid and invalid distances) and simply waits for the next update.

Our second optimization is intended to recover from invalid distances that slip through the validation check. When the number of *potentially invalid* and *probably invalid* distances in an update message exceeds C, the router randomly picks up to C of them to verify, and the rest are not checked. Unless the entire update is rejected (see above), a *potentially invalid* distance will be accepted. Furthermore, once a *potentially invalid* distance

is accepted, future updates might not trigger violations in the triangle theorem, thus these *potentially invalid* distances could remain in the routing table indefinitely. However, because these distances do not have the verification bit set in the routing table, they are candidates for verification next time a periodic update message containing the same destinations arrive. A node R may verify up to C distances per received update message, when the update contains less than C *potentially invalid* or *probably invalid* distance, R will pick among those destination entries in the routing table that do not have the verification bit set. Unchecked entries remain as candidates for checking until the validation bit is set. Therefore even those paths that are not checked initially will be verified eventually over time.

Finally, the verification results of one router can be shared among neighbor nodes. We utilize a 16-bit reserved field in RIP update message for this purpose and a value of Y in this field indicates that the first Y entries in the update message have been verified[3]. Note that this verification is only piggybacked in the update between neighbor nodes, increasing the power of triangle theorem checking without introducing additional transmission overhead.

C. Design Summary

Our design favors simplicity, low overhead, incremental deployability and minimal impact on normal RIP operation. For example, our algorithm does not store or propagate the negative verification results (which would introduce more storage and communication overhead), and positive verification results are $only$ piggybacked on the scheduled routing update messages. A single router R can benefit from deploying this detection algorithm without any support from other routers, as long as other routers in the system respond to probing messages, which is part of standard ICMP functionality [9]. Our design does not consider the case of a malicious router intentionally modifying the probing or ICMP report messages, though it does consider the possibility of lost probing or ICMP report messages. Our design can deal with both the cases where there are only a small number of invalid distances and where most of the distances in one RIP update are invalid. In Section IV we show the benefit of triangle theorem checking and our optimization techniques by comparing it with a pure random probing approach (called RIP-RP).

IV. Simulation Evaluation

To show the benefit of triangle theorem checking and our optimization techniques, instead of comparing RIP-TP against the standard RIP we choose to compare against a variant of RIP that uses pure random probing (called RIP-RP). This is because the result for the former is trivial: routers would simply accept all updates, and an invalid but shorter distance will be accepted and further propagated by a router. RIP-RP represents a simple approach in which a router R attempts to verify at most K (a configured parameter) randomly chosen destinations from each update message. RIP-RP neither propagates verification results to neighboring routers nor uses any other optimization techniques. However note that a RIP update message may contain entries to destinations with infinity distances, especially in cases of updates triggered by link failures which may only report unreachable destinations due to the failure. RIP-RP does not consider those entries that have infinity distances when randomly choosing entries from the update to probe. Thus the actual number of probing messages RIP-RP sends per received update will be less than K if an update has less than K non-infinity entries.

We implemented *RIP-TP* and *RIP-RP* in the IRLSim simulator [11]. Our simulation tests use a set of flat mesh topologies which differ only in size and connectivity. A single parameter N determines both the number of nodes in the network(which is $N * N$), and the node degree(which is N, i.e., each node is connected to N neighbor nodes). One node which is closest to the center of the network topology is chosen as the single faulty node. The faulty node does not remove any destination entries. Instead, the faulty node randomly selects I destinations in its routing table, decreases their distances by 1 and puts these faulty entries in the update it sends out to each neighbor [4]. In addition to the faulty node, every second during the simulation run, with probability P, a randomly chosen link fails. A failed link may recover with a probability of 0.5 every second after its failure. Each of our simulation tests runs for 6000 seconds.

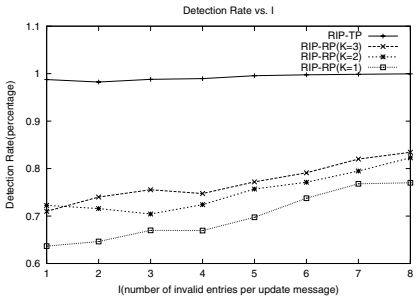
In our simulation, invalid distances include those announced by the faulty router as well as those accepted and further propagated by non-faulty routers. Invalid distances in an update may or may not change a router R's routing table. As mentioned earlier, we are only concerned with those distances that can change R's routing table. Let M be the total number of invalid distances which are marked as *potentially invalid* or *probably invalid* during simulations and let L be the total number of invalid distances that are detected and prevented from becoming new shortest paths. We define the detection rate $D = L/M$ and use this as the measure for RIP-TP's effectiveness. RIP-TP's overhead is defined as O= (total number of probing messages)/(total number of RIP update messages received).

We measure how D and O react to various values of I, P and N, and compare the results with RIP-RP (with $K = 1, 2, 3$), a pure random probing approach described above. The results in Figures 2, 3, and 4 show that, in all the simulation tests, *RIP-TP*'s detection rate D is at least 95%, and the overhead O is at most 1.1, demonstrating that *RIP-TP* is an effective and low overhead approach to faulty update detection.

Recall that the faulty node randomly selects I destinations to decrease their distances, Figure 2 shows that *RIP-TP*'s detection rate is not affected by the value of I, and the overhead increases

[3]Only routers that deployed our algorithm will set this field in their updates and we assume the sender of the update will arrange the entries by putting the destinations with verified distance first.

[4]Note that the I destinations chosen by the faulty node might be different at different times, and an invalid distance sent by the faulty router might be "corrected" later by a later update. However, since the triangle checking and probing happen when the invalid distance is *first* received, this specific simulation detail should not affect our simulation results given the definition of detection rate and overhead defined shortly.



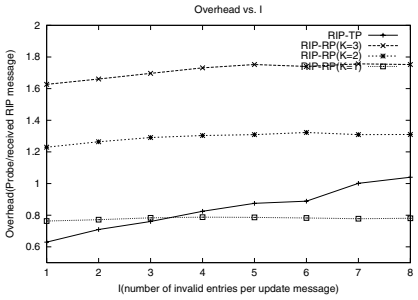
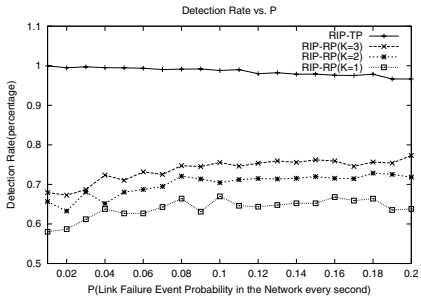


Fig. 2. N=4, P=0.1, changing I



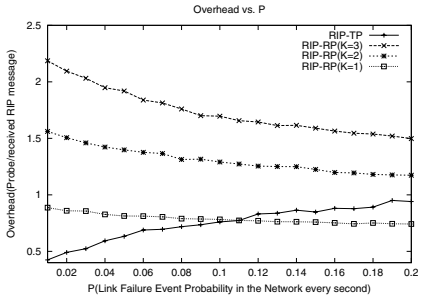


Fig. 3. N=4, changing P, I=3

only slightly as the value of I increases. A faulty router cannot increase the chance invalid entries go undetected by increasing I; the more invalid entries in an update message, the more likely a receiving node will drop the whole update due to the number of invalid entries exceeding $Thresh\_Drop$. A larger I value does lead to higher overhead, since more *potentially invalid* or *probably invalid* distances lead to more probing messages. However, Figure 2 shows that the overhead of *RIP-TP* increases slowly, and eventually it will be bounded by C_{max}, a control parameter one can adjust. Comparison of *RIP-TP* with *RIP-RP* shows that *RIP-TP* has both significantly higher detection rate and much lower overhead than random probing with K=2 or 3.

As shown In Figure 3, increased link failure rate has an impact on RIP-RP's detection rate and overhead, since failed links force nodes to select new paths which could be *potentially invalid* routes, they could also lead to probing or ICMP report message losses. As a result, the probability that $invalid$ distances slip through RIP-TP's verification increases. Although the detection rate of *RIP-TP* decreases slightly, it remains above 95% even with a link failure probability of 0.2 per second. Increased slip-through invalid distances will be further propagated and marked as *potentially invalid* and *probably invalid*, increasing the verification overhead. On the other hand, RIP specifies a damping timer for triggered updates: within T (a random value between 1 and 5) seconds, a node can send only one triggered update. When P increases, this damping timer results in more entries in one triggered update. However because a randomly chosen link fails with probability P, the number of impacted nodes, thus the number of update messages will still increase with P. The number of probing messages in RIP-TP increases with the number of *potentially invalid* and *probably invalid* distances in an update but upper bounded by $C_{max} = 5$, while RIP-RP does not adapt, and bounded by K. As an overall result, the overhead, measured as the *ratio* of number of probing messages over the total number of routing update messages, decreases in RIP-RP while increases in RIP-TP.

Figure 4 shows that *RIP-TP* is scalable as the network size increases in that its detection power is largely independent from the network size, and its overhead decreases as the network size increases.

V. Related Work

[12] proposed a secure traceroute service to help verify whether an announced path is valid. In some sense, this service is a recursive version of our probing message verification. But unlike *RIP-TP*, the secure traceroute service requires all routers on the path to be secure-traceroute-capable and cryptographically sign their portion of the path. The service also doesn't specify what triggers the verification service, but suggests that it's triggered when a normal traceroute identifies some problems while *RIP-TP* uses the triangle theorem checking to facilitate detection.

[13] proposed to place a set of sensors onto some (or all) of the links within a RIP network. Each of the sensor is provided with network topology and the positions of all the other sensors and each sensor computes all the possible paths from each router to each subnet. The sensor then analyzes the routing updates and checks the distance using its information. If a distance in RIP update is not in the legitimate range, alarm

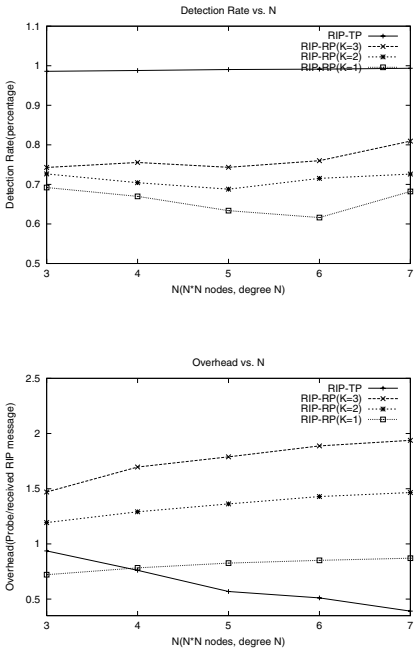


Fig. 4. changing N, P=0.1, I=3

is raised. Otherwise, a query is sent to all the sensors along all the possible paths that have this distance in order to further verify the distance. This approach might have difficulty in a very dynamic network since it requires manual configuration and global knowledge.

[14] adds a cryptographically signed PREDECESSOR (second last hop) to the distance vector protocol. A path-finding algorithm [15] then allows a node to recursively reconstruct the path and verify distances, provided every node is a destination. But this approach introduces more overhead (including cryptographic checks) and assumes every router has deployed this technique. Our approach fully utilizes the existing (but very limited) information in RIP and even a single *RIP-TP* router can benefit without the collaboration of other nodes.

[16] outlines a multi-fence framework for routing protocols resilient to faults including invalid routing announcements and provides a detailed survey of techniques for link state, path vector, and distance vector protocols. The techniques are divided into: 1)Utilize existing information propagated by the protocol; 2)Add new protocol information; 3)Add new query behavior; 4) Pre-configuring information. Our work makes use of 1) and 3), and makes limited use of 2) (neighbor-to-neighbor validation reporting). In this section, we have focused on techniques for distance vector protocols, and the reader is referred to [16] for related path vector and link state techniques.

VI. SUMMARY

In a system as large as today's Internet, faults are inevitable. Given that all Internet based communications rely on a dependable packet delivery service, it is critically important to make network routing protocols fault resilient.

In this paper we developed a simple and effective approach to detecting invalid routing announcements in RIP. Our design emphasizes effectiveness, simplicity, low overhead (both in transmission and in storage), backward compatibility with the RIP standard, and support for incremental deployment. For description clarity this paper presented RIP-TP by using hop-count as the routing metric, and we also briefly showed that the triangle theorem checking can be easily generalized to other types of link metrics and distance vector routing protocols, while generalizing the probing mechanism is part of our future work. We have evaluated the design through simulation experiment to demonstrate its effectiveness. Our work shows that, by carefully exploring the design space of invalid announcements checking, existing routing protocols can be enhanced with effective fault detection capability, as we have demonstrated with RIP, a routing protocol considered least capable of fault detection, in this paper.

ACKNOWLEDGMENTS

This work is partially supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No DABT63-00-C-1027, by National Science Foundation(NSF) under Contract No ANI-0221453, and by a research grant from Cisco Systems. Any opinions, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA, NSF, or Cisco. We would also like to thank Jelena Mirkovic for helping set up the simulator used in this study.

REFERENCES

[1] G. Malkin, "Routing Information Protocol Version 2," SRI Network Information Center, RFC 2453, November 1998.
[2] J. Moy, "OSPF Version 2," September 1998.
[3] Y. Rekhter and T. Li, "Border Gateway Protocol 4," SRI Network Information Center, RFC 1771, July 1995.
[4] R. Perlman, "Network layer protocols with byzantine robustness," Ph.D. dissertation, MIT Lab. for Computer Science, 1988.
[5] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, F. S. Wu, and L. Zhang, "Improving BGP Convergence Through Assertions Approach," in *Proceedings of the IEEE INFOCOM*, June 2002.
[6] J. M. McQuillan, G. Falk, and I. Richer, "A Review of the Development and Performance of the ARPANET Routing Algorithm," *IEEE Transactions on Communications*, vol. 26, no. 12, pp. 1802–1811, 1978.
[7] D. Bertsekas and R. Gallager, *Data Network*. Prentice-Hall, 1992.
[8] J. Postel, "User Datagram Protocol," SRI Network Information Center, RFC 768, August 1980.
[9] ——, "Internet Control Message Protocol," SRI Network Information Center, RFC 792, September 1981.
[10] V. Jacobson, "Traceroute," http://www-nrg.ee.lbl.gov/traceroute.tar.gz.
[11] A. Terzis, K. Nikoloudakis, L. Wang, and L. Zhang, "IRLSim: A General Purpose Packet Level Network Simulator," in *Proceedings of the 33rd ACM-SIAM Symposium on Discrete Algorithms*, April 2000.
[12] V. N. Padmanabhan and D. R. Simon, "Secure Traceroute to Detect Faulty or Malicious Routing," in *Proceeds of HOTNETS 2002*, 2002.
[13] V. Mittal and G. Vigna, "Sensor-Based Intrusion Detection for Intra-Domain Distance-Vector Routing," in *ACM CCS's 02*, November 2002.
[14] B. R. Smith, S. Murphy, and J. J. Garcia-Luna-Aceves, "Securing distance-vector routing protocol," in *Global Internet'96*, Februray 1997.
[15] J. Garcia-Lunes-Aceves and S. Murthy, "A Loop-Free Path-Finding Alogirthm: Specification, Verification and Complexity," in *Proceedings of the IEEE INFOCOM*, April 1995.
[16] D. Pei, D. Massey, and L. Zhang, "A Framework for Resilient Internet Routing Protocols," *IEEE Network Special Issue on Protection, Restoration, and Disaster Recovery*, 2004.