Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection

Steven Cheung
Department of Computer Science
University of California
Davis, CA 95616
cheung@cs.ucdavis.edu

Karl N. Levitt
Department of Computer Science
University of California
Davis, CA 95616
levitt@cs.ucdavis.edu

Abstract

We present a solution to the denial of service problem for routing infrastructures. When a network suffers from denial of service, packets cannot reach their destinations. Existing routing protocols are not well-equipped to deal with denial of service; a misbehaving router—which may be caused by software/hardware faults, misconfiguration, or malicious attacks—may be able to disable entire networks. To protect network infrastructures from routers that incorrectly drop packets and misroute packets, we hypothesize failure models for routers and present protocols that detect and respond to those misbehaving routers. Based on realistic assumptions, we prove that our protocols have the following properties: (1) A well-behaved router never incorrectly claims another router as a misbehaving router; (2) If a network has misbehaving routers, one or more of them can be located; (3) Misbehaving routers will eventually be removed.

1 Introduction

Through a myriad of applications, including electronic mail, WWW, and electronic commerce, computer networks play an increasingly important role. Most of the existing network security work concerns confidentiality, data integrity, user authentication, and non-repudiation, typically as associated with hosts. Until recently, very little attention was given to securing routing infrastructures. By routing infrastructures, we refer to routers and routing protocols. Denial of service for the routing infrastructures may be caused by natural faults as well as by malicious attacks. Because disabling a network can have a huge impact (e.g., time-critical information cannot be communicated) on a large scale, networks are inviting targets for sabotage.

We use a detection-response (i.e., an expansive view of intrusion detection) approach to protect networks from denial of service. In our approach, routers cooperatively diagnose each other to detect, locate, and respond to misbehaving routers. The idea of system diagnosis is not new; Preparata's, Metze's, and Chien's seminal paper [19] proposed a framework for automated system diagnosis. Our contribution is on designing tests specifically for router diagnosis and proving their detection and response properties. In simple terms, a testing router A sends a packet to a tested router B and verifies B's behavior against its expected behavior. The verification problem includes two sub-problems: determining B's expected behavior and determining B's actual behavior.

In a network that uses a dynamic routing protocol, B's behavior depends on the current state of the network. Moreover, A and B may not always share the same view of the state. Thus A may not always know the expected behavior of B. We argue that by concentrating on certain types of routing protocols (e.g., in link state routing, unlike distance vector routing, a router propagates routing updates to its neighbors as soon as it receives

[0]DRAFT: To appear in Proc. New Security Paradigms Workshop, Cumbria, UK, September 23-26, 1997.

them) and careful test assignments (e.g., choosing the tester A to be a direct neighbor of the tested B), A and B can see the same network state most of the time. (We will further justify this point in Section 8.) These approximations seem to be necessary because of the impossibility of constructing global states of distributed systems.

This paper assumes that A can determine the expected behavior of B and focuses on the second sub-problem (i.e., determining B's actual behavior). There are two basic ways to choose "test" packets—normal traffic or packets created specifically to test B. As we will see later, these two strategies give rise to different diagnosis techniques, both of which we consider. If A generates its packets to test B, a major issue is what packets A should generate to uncover the bad behavior of B, if any. Solutions may not exist in all cases. If we assume the worst-case scenario in which B could distinguish ordinary packets from those test packets, B could misbehave only on ordinary packets to avoid being detected. (This motivates us to use normal traffic to diagnose a router, which will be discussed in Section 7.) To further complicate the problem, unless the path traversed by a test packet avoids routers other than A and B. The tester A may need to collaborate with other routers and depend on their reports to diagnose B. (Section 6 discusses a technique on choosing a tester and a test packet that avoids using a third router for the diagnosis.) Using multiple routers to test a router gives rise to additional issues. First, if A uses reports from mischievous routers for its analysis, it may incorrectly deduce that B is a misbehaving router or that B is a good router. Second, the set of testing routers need to communicate without being affected by the presence of misbehaving routers in the network.

Router diagnosis seems to be very expensive if we assume the worst-case adversary. We find that there are special cases that have practical significance and are indeed solvable without incurring substantial overhead. We develop failure models that characterize the behavior and the "strength" of misbehaving routers. For example, we assign routers that misbehave permanently and those that misbehave intermittently to different failure classes, with the former being a subclass of the latter. Based on these models, we design distributed diagnosis protocols that detect and logically remove misbehaving routers. Once misbehaving routers are located, the other routers respond by reconfiguring the network to restore its operational status. It is essential that misbehaving routers cannot misuse the network reconfiguration capability to give themselves additional power to disable the network. Our protocols solve the misuse problem by only allowing a router to disconnect itself from its neighbors, which misbehaving routers can emulate by dropping or misrouting packets, yet guarantee that all misbehaving routers will eventually be removed.

The outline of this paper is as follows: Section 2 presents some denial of service examples for computer networks. Section 3 reviews related work on securing routing protocols and routers, and on intrusion detection. Section 4 describes our system model and failure models for routers. Section 5 presents our overall approach for diagnosing routers and the desirable properties of diagnosis protocols. Sections 6 and 7 detail our techniques and protocols for misbehaving-router detection and present how automated response can be carried out to logically remove those routers, thus restoring the operational status of the networks. Section 8 concludes the paper and discusses the limitations of our work.

2 Examples of Routing Infrastructure Failures

In this section, we describe three denial of service examples related to routers and routing protocols. They are the 1980 ARPANET collapse, "black hole" routers, and routers that misroute packets.

In the 1980 ARPANET collapse [20], the source of the problem was mainly due to a faulty router which generated a sequence of bad control packets. The sequence numbers of these control packets were ordered such that one was "fresher" than another and thus formed a cycle. Because those control packets received a higher priority than the data packets, the routers in the ARPANET spent most of their time handling these routing updates. Thus the network was unavailable for hours. This particular problem was fixed, but other similar problems might still exist. Finn's comments [5] on the ARPANET incident are as follows:

> "It is clear that many such update sequences can be found. This occurred entirely by accident, from an unlikely set of circumstances. Network designers did not consider it a serious possibility. However,

a malicious router could easily create this situation and halt the network. Such an attack would be extremely damaging, difficult to prevent, and difficult to correct once it occurred."

Routers exchange control packets to reflect changes, such as topology changes, in a network. A black hole router (e.g., [5]) sends out routing updates claiming that it is on zero-cost (or low-cost) paths to all destinations and then proceeds to drop the packets that it receives. In shortest-path-based routing protocols, the most common kind of routing protocols, routers in the neighborhood of a black hole router will direct (some of) their network traffic to the black hole. Figure 1 depicts a black hole router. The black hole problem has occurred in operational networks and can cause a widespread denial of service.

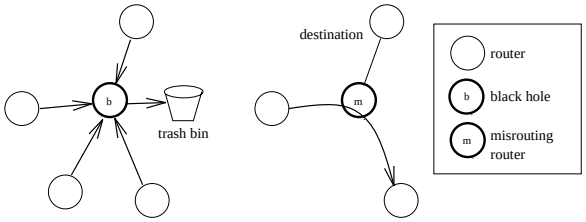

Figure 1: Black Hole Routers and Misrouting Routers.

Ideally, routers cooperate with each other to deliver the packets to their destinations. However, if the routers make their routing decisions based on different views of the state of the network, routing loops may be formed and the packets caught in them may never reach their destinations. Temporary routing loops occur naturally, say when a link goes down, and solutions have been proposed to deal with them (e.g., [3]). Permanent routing loops or misrouting by a malicious router, depicted in Figure 1, are more serious problems. In an IP network, packets have a time-to-live (TTL) field, which guarantees a packet will not stay in the network forever. Hence routing loops and misrouting can cause packets to be dropped.

The first example belongs to a family of problems in which routers receive a lot of high priority control packets originating from a router, and the routers spend a significant portion of their time processing those packets. This type of problems is easy to detect and excessive control packets can usually be dropped. The second example (i.e., black hole routers) and the third example (i.e., misrouting routers) are difficult to counter. We will model these two types of failures and present diagnosis protocols to detect and to respond to them.

3 Related Work

Many existing routing protocols are not very secure. For example, sending plain-text passwords in the clear is the only authentication method currently defined to protect routing update packets in RIP version 2 [10]. For OSPF version 2 [14], the OSPF standard defines a cryptographic authentication scheme in addition to a simple plain-text password scheme. However, in that cryptographic authentication scheme, routers on a network/subnet use a secret shared key to authenticate routing protocol packets. Thus the cryptographic authentication scheme does not offer adequate protection against some misbehaving routers. Perlman [17, 18] presents a scheme for public-key distribution and for protecting link state updates by means of digital signatures. Finn [5] discusses using public-key and secret-key routing update authentication in general and proposes a secure routing protocol. Kumar and Crowcroft [8] propose a design to secure IDPR, an inter-domain routing protocol. Murphy and Badger [16] propose a design to incorporate public-key distribution and signing link state updates in OSPF. Using strong authentication methods on routing information does not solve all the problems. If a router is faulty or compromised, it may send out erroneous but authentic routing control packets. Thus when we remotely download router software to routers or configure routers, we need to use secure remote access protocols. Finn's report [5] is a good source of background information on the vulnerabilities of computer networks.

Intrusion detection (e.g., [4, 6, 9, 15]) is a retrofit approach to improve the security of computer systems and networks. Intrusion detection systems detect and possibly respond to policy violations. A fundamental assumption of intrusion detection is that we have to live with existing systems and network infrastructures. Thus changes to them should be kept at a minimum when we improve their security. It is impractical to assume that we will replace

the existing (insecure) computer and network systems by secure systems in the near future because of the huge costs and the difficulties in building a useful yet perfectly secure system. Designing and deploying secure systems and protocols are important; we should do everything we can to prevent accidents and attacks. We view intrusion detection as the second line of defense. To the best of our knowledge, no work has been published on intrusion detection in routing infrastructures. Moreover, very little intrusion detection work has been done on detecting denial of service attacks [1].

4 Our Model

A network is modeled by a directed graph $G = (V, E)$. Vertices represent routers and edges represent communication channels, which may be point-to-point links or networks attached to more than one router. Note that we do not model hosts that are not routers. If a source host cannot send a packet directly to the destination host, the source will send the packet to a router. We call this router the *source router*. When a router receives a packet, it will send the packet directly to the destination host if it can; otherwise, it will forward the packet to another router "closer" to the destination host. We call the router that delivers the packet to the destination host the *destination router*. A packet generated by a host is represented by a packet generated by the source router. That packet is called a *source packet* with respect to the source router. Moreover, a packet destined for a host is represented by a packet destined for the destination router. That packet is called a *destination packet* with respect to the destination router. Packets processed by a router that are neither source packets nor destination packets of the router are called *transit packets* with respect to that router.

We make the following network assumptions. Assumptions 1, 2, and 3 are used to ensure that a testing router knows the expected behavior of the tested routers. Note that Assumptions 1 and 2 can be realized by using link-state routing[1]. We will justify Assumption 1 in Section 8.

Assumption 1 (Shared Views on Network States) *Neighboring routers share the same map that shows how routers are connected and the cost of the communication links.*

Assumption 2 (Shortest-path Routing) *A router always chooses the shortest path to route a packet to its destination.*

Assumption 3 (Bidirectional Channels) $\forall i, j \in V$, $(i, j) \in E \Rightarrow (j, i) \in E$. *In other words, neighboring routers can send packets directly to each other.*

Traditionally, security research works on the worst-case assumption that an adversary has unlimited power. Solutions developed under that assumption, if they exist at all, may be impractical to use [12]. In reality, some failures may be less likely to occur than others. For example, many router failures are caused by accidents. Another example is that an attacker may be able to change the routing table of a router but not the router software. Modifying the router software may require detailed knowledge about the routing protocol and the router's operating system and access to the (possibly proprietary) source code. In the rest of this section, we present failure models for routers by characterizing the behaviors of an adversary. Enumerating failure models allows us to study the problems and develop solutions for them.

We define a *network sink* as a router that drops (some of) its transit packets. A *black hole* is a network sink that also sends out routing advertisements claiming it can reach certain destinations with costs lower than that it should advertise according to the routing protocol specification. We define a *misrouting router* as a router that forwards a transit packet to a router other than the one on the shortest path to the destination router. A router that exhibits network sink or misrouting behavior is called a *bad router*; otherwise, it is called a *good router*. Bad routers may be caused by software/hardware faults, misconfiguration, or malicious attacks. We make the following assumption about good routers.

[1] In link-state routing, a router periodically computes the cost (e.g., delay) to each of its neighboring routers and generates an update packet that contains its own identity and the costs to neighboring routers. The update packet is then distributed to all other routers by flooding. Each router collects the update packets from all other routers, constructs the shortest path tree with itself as the root, and updates its own routing table. Examples of link-state routing protocols are Open Shortest Path First (OSPF)[14], IS-IS[7], a proprietary protocol used in the Internet core system known as SPREAD, and a proprietary routing protocol used in the ARPANET[11].

Assumption 4 (Existence and Connectivity of Good Routers) *There exists at least one good router in the network. Good routers are connected via good routers. In other words, bad routers do not partition the network.*

We present two independent ways to classify bad routers with the property that a stronger class is a subset of a weaker class. Thus any solution for a weaker class is also applicable to a stronger class. Note that a bad router from a weaker class is usually harder to detect than one from a stronger class. Our classifications are depicted in Figure 2. The first

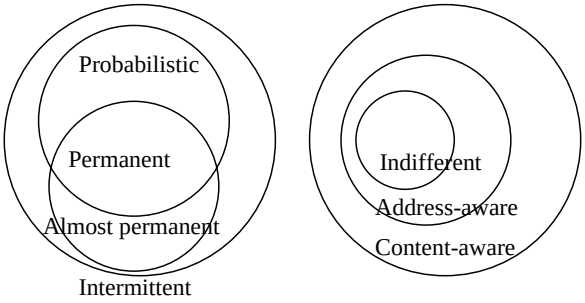

Figure 2: Classifications of Bad Routers.

classification concerns the *timing* when bad routers misbehave, and the second classification addresses on *what packets* bad routers misbehave. In the first classification, the classes are *permanent*, *almost permanent*, *probabilistic* and *intermittent*. A permanently bad router exhibits the anomalous (i.e., network sink or misrouting) behavior all the time. An almost permanently bad router is like a permanently bad router, except when it sees explicit control packets that correspond to diagnosis which will reveal its anomalous behavior, it may behave like a good router. After the diagnosis is over, it may switch back to the bad-router mode. The purpose for the almost permanent class is to model how a bad router can "trick" a diagnosing router. For every transit packet, a probabilistic bad router exhibits the anomalous behavior with a certain probability. An intermittently bad router may not exhibit the anomalous behavior consistently but misbehaves infinitely often[2]. "Permanent" is weaker than both "almost permanent" and "probabilistic". "Almost permanent" and "probabilistic" are in turn weaker than "Intermittent".

In the second classification, the classes, from the strongest to the weakest, use the following criteria to drop or to misroute packets: all packets, the values of source or destination attributes[3] that satisfy certain conditions, and the contents of entire packets, which include the values of the source/destination attributes and the packet payload, that satisfy certain conditions. Those classes are called *indifferent*, *address-aware*, and *content-aware* respectively. In our definition, the "address-aware" class includes the "indifferent" class; an indifferent bad router is a special case of address-aware bad routers in that it does not use the address information. Similarly, an address-aware bad router is a special case of content-aware bad routers. For example, an address-aware bad router may act on packets sent by a certain organization and a content-aware bad router may act on packets that contain certain keywords in their payload.

5 Our Approach

In our approach, routers diagnose each other to identify the bad routers. Preparata, Metze, and Chien (PMC) [19] proposed a framework for this kind of diagnosis. (Barborak, et al.'s paper [2] surveys work that [19] has initiated.) Preparata, et al. modeled a system equipped with automatic fault diagnosis in which system components can test each other to detect and to locate faulty components. After a component applies a test to another component, the tester will know if the tested component is fault-free or faulty. Permanent faults and perfect test coverage[4] are assumed. In the PMC model, a centralized supervisor is used to collect and analyze all the test results and determine which components are faulty. Note that the test results from a faulty component may be unreliable. The PMC model is

[2] Routers that misbehave only once or a small number of times are usually not very harmful. In fact, virtually all existing routing protocols support best-effort delivery only; there is no guarantee that all packets can reach their destinations. Bad routers that misbehave only once or a finite number of times may still be detected by a good router using the protocol presented in Section 7; however, we cannot guarantee that the protocol can completely disconnect a bad router in this case because that may require multiple rounds of diagnosis and reconfiguration.

[3] For IP networks, the attributes for a source or a destination are an IP address and a port number. Standard services are usually associated with well-known port numbers.

[4] A fault-free tester can always determine accurately the state of a tested component.

a starting point for our work, but we need a more realistic model in the context of routing infrastructures.

There are two main issues in our approach. First, given a failure model, we need to design tests that can reveal the anomalous behaviors of bad routers. Second, we need to determine how to carry out the diagnosis. In the PMC model, test assignments are designed assuming that a component can test any other component directly. However, we need to consider the topology of the underlying physical communication network in router diagnosis—how routers can communicate/coordinate with or test each other without being affected by the presence of bad routers. Sections 6 and 7 present two different techniques for detecting network sinks and misrouting routers, namely distributed probing and flow analysis, and discuss how to perform automated response to reconfigure the network so as to logically remove the bad routers. *Distributed probing* assumes a more benign bad router model and has a lower cost. Moreover, it works well even if multiple bad routers exist simultaneously. *Flow analysis* works on a more malicious bad router model; however, it is more expensive because it requires the routers to monitor every transit packet. Our diagnosis protocols are distributed in nature and do not assume a centralized analyzer that gathers and analyses the test results, which may become a single point of failure.

Assuming that testing routers can determine the expected behavior of tested routers and that well-behaved routers do not drop or misroute transit packets, we use the following criteria—the first two concern detection and the third concerns response—to evaluate our diagnosis protocols:

- **Soundness:** If a router is diagnosed as a bad router by good routers, the router is a bad router.
- **Completeness:** If there are bad routers in the network that have misbehaved, our diagnosis routine can locate at least one of them at a time.
- **Responsiveness:** Eventually, all bad routers in the network will be identified and logically removed, and the good routers will still be connected.

6 Distributed Probing

In distributed probing, a router diagnoses its neighboring routers by sending them directly (i.e., without passing through intermediate routers) a test packet whose destination router is the tester itself. Based on whether a tester can get back the test packet within a certain time interval[5], the tester can deduce the goodness of the tested router. Note that this test is not applicable for all neighboring router pairs, but we will show that there are enough of them to meet the soundness, the completeness, and the responsiveness criteria. If the shortest path from the tested router to the tester involves other intermediate routers, the fact that the test packet cannot reach the tester does not necessarily mean the tested router is bad.

Distributed probing is applicable to detecting network sinks and misrouting routers that cause denial of service—that is, the misrouted packets cannot reach their destinations. In this section, we will present two protocols that detect two different classes of bad routers. The first protocol works for almost permanent, indifferent bad routers. The second protocol works for almost permanent bad routers that are source-address-aware and payload-aware. Before we present our protocols, we will define our notation and state additional assumptions that are specific to distributed probing.

Recall that a network is modeled by a directed graph $G = (V, E)$ where vertices denote routers and edges denote communication channels. Let $e = (i, j) \in E$ be an edge that goes from vertex i to vertex j. The cost of e is denoted by $cost(i, j)$ or $cost(e)$. Note that our definition allows $cost(i, j) \neq cost(j, i)$ to model asymmetric cost metrices. An edge $(i, j) \in E$ is called *testable* if $cost(j, i)$ is strictly less than the cost of any other path from j to i in G, where the cost of a path is the sum of the costs of all edges on the path. We use Assumption 1 to ensure that i and j see the same network state. The notion of testable edges characterizes the edges useful to distributed probing. Consider a network, depicted in Figure 3, that has three routers, namely a, b, and c. We denote $cost(b, c)$, $cost(b, a)$, and $cost(a, c)$ by c_1, c_2, and c_3 respectively. The edge (c, b) is testable if $c_1 < (c_2 + c_3)$. If (c, b) is testable and router c sends a packet p whose destination is c itself to b, then

[5] The time interval is set as an upper bound of the round-trip time between the testing router and the tested neighbor.

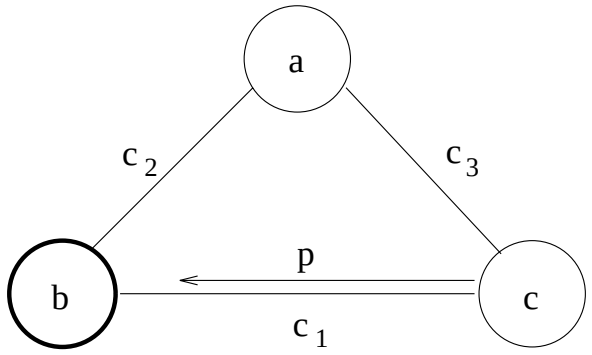

Figure 3: Testable Edges.

p will return to c if and only if b does not misbehave on p. For $i \in V$, $N(i) = \{j \mid (i,j) \in E\}$ denotes the set of neighbors of vertex i. For $S \subset V$, $N(S) = \{j \notin S \mid (i,j) \in E \wedge i \in S\}$ denotes the set of neighbors of S, a set of vertices. If the context is clear, we sometimes use i, $i \in V$, to refer to the router represented by vertex i.

Assumption 5 (Positive-Cost Edges) $\forall e \in E$, $cost(e) > 0$.

Assumption 6 (Pairwise Private Addresses) *For all $i \in V$ and $j \in N(i)$, i has an address that i can, but j cannot, reach without using any intermediate routers*[6]*. We call this address the pairwise private address of vertex i with respect to vertex j and denote it by $paddr_j(i)$. This requirement ensures that a testing router can generate a packet whose destination is the testing router itself and the packet is a transit packet for the tested router.*

Protocol 1 (Autonomous Distributed Probing)
$\forall i \in V$, vertex i executes the following at random times[7]*:*

For each $j \in N(i)$ *such that* (i,j) *is testable*
Send a packet whose destination is $paddr_j(i)$, say p, to j via (i,j);
If p *does not return to* i
Then i *ceases its neighbor relationship with* j[8] *(i.e., i thinks j is bad)*
Else i *does nothing (i.e., i thinks j is good)*

Lemma 1 *Given that bad routers are almost permanent and indifferent, Protocol 1 is sound.*

Proof: Because Protocol 1 does not use any control packets, it is applicable to diagnosing almost permanently bad routers. The soundness of the protocol follows from the definition of testable edges. □

Lemma 2 *Given that bad routers are almost permanent and indifferent, Protocol 1 is complete.*

Proof: Consider a maximal connected component of bad routers in G. We denote the set of those bad routers by B. If $N(B)$ is empty, then by Assumptions 3 and 4 the bad routers are disconnected from the network. Otherwise, we claim that at least one vertex in $N(B)$, the set of good neighbors of B, has a testable edge to a vertex in B. On the contrary, we assume that none of the vertices in $N(B)$ has a testable edge to a vertex in B. Let $BN = \{(x,y) \mid x \in B \wedge y \in N(B)\}$, the set of edges incident to a vertex in B and a vertex in $N(B)$. Moreover, let $(b,n) \in BN$ be an edge such that $\forall e \in BN, cost(b,n) \leq cost(e)$. Because we assume (n,b) is not testable, there exists a multi-edge path $P = (b \rightarrow ... \rightarrow n)$ such that $cost(P) \leq cost(b,n)$. Thus, by Assumption 5, $\exists e \in BN$ such that $cost(b,n) > cost(e)$, which contradicts the choice of (b,n). □

Lemma 3 *Given that bad routers are almost permanent and indifferent, Protocol 1 is responsive.*

Proof: Lemma 2 proves that at least one of the bad routers, say b, will be located by a good router, say g. Then, by Protocol 1, g will cease its neighborhood relationship with b. Recall that we assume the good routers are connected in G. The new graph $G' = (V, E')$ where $E' = E - \{(b,g),(g,b)\}$ has all the good routers remaining connected. Note that bad routers disconnecting themselves from their neighbors, no matter good or bad, does not affect the result. Thus running Protocol 1 continuously will eventually remove all the edges between a good router and a bad router, yet maintaining good routers connected. □

[6] If necessary, we may assign an unused address to a router interface to realize this requirement.

[7] The protocol is executed at random times to assure a tester does not reveal its testing mode.

[8] Broadcasting neighbor relationship changes can be done by flooding, the procedure used by link state protocols to distribute routing updates.

Theorem 1 *Given that bad routers are almost permanent and indifferent, Protocol 1 is sound, complete, and responsive.*

Proof: The proof follows from Lemma 1, Lemma 2, and Lemma 3. □

Protocol 1 can be modified to cope with permanently bad routers that are source-address-aware and payload-aware. A fresh and authenticated[9] diagnosis request that contains the values of source attributes and payload can be distributed to all routers by flooding. Then the routers will use those values to construct their test packets. However, the request may alert the bad routers about the upcoming diagnosis. Thus the flooding of that diagnosis request disqualifies the protocol for detecting almost permanently bad routers. In the following, we present another modification to Protocol 1, which we call source-initiated distributed probing, that (almost) avoids the problem of alerting bad routers about the diagnosis.

In *source-initiated distributed probing*, a fresh and authenticated start diagnosis request that contains an identifier, *id*, and the values of source attributes and payload is sent to a router, say r. The protocol assumes that r is a good router. For example, we can choose the source router with respect to the values of source attributes as r. If a host finds out its packets cannot reach their destinations, it can send a request that contains the information about those packets, which can be used to construct a diagnosis packet, to the source router r. Note that in Protocol 2, a router forwards a start diagnosis request to a neighbor only after that neighbor is diagnosed to be a good router. Thus routers will not be alerted about the diagnosis before they are judged to be good routers unless r is a bad router. To stop the diagnosis, an authenticated quit diagnosis request that contains *id* will be sent to all routers.

Protocol 2 (Source-initiated Distributed Probing)
$\forall i \in V$, if i receives a fresh and authenticated start request, sr, that contains the values for source attributes, s, and the payload, l, then i executes the following at random times:

If *i receives the authenticated quit request qr*
Then *i forwards qr to all neighbors to which i has sent the corresponding sr;*
i quits the diagnosis;
For each *$j \in N(i)$ such that (i,j) is testable*
Send a packet p whose source is s, destination is $paddr_j(i)$, and payload is l, to j via (i,j);
If *p does not return to i*
Then *i ceases its neighbor relationship with j (i.e., i thinks j is bad)*
ElseIf *i has not forwarded sr to j, do so (i.e., i thinks j is good)*

Lemma 4 *Given that bad routers are almost permanent and source-address- and payload-aware, and the first router chosen to initiate the diagnosis is good, Protocol 2 is complete.*

Proof: Let r be the router chosen to initiate the diagnosis and KG be the set of known "good" routers. KG is initialized to $\{r\}$. We will prove the completeness of Protocol 2 by induction on the cardinality of KG.

Base case (i.e., $KG = \{r\}$): Let $MIN = \{i \in N(r) \mid \forall j \in N(r), cost(j,r) \geq cost(i,r)\}$. We claim that r has a testable edge to every vertex in MIN; otherwise, by Assumption 5, it violates the definition of MIN. If all vertices in MIN are good, then the new KG equals the old $KG \cup MIN$ (i.e., the cardinality of KG is increased by at least one). Otherwise, a bad router is located.

Induction step: Consider an arbitrary vertex $g_1 \in KG$. Let $x_1 \in N(g_1) - KG$. If (g_1, x_1) is testable, then either the new KG equals the old $KG \cup \{x_1\}$ or x_1 is diagnosed as a bad router. If (g_1, x_1) is not testable, then $\exists g_2 \in KG \wedge x_2 \in N(g_2) - KG$ such that $cost(x_1 \rightarrow ...x_2 \rightarrow g_2 ... \rightarrow g_1) \leq cost(x_1, g_1)$. In other words, we have $cost(x_2, g_2) < cost(x_1, g_1)$. Again, if (g_2, x_2) is not testable, then we can apply the same argument and eventually we can find a testable link originating from a vertex in KG. As a result, either the cardinality of KG can be increased by at least one, or a bad router can be located. □

Theorem 2 *Given that bad routers are almost permanent and source-address- and payload-aware, and the first router chosen to initiate the diagnosis is good, Protocol 2 is sound, complete, and responsive.*

Proof: The proof follows from Lemma 4 and the fact that the proofs of the soundness and the responsiveness properties are the same as those of Protocol 1. □

[9] The message authentication requirements for our diagnosis protocols are the same as those for link state routing protocols. Thus one may use the digital signature schemes proposed by Perlman[17, 18] and Murphy and Badger[16] to authenticate the diagnosis control messages.

7 Flow Analysis

Flow analysis monitors the transit packets flowing in and out of a router to detect abnormal behaviors. For each router, the neighbors collaborate with each other to diagnose the router. To enable robust communication among the neighbors, flooding, a technique first proposed by Perlman [17, 18], is used. To detect network sinks, the neighbors verify "conservation of transit traffic", depicted in Figure 4, by

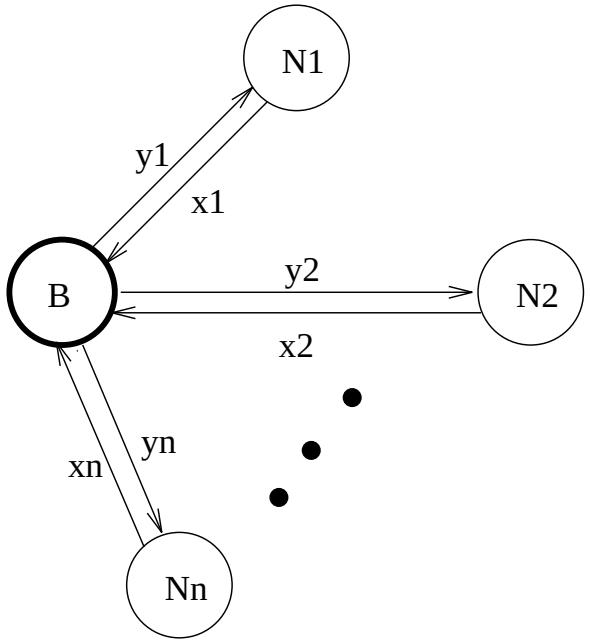

Figure 4: Conservation of Transit Traffic: $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$.

comparing the amount of transit traffic with respect to the tested router going in and that going out of the router. To detect misrouting routers, they verify that the transit packets coming out of the tested router are correctly forwarded. Flow analysis is applicable to bad routers that are intermittent and content-aware—that is, all failure models discussed in Section 4. In this section, we will first define our notation, and then state additional assumptions that are specific to flow analysis. Finally, we will present our diagnosis protocol and prove its properties.

For all $(i,j) \in E$ and $k \in \{i,j\}$, let $t_{(i,j)}(k)$ be the accumulated number of bytes of the packet payload[10] for the transit packets with respect to both i and j sent from i to j from k's point of view[11], $n_{(i,j)}(k)$ be the accumulated number of bytes of the packet payload for the packets that are transit to i but non-transit to j sent from i to j from k's point of view, $g_{(i,j)}(k)$ be the accumulated number of bytes of the packet payload for the packets that are source packets of i and transit to j sent from i to j from k's point of view, and $m_{(i,j)}(j)$ be the accumulated number of bytes of the packet payload for the misrouted transit packets with respect to i sent from i to j from j's point of view. Figure 5 depicts $t_{(i,j)}(k)$, $n_{(i,j)}(k)$, and $g_{(i,j)}(k)$, which concern packets sent from router i to router j. A router can compute the above counters because of the assumption that routers know the topology of the network and the costs of the edges (i.e., Assumption 1).

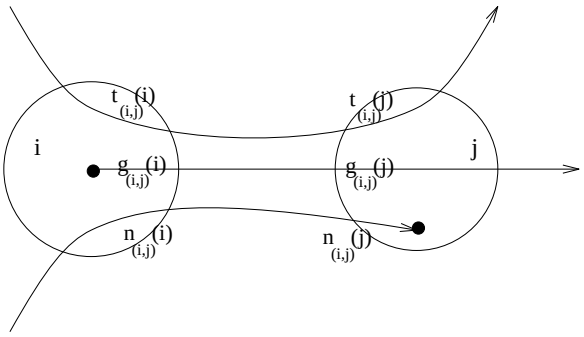

Figure 5: $t_{(i,j)}(k)$, $n_{(i,j)}(k)$, and $g_{(i,j)}(k)$, where $k \in \{i,j\}$.

Assumption 7 (No Adjacent Bad Routers) *$\forall (i,j) \in E$, i is a good router or j is a good router.*

Assumption 8 (Good Routers are in the Majority) *The number of good routers $> |V|/2$.*

Assumption 9 (Per-hop Packet Delay is Negligible) *The propagation delay, the processing delay, and the queueing delay per hop are negligible.*

[10] Because of possible packet fragmentation, we use packet payload sizes instead of packets counts. Packet fragmentation occurs because networks have different maximum packet sizes, also known as maximum transfer units (MTU).

[11] We introduce k here to detect disagreements between i and j.

The execution of Protocol 3 is divided into phases. In simple terms, routers checkpoint their counters at the same time[12] and then broadcast their values via flooding to other routers. Assumption 9 ensures that conservation of transit packets holds[13]. Based on the checkpoint messages received, routers can decide if a router exhibits anomalous behaviors—not sending out checkpoint messages, sending out bogus checkpoint messages, removing transit packets, or misrouting packets. Protocol 3 would be simpler if we assume that the routers' clocks are synchronized so that all good routers can depend on their clocks to advance to the next phase. We do not make that assumption. In Protocol 3, a router waits until it receives a "phase change" message (which we call a *next phase ready* message) from $\lceil(|V|+1)/2\rceil$ routers before advancing to the next phase. Because we assume the majority of the routers are good (Assumption 8), at least one good router is involved in each phase change. Bad routers cannot significantly increase or decrease the time interval between any two consecutive phases.

Protocol 3 (Flow Analysis)
$\forall i \in V$, i initializes the current phase variable, $phase_i$, to zero. Let π be the pre-defined time interval between consecutive phases and Δt_i be the local time elapsed since the last phase started. If i has just started, "last phase started" denotes the time i started running the protocol. A message is called current if its phase number equals $phase_i$. Then i executes the following:

Wait until *(1) $\Delta t_i = \pi$ or (2) $\lceil(|V|+1)/2\rceil$ authenticated current next phase ready messages have been received;*
Broadcast an authenticated next phase ready message that contains $phase_i$;
Wait until $\lceil(|V|+1)/2\rceil$ *authenticated current next phase ready messages have been received;*
Store and then reset local counters (i.e., $t_{(i,j)}(i), t_{(j,i)}(i), n_{(i,j)}(i), n_{(j,i)}(i), g_{(i,j)}(i), g_{(j,i)}(i)$, and $m_{(j,i)}(i)$);
Set $\Delta t_i = 0$;
Broadcast an authenticated checkpoint message to all routers that contains (1) $phase_i$, (2) $\forall j \in N(i)$, $t_{(i,j)}(i)$, $n_{(i,j)}(i)$, and $g_{(i,j)}(i)$, and (3) $\forall k \in V$ such that $i \in N(k)$, $t_{(k,i)}(i)$, $n_{(k,i)}(i)$, and $g_{(k,i)}(i)$;
For each $j \in N(i)$
  If *j's authenticated current checkpoint message has been received and $t_{(i,j)}(i) = t_{(i,j)}(j) \wedge n_{(i,j)}(i) = n_{(i,j)}(j) \wedge g_{(i,j)}(i) = g_{(i,j)}(j)$*
  Then
    If *$\forall k \in V$ such that $k \in N(j)$ or $j \in N(k)$, k's authenticated current checkpoint message has been received and $(k \in N(j) \Rightarrow t_{(j,k)}(j) = t_{(j,k)}(k) \wedge n_{(j,k)}(j) = n_{(j,k)}(k) \wedge g_{(j,k)}(j) = g_{(j,k)}(k))$ and $(j \in N(k) \Rightarrow t_{(k,j)}(j) = t_{(k,j)}(k) \wedge n_{(k,j)}(j) = n_{(k,j)}(k) \wedge g_{(k,j)}(j) = g_{(k,j)}(k))$*
    Then
      If *$\sum_{l \in \{n \mid j \in N(n)\}}(t_{(l,j)}(j) + g_{(l,j)}(j)) \neq \sum_{l \in N(j)}(t_{(j,l)}(j) + n_{(j,l)}(j))$ (i.e., conservation of transit traffic violated)*
      Then *i ceases its neighbor relationship with j;*
      Else *do nothing (because other routers will respond to the problem);*
  Else *i ceases its neighbor relationship with j;*
For each $j \in \{n \mid i \in N(n)\}$
  If *j's authenticated current checkpoint message has not been received or $t_{(j,i)}(i) \neq t_{(j,i)}(j) \vee n_{(j,i)}(i) \neq n_{(j,i)}(j) \vee g_{(j,i)}(i) \neq g_{(j,i)}(j) \vee m_{(j,i)}(i) \neq 0$*
  Then *i ceases its neighbor relationship with j;*
Set $phase_i = phase_i + 1$

Lemma 5 *Given that bad routers are intermittent and content-aware, Protocol 3 is sound.*

Proof: We prove the lemma by case analysis. First, there are three ways a router, say b, can be diagnosed as a network sink by good routers:

- $\exists i \in V$, a good router, such that $i \in N(b)$ or $b \in N(i)$ and i does not receive authenticated current checkpoint messages from b. Because

[12] In practice, we only require a router and its neighbors to checkpoint at approximately the same time. Because communication channels are bidirectional (Assumption 3) and flooding is used, neighboring routers see other routers' "phase change" messages at roughly the same time. A bad router could delay forwarding packets to its neighbors; however, it can only cast suspicion on itself for not having a consistent view with its neighbors.

[13] We can realize negligible per-hop packet delay by choosing an appropriate π, time interval between consecutive phases.

we assume all good routers are connected, and flooding, which takes negligible time, is used to broadcast checkpoint data, i's not receiving b's checkpoint data implies that b has not sent any.

- $\exists i \in V$, a good router, such that $(i \in N(b) \wedge (t_{(b,i)}(i) \neq t_{(b,i)}(b) \vee n_{(b,i)}(i) \neq n_{(b,i)}(b) \vee g_{(b,i)}(i) \neq g_{(b,i)}(b)))$ or $(b \in N(i) \wedge (t_{(i,b)}(i) \neq t_{(i,b)}(b) \vee n_{(i,b)}(i) \neq n_{(i,b)}(b) \vee g_{(i,b)}(i) \neq g_{(i,b)}(b)))$ implies either b or i has lied. Thus b must be a bad router.

- $\sum_{k \in \{n \mid b \in N(n)\}} (t_{(k,b)}(b) + g_{(k,b)}(b)) \neq \sum_{k \in N(b)} (t_{(b,k)}(b) + n_{(b,k)}(b))$ implies b is a network sink because the amount of transit traffic flowing in b is not equal to that flowing out of b.

Note that b is diagnosed by $i \in N(b)$ as a misrouting router only when $m_{(b,i)}(i) \neq 0$. Hence Protocol 3 is sound. □

Lemma 6 *Given that bad routers are intermittent and content-aware, Protocol 3 is complete.*

Proof: By performing a case analysis similar to that of Lemma 5 together with Assumption 7, one can show that bad routers which have misbehaved and are connected to the network will be located by at least one of their good neighbors in the next phase. □

Lemma 7 *Given that bad routers are intermittent and content-aware, Protocol 3 is responsive.*

Proof: By Assumption 7 and Lemma 5, we know that only edges incident on a good router and a bad router are removed from E. Thus good routers will remain connected. By Lemma 6, when a bad router that is connected to the network misbehaves, it will be located by a good router in the next phase. Together with the fact that an intermittently bad router misbehaves infinitely often, eventually all bad routers will be logically removed from the network. □

Theorem 3 *Given that bad routers are intermittent and content-aware, Protocol 3 is sound, complete, and responsive.*

Proof: The proof follows from Lemma 5, Lemma 6, and Lemma 7. □

8 Discussion

This paper addresses denial of service on routers and routing protocols. We present failure models for routers that characterize the behavior of failed routers, which may be due to natural faults or malicious attacks. Based on the failure models, we develop techniques and protocols to detect and to logically remove misbehaving routers from the network, and prove properties of the protocols, namely soundness, completeness, and responsiveness. Our diagnosis protocols are designed to avoid introducing additional vulnerabilities to the routing infrastructures through the use of them—a bad router cannot disconnect a good router from the rest of the network and a bad router cannot initiate the diagnosis too often to make all routers spend most of their time executing the protocol (c.f. the flow analysis protocol in Section 7). In conclusion, if there is a path between the source and the destination on which all routers are good, our protocols guarantee that the network will eventually be able to deliver packets from the source to the destination.

Our work does not solve the entire denial of service problem of routing infrastructures. This paper represents a first step to protect routing infrastructures from denial of service using an intrusion detection approach. Issues not addressed include the following:

- There are router failures not covered by our failure models (c.f. the motivation of the failure models in Section 4). For example, a compromised router may modify the body of a transit packet. Distributed probing can be adapted to handle this problem—a router can check the integrity of test packets after they are sent back by tested routers. However, adapting flow analysis to diagnose this kind of failures appears to be non-trivial and is a future work item. Another example is that a failed router may generate spurious packets to overwhelm links or other routers. One may impose a limit on the amount of source packets that can be generated by a router per unit time and have its neighboring routers to verify it. Moreover, the flow analysis technique can be used to cope with replication of transit traffic.

- Link failures are not modeled. Note that in our protocols, a link failure that results in packet loss may be viewed as a node failure. The

routers incident on the link will detect the failure and cease the neighbor relationship. Consequently, the failed link will not be used.

- Our definition of bad routers may include behaviors of legitimate routers, which may lose packets due to congestion. We may restrict our protocols to be used in lightly loaded networks or for network connections with reserved bandwidth. To apply our protocols for a more general setting, we may extend them by using a threshold on how many packets a good router can drop. To illustrate, we may incorporate the "k-out-of-n" method in our distributed probing protocols. A router is considered good if it can pass k out of n tests. A related future work is to address inter-domain issues such as policy routing (in which routers may not use shortest paths to route packets) and firewalls (e.g., packet filters).

- We assume that neighboring routers see the same network state so that testable edges in the distributed probing protocols can be determined and consistent checkpointing in the flow analysis protocol can be performed. We argue that is a reasonable assumption. First, by using flooding to disseminate routing updates (as is done in link-state routing protocols) and checkpoint packets and requiring communication channels to be bidirectional, neighboring routers see the control packets at almost the same time. Although a bad router could delay forwarding those control packets, it only hurts that bad router itself for not having a consistent view among that router and its neighbors. Second, as noted in [13], link costs are static, independent of link load, in modern link-state routing protocols. Thus normally link states do not change often. To cope with the few cases in which our assumption does not hold, again the threshold technique can be used. For the checkpointing problem, we can also use a longer time interval between consecutive phases to reduce the impact of slightly different checkpoint times among neighboring routers. Future work is needed to validate the practicality of the assumption.

- Our models only consider transit traffic. In other words, packets sent by source hosts to source routers and those sent by destination routers to destination hosts are not addressed. Our work is useful in containing the damage that can be caused by a router to its source and destination packets. A related issue is that a router may claim that it is directly connected to a local network, thus becoming a source/destination router for that local network. To address this problem, routers can be given a list of potential neighbors and use it to identify those false advertisements.

- We have not examined the diagnosis overhead on routers in detail, but the overhead does not seem to be excessive. For distributed probing, a router needs to determine the testable links from itself to its neigbhoring routers, based on the link state updates received, and then sends (and receives) a test packet to (and from), in the worst case, each of its neighbors. The overhead depends on how often the diagnosis is performed. For flow analysis, there are two main sources of overhead: First, for each transit packet, the router needs to lookup a table, which may need to be updated when there is a topology change, and then increments the appropriate counter. Second, at the end of each phase, routers broadcast an authenticated packet containing the values of their local counters and an authenticated packet to signal that it is ready to advance to the next phase. Then routers will verify the goodness of their neighboring routers by computing the amount of transit traffic flowing in and out of those neighbors.

Acknowledgments

This research is supported by DARPA under grant ARMY/DAAH 04-96-1-0207. We would like to thank Che-lin Ho, Frank Jou, and our colleagues in the Computer Security Laboratory at UC Davis for helpful discussions. We thank Kirk Bradley and the workshop participants for their useful comments on this paper.

References

[1] The Fourth Workshop on Computer Misuse and Anomaly Detection (CMAD IV), Monterey, California, November 12-14, 1996.

[2] M. Barborak, M. Malek, and A. Dahbura, "The Consensus Problem in Fault-Tolerant Computing." *ACM Computing Surveys*, Vol.25, No.2, June 1993, pp.171-220.

[3] D. Comer, "Internetworking with TCP/IP." Vol.1, Prentice Hall, 1991.

[4] D. Denning, "An Intrusion-Detection Model." *IEEE Transactions on Software Engineering*, Vol.SE-13, No.2, Feb. 1987, pp.222-232.

[5] G.G. Finn, "Reducing the Vulnerability of Dynamic Computer Networks," ISI Research Report RR-88-201, University of Southern California, June 1988.

[6] K. Ilgun, R.A. Kemmerer, and P.A. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection Approach." *IEEE Transactions on Software Engineering*, Vol.21, No.3, March 1995, pp.181-199.

[7] Joint Technical Committee ISO/IEC JTC 1 *Information Technology*, "Intermediate System to Intermediate System Intra-domain Routeing Information Exchange Protocol for Use in Conjunction with the Connectionless-mode Network Service (ISO 8473)," ISO/IEC 10589, April 1992.

[8] B. Kumar, and J. Crowcroft, "Integrating Security in Inter-domain Routing Protocols." *Computer Communication Review*, Oct. 1993, Vol.23, No.5, pp.36-51.

[9] T.F. Lunt, "A Survey of Intrusion Detection Techniques." *Computer and Security*, June 1993, Vol.12, No.4, pp.405-418.

[10] G. Malkin, "RIP Version 2 — Carrying Additional Information." RFC 1723, November 1994.

[11] J. McQuillan, I. Richer, and E. Rosen, "The New Routing Algorithm for the ARPANET." *IEEE Transactions on Communications*, Vol.COM-28, No.5, May 1980, pp.711-719.

[12] C. Meadows, "The Need for a Failure Model for Security", *Proceedings of the* 4^{th} *Conference on Dependable Computing for Critical Applications*, San Diego, California, January 4-6, 1994, pp.383-385.

[13] J. Moy, "Link-State Routing", in M. Steenstrup, ed., *Routing in Communications Networks,* Prentice Hall, 1995, pp.135-157.

[14] J. Moy, "OSPF Version 2." RFC 2178, July 1997.

[15] B. Mukherjee, L.T. Heberlein, and K.N. Levitt, "Network Intrusion Detection." *IEEE Network*, May-June 1994, Vol.8, No.3, pp.26-41.

[16] S.L. Murphy, and M.R. Badger, "Digital Signature Protection of the OSPF Routing Protocol", *Proceedings of the Symposium on Network and Distributed System Security (SNDSS '96)*, San Diego, California, February 22-23, 1996, pp.93-102.

[17] R. Perlman, "Network Layer Protocols with Byzantine Robustness." Ph.D. dissertation, Massachusetts Institute of Technology, August 1988.

[18] R. Perlman, "Interconnections: Bridges and Routers." Addison-Wesley, 1992.

[19] F.P. Preparata, G. Metze, and R.T. Chien, "On the Connection Assignment Problem of Diagnosable Systems." *IEEE Transactions on Electronic Computers*, Vol.EC-16, No.6, Dec. 1967, pp.848-854.

[20] E. Rosen, "Vulnerabilities of Network Control Protocols: An Example." RFC 789, September 1981.