Using Conservation of Flow as a Security Mechanism in Network Protocols

John R. Hughes, Tuomas Aura, Matt Bishop
Department of Computer Science
University of California, Davis
Davis, CA 95616
hughesj@cs.ucdavis.edu, tuomas.aura@hut.fi, bishop@cs.ucdavis.edu
http://www.cs.ucdavis.edu

Abstract

The law of Conservation of Flow, which states that an input must either be absorbed or sent on as an output (possibly with modification), is an attractive tool with which to analyze network protocols for security properties. One of its uses is to detect disruptive network elements that launch Denial of Service attacks by absorbing or discarding packets. Its use requires several assumptions about the protocols being analyzed. In this paper, we examine the WATCHERS algorithm to detect misbehaving routers. We show that it uses Conservation of Flow without sufficient verification of its assumptions, and can consequently be defeated. We suggest improvements to make the use of Conservation of Flow valid.

1. Introduction

The law of Conservation of Flow states that input to a system must either be absorbed at that system or passed along to another system. Similarly, every output of that system was produced either by an input to the system or originated within the system itself. This law is very useful in analyzing flows throughout a network. It is intuitively appealing for detecting potential denial of service attacks. If a packet is discarded, Conservation of Flow is violated because a packet entered but did not leave the system. Either the system is lying, or the entity claiming the packet never left is lying.

In this paper we analyze the use of Conservation of Flow with respect to the detection of malicious routers. After first reviewing the WATCHERS protocol, we illustrate problems that arise in assuming Conservation of Flow holds by discussing several attacks that defeat the protocol. We then follow with a discussion of the problems in detail, and how to fix them, concluding with an overview of the impact each fallible assumption creates.

2. WATCHERS Overview

A *malicious router* is a router that discards or misroutes (routes sub-optimally) packets that pass through it. WATCHERS is a distributed network monitoring protocol designed to detect and isolate these malicious routers.

WATCHERS is designed to work on networks that meet the following four assumptions:

1. The routing protocol must be a link state routing protocol (the *link state condition*);
2. Every router must be directly connected to a non-malicious router (the *good neighbor condition*);
3. Each pair of non-malicious routers must be connected by a path of non-malicious routers only (the *good path condition*); and
4. There must be at least as many non-malicious routers as malicious routers (the *majority good condition*).

Additionally, the original paper [1] asserts WATCHERS is *correct* (no Good router will diagnose another Good router as Bad) if the following two conditions hold:

1. When any router sends a WATCHERS message to a neighbor, the message arrives intact with no delay (the *perfect transmission condition*); and
2. Neighboring Good routers *always* agree on the network topology (the *neighbor agreement condition*).

The WATCHERS protocol also defines malicious routers to include those that do not participate in, or give incorrect information during, the execution of the WATCHERS protocol.

This research was funded by grant NAG21251 from the National Aeronautical and Space Administration and a gift from Microsoft Corporation to the University of California at Davis. Thanks go to WATCHERS co-authors Steven Cheung and Nick Puketza for their constructive input.

The WATCHERS protocol requires each pair of directly connected routers to maintain 7 counter types. Let *X* and *Y* be adjacent routers. Define $T_{X,Y}$ as the number of packets transiting through *X* and then *Y*, $S_{X,Y}$ as the number of packets with source *X* that pass through *Y*, and $D_{X,Y}$ as the number of packets with destination *Y* that pass through *X*. Finally, let $M_{X,Y}$ be the number of times *X* misroutes a packet to *Y*. Then *X* and *Y* each keep counters $T_{X,Y}$, $S_{X,Y}$, $D_{X,Y}$, $T_{Y,X}$, $S_{Y,X}$, and $D_{Y,X}$, and *X* also keeps $M_{Y,X}$, and *Y* keeps $M_{X,Y}$.

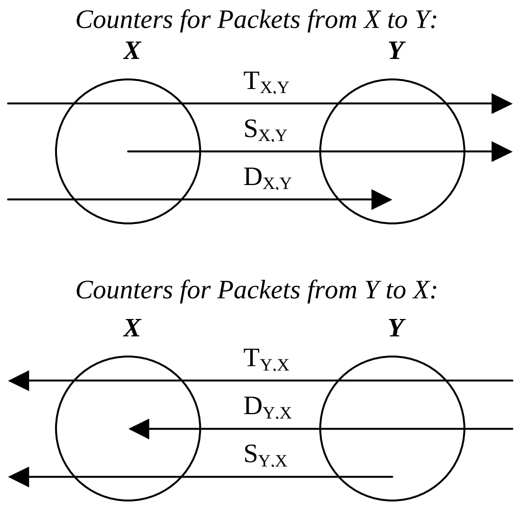

Counters for Packets from X to Y:
X
Y
T X,Y
S X,Y
D X,Y
Counters for Packets from Y to X:
X
Y
T Y.X
D Y.X
S Y.X

Figure 1: Transit packet byte counters

Periodically, each router sends a broadcast message to begin a round of WATCHERS. When a router has received a request message from a majority of routers, it floods the network with the value of its counters. When the router receives all the responses it requires, it begins the diagnosis phase.

The first part of diagnosis, *validation*, checks that the router's neighbors have counters with values that match those of the testing router. If not, the guilty routers are labeled as malicious.

Conservation-of-Flow analysis comes next. Each router performs this test on its neighbors, having received the counters from each neighbor's neighbors. The number of incoming packets minus packets destined for that router is compared to the number of outgoing packets minus packets originating with that router. If this difference exceeds some specified threshold, the tested router is diagnosed as malicious.

In order to detect groups of malicious routers that conspire to hide their misbehavior, maintenance requirements are heavily increased: Routers keep two source and two transit counters *per destination*, as opposed to per neighbor. Thus, they can perform a *consorting router* test, similar to the *Conservation-of-Flow* test, but performed separately for each destination. In this way, some neighboring routers that jointly manipulate their counters maliciously can be detected.

Regardless of which type of misbehavior is detected, the protocol directs that the malicious routers be announced within the autonomous system (AS) and be *logically removed* from the network. No longer sending messages to, or accepting messages from, detected malicious routers accomplishes this.

The WATCHERS protocol is intuitively attractive. Conservation of Flow is basic to any reliable network, and WATCHERS uses it to examine flows between neighbors and to the endpoints. Unfortunately, the Internet is not reliable, and WATCHERS makes several implicit assumptions that do not hold.

3. Attack Scenarios

Routers are the prime movers of packets throughout the Internet. Of course, a primary assumption is that the routers are trustworthy. WATCHERS attempts to verify this. We consider several attacks on routers to examine WATCHERS' strength. Unless specified otherwise, the counter disagreement and misrouting thresholds are assumed to be zero (a difference of just one missing or misrouted packet indicates a Bad router).

3.1 Packet Modification

Conservation of Flow does not speak to *which* packet proceeds to the destination. It merely ensures that *some* packet arrives at the destination. Even with the per-destination counters outlined in [1], malicious routers may continue to misroute packets undetected. Suppose router *A* in Figure 3.1 sends two packets, one to *C* and one to *X*. Bad router *B* deliberately swaps the destination addresses. Regardless of any added nodes or links to this topology, no router will be able to detect *B*'s misbehavior through the WATCHERS algorithm.

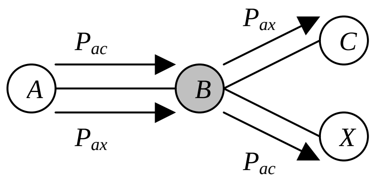


Figure 3.1: Switching packet destinations.

A	***B***	***C***	***X***
$S_{A,B}[C]=1$	$S_{A,B}[C]=1$	$D_{B,C}=1$	$D_{B,X}=1$
$S_{A,B}[X]=1$	$S_{A,B}[X]=1$		
	$D_{B,C}=1$		
	$D_{B,X}=1$		

Table 3.1: Non-zero WATCHERS counters at the conclusion of this attack.

3.2 Packet Substitution

Like packet modification, Conservation of Flow does not affect the ability to substitute packets. Per destination counters are not sufficient to detect consorting routers substituting packets. In Figure 3.2, router *A* sends a packet, P_{ax}, destined for router *X*. Bad routers *B* and *C* conspire to drop that message, replacing it with packet P_{bx}. Routers *B* and *C* then lie by incrementing their $T_{B,C}[X]$ counters (instead of their S counters). Even when an additional path of Good routers exists between *A* and *X* (satisfying the required *conditions*), as far as the WATCHERS protocol is concerned, the Good routers cannot detect this misbehavior.

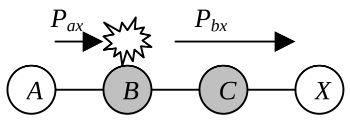


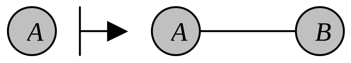
Figure 3.2: Packet Substitution

A	***B***	***C***	***X***
$S_{A,B}[X]=1$	$S_{A,B}[X]=1$	$T_{B,C}[X]=1$	$D_{C,X}=1$
	$T_{B,C}[X]=1$	$D_{C,X}=1$	

Table 3.2: Non-zero WATCHERS counters at the conclusion of this attack.

3.3 Ghost routers

Conservation of Flow says nothing about the nodes, or interior configuration of nodes, over which the flow is measured. If incoming and outgoing flows are measured, the entity between the measuring points may be one node or multiple nodes; the measurer cannot tell. So, if routers are not only able to broadcast link-state network status messages, but also topological information, the following attack becomes possible. Figure 3.3.1 depicts how Bad router *A* can announce to the network that it is really composed of two routers, *A* and *B*. The creation of such "ghost" routers allows *A* to misbehave, while shifting blame to its ghost(s). Generalizing this scenario, bad routers may invent arbitrary network topologies in place of themselves.



$$\begin{tikzcd} A \arrow[r, mapsto] & A \arrow[r, dash] & B \end{tikzcd}$$

Figure 3.3.1: Ghost router creation

As one example, a single router can mount the packet substitution attack of the previous section by pretending to be two adjacent routers. Another application of ghost routers can be seen in Figure 3.3.2. Yet undetected Bad router *A* attacks router *X* by attempting to trick *X*'s neighbors into believing *X* is Bad.

Figure 3.3.2 (a) shows router *A* sending a broadcast topology message throughout the network indicating link *A*-*E* is down (broadcast packets are not currently accounted for in the WATCHERS protocol, and thus have no effect on the WATCHERS counters in Table 3.3). For step-wise clarity, Figure 3.3.2 parts (b) and (c) depict shorter time slices in the message passing. Part (b) shows router *C* receiving *A*'s next message, claiming *A*-*E* is now up. In part (c), router *C* passes this message on, simultaneously receiving P_{xa}. Even if we assume router *E* has insisted link *A*-*E* is up, only now does router *C* believe so; *C* also concludes that router *X* has misrouted P_{xa}, the packet intended for *A* (since link *A*-*E* is now up, router *X* should be sending all traffic destined for *A* through *E*). We can thus convince one of *D*'s neighbors that *D* is bad.

To convince the other neighbor, Bad router *A* can export false topology information in such a way that it appears as though it consists of more than one router. Using these ghost routers, router *A* above can arbitrarily make the clockwise or counter-clockwise path between it and any other router the shorter of the two. Thus, router *A* can use the above attack on any router in the ring from either direction, enabling *A* to cause both neighbors of the attacked router to consider it Bad. Note that router *A* can solicit traffic from the nodes it attacks by using acknowledgement-required protocols, and with appropriate timing, increase the likelihood routers *C* and *X* exchange packets simultaneously as in Figure 3.3.2 (c).

Thus, given a sufficiently long WATCHERS round, a single compromised router can cause all other routers in a ring network to be falsely convicted as being Bad.

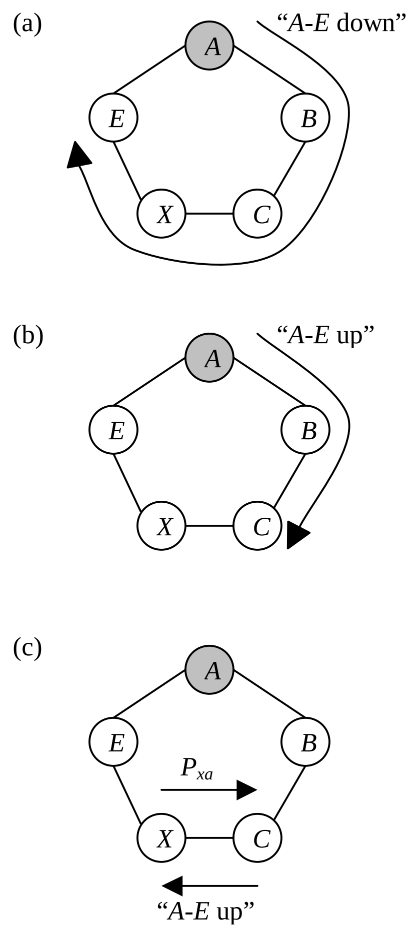


Figure 3.3.2: Topology changes used to induce packet misrouting

A	***B***	***C***	***X***	***E***
		$S_{X,C}$=1	$S_{X,C}$=1 $M_{X,C}$=1	

Table 3.3: Non-zero WATCHERS counters at the conclusion of this attack

3.4 Hot Potato

For this attack, we hypothesize that some commercial routers are more interested in processing packets quickly, than checking special and rare conditions. As an example, some venders have chosen to boost performance by not verifying IP header checksums [3].

Even when Conservation of Flow holds, entities may engage in malicious activity undetected. In a ring-network such as that in Figure 3.4, if routers are willing to believe a neighbor's claim that a shared link is down, when that message itself comes over that *same shared link*, the following attack becomes possible.

If Bad router *A* continuously broadcasts the topology update messages as shown in Figure 3.4, any packet destined for *A* may be sent back and forth between *B* and *C*, due to the "thrashing" topology. During this period, the Time to Live (TTL) of this packet may expire. Although an ICMP TTL Expired message may be sent back to the originator, WATCHERS does not view this kind of effort as compensation for the dropped packet. Thus, the router that dropped the expired packet will fail the *Conservation-of-Flow* test, causing it to be labeled as Bad.

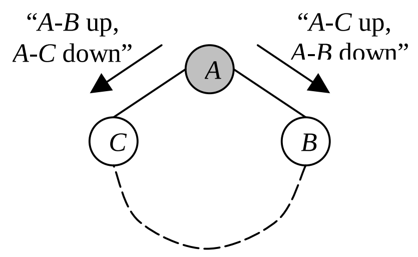


Figure 3.4: Topology changes used to delay incoming messages

3.5 Kamikaze routers

Implementing a distributed *Conservation-of-Flow* test may cause Good routers to be labeled Bad. If a Bad router neighbors a router critical to the network, it may be in its interest to force both the critical router and itself to be declared as Bad. In Figure 3.5, Bad router *A* can announce its WATCHERS counters in disproportion to critical router *B*. Since *B* is Good, it will dutifully pass on *A*'s information to *B*'s neighbors, who will subsequently find that both *A* and *B* fail the *Conservation-of-Flow* test. Both will be declared as Bad by their neighbors and will be *logically removed* from the network. Although no packets will be dropped or misrouted as a direct result of this attack, portions of the network may become unreachable; this may invalidate one or more of the necessary *good neighbor*, *good path*, and *majority good conditions*. Note that if ghost router creation is not prohibited, the malicious router mounting this attack may even escape detection.

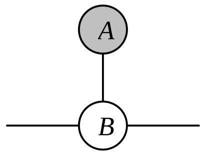


Figure 3.5: Bad router *A* neighboring critical router *B*

3.6 Source routing

Discarded packets must be accounted for. One example arises with source routing. A malicious router can place on the network a self-addressed packet, requesting either loose or strict source routing, and specifying either a logically removed or non-existent router as a required hop. Setting a legitimate and reachable router as the hop immediately preceding the unreachable one specifies the target of the attack. We postulate that intermediate routers will only check whether the next hop is reachable, thus, when the packet arrives at the router under attack, it will be forced to drop the packet. This router will be identified as Bad for failing WATCHERS' *Conservation-of-Flow* test.

3.7 Premature Aging

Internet packets have a Time to Live (TTL) field that, upon reaching 0, will cause the packet to be discarded. Even if secure transport and routing protocols are used, packets can still be prematurely aged: Similar to Section 3.4, a router can set the TTL field to 1 in both originating and transient packets, forcing the next hop to drop the packet before reaching its destination. Generalizing this, any successive router along the path could be forced to drop the packet by setting the TTL to a value less than the remaining distance to the destination.

Similar attacks are possible against link-state routing protocols; use of these protocols is required by the *link-state condition*. The WATCHERS paper [1] suggests using OSPF [4], unfortunately OSPF v2 contains vulnerable fields in its topology-advertising broadcast packets [6]. When these topology packets, or Link State Advertisements (LSAs), are passed among routers in an AS, the Age field is incremented. Upon reaching MaxAge, associated topology information is no longer used in calculating the routing table. At that time, the LSA is re-flooded across the network with its Age set to MaxAge, forcing the originating router to increment the LSA's sequence number and try resend.

Thus, by setting the Age field to MaxAge for all transient LSAs, a malicious router can force frequent retransmissions, potentially saturating the network. Note that the Age field is the only one excluded from OSPF message checksums. Due to the rapid topology announcements, routers are also more likely to have inconsistent views of the network topology at any given time, possibly with sections of the network being unreachable.

3.8 Attack Summary

Each of these attacks plays havoc with an assumption WATCHERS makes. The WATCHERS algorithm asserts that an altered packet is equivalent to a misrouted packet. While true in a philosophical sense, the two can act very differently. The attack in Section 3.3 requires, as does the WATCHERS specification, that all routers running the protocol know the network topology. Section 3.4 assumes that changes to the topology can occur faster than the time of a WATCHERS round. The attacks in Section 3.6 and 3.7 force packets to be discarded en route, thereby violating WATCHERS' Conservation of Flow.

4. Questionable Assumptions

WATCHERS implicitly makes several assumptions. In this section we examine those assumptions in detail.

4.1 Assumption: Spoofing and packet modification will not occur

In order for WATCHERS to function correctly, routers must not be allowed to spoof Administrative messages (WATCHERS, network topology, *etc*.) or modify packets as in Sections 3.1, 3.2, and 3.7. If a Bad router changes a packet's destination address without detection, the WATCHERS packet counters will not reveal any misbehavior. Alternatively, if network topology messages can be spoofed, the problems in Sections 3.3 and 3.4 are compounded.

Possible solution: WATCHERS currently verifies the integrity of its own communications, however, this must also be done for network topology messages. OSPF claims all messages authenticated, but two of its supported authentication options are “null” and simple password checking, each inadequate to prevent the issues raised in Section 3.

Integrity checking of routed packets has been discussed extensively elsewhere [5,7] and is an element of IPv6 [3]. However, these features are not ubiquitous throughout the current Internet. Hence WATCHERS must include this as a requirement. See Section 5.1 for a discussion of IPv6 and its impact on these scenarios.

4.2 Assumption: All possible routing actions and methods are observable and appropriately validated by the WATCHERS protocol

Misrouted, forged, modified, and expired packets must also be accounted for in the WATCHERS protocol. Whatever the situation, packets cannot simply be dropped, lest the router become suspected of doing so with legitimate packets.

Additionally, [1] does not indicate how to account for broadcast, multicast, and fragmented packets when updating its counters. Although an easy answer would be to eliminate these from analysis, this would open the door to Bad routers dropping such packets undetected.

Possible Solutions: (1) For misrouted packets, the receiving node should simply forward the packet to the next hop according to its routing table. Only when routers do not share a consistent view of the network topology will nodes other than those actually misrouting be accused as such. At tremendous expense, routers *could* be required to synchronize their routing tables, and hold them constant, while messages are being exchanged.

In any scenario, diagnosis of a Bad router might include what type of malicious behavior was detected, thus enabling system administrators with additional room for insight, *e.g.*, if routers are being accused primarily of misrouting, the source of the problem may only be a malfunctioning link-state protocol subsystem.

(2) Packet modification and forging are not among the behaviors WATCHERS was designed to detect. In order to comply with the Conservation of Flow principle, these packets must also be sent on. Such packets are detectable using authentication and integrity checking; the destination node should be empowered to resolve the problem itself.

(3) Broadcast, multicast, and fragmented packets do not observe the Conservation of Flow principle. A single packet may induce the creation of numerous packets, or the reverse for defragmentation. By requiring that routers discover the MTU for the path a packet is likely to take, we can significantly decrease the likelihood of fragmentation by requiring that message lengths remain below this value.

Broadcast packets can be accounted for if the network topology consists only of pair-wise connections, *i.e.*, each link connects only two routers: Consider each broadcast packet sent over a link as a message originating with the sender and destined for the receiver. However, if more than two nodes share a common link, *e.g.*, an Ethernet repeater, only the MAC layer is aware of the originating network address [2]. Here, two equally inelegant options exist: Either operate all WATCHERS routers in promiscuous mode, enabling them to obtain the originator's identity, or ignore broadcast packets entirely. If the originator’s identity cannot be verified, a malicious router may resend a broadcast message just received, making it appear as though the message was sent twice from the same router. It should be noted that the WATCHERS model requires that the topology consist exclusively of pair-wise connections. An extension to handle this discrepancy is discussed in Section 4.5.

Handling multicast packets is considerably more complicated because some routers pass them through as one packet, and others will expand them into multiple packets. We leave this an open question.

(4) See Section 4.4 for discussion of expired packets.

4.3 Assumption: Routers that have external links are not required to be Good

We define an external machine or network, either inside or outside the AS, as one that does not participate in the WATCHERS protocol. An external link is simply a link connected from a participating node to an external one. Any node connected to an external system not participating in the WATCHERS protocol can arbitrarily drop packets to and from that system or, assuming authentication and per-source counters are not employed, substitute internal packets with externally-originated ones with the same destination or vice-versa.

Possible Solution: It is not appropriate to view the set of all external routers as a single external node as [1] does, as they may not be connected themselves. Instead, we should model external routers (including terminals, *etc*.) in a more realistic fashion, *e.g.*, each set of interconnected external machines as a single node. Just as the *good path condition* provides a trustworthy path within the WATCHERS network, we should require that each node connected to an external network also be Good. Note that it may prove difficult to obtain the correct source or destination for the purpose of updating WATCHERS' counters when a packet originates with, or is destined for, an external node, particularly if external links connect this node to multiple WATCHERS participants.

4.4 Assumption: Inconsistencies in nodes' views of the network topology will be short-lived and will have only minor affects

Two problems arise here. If all changes in network topology are broadcast using an unreliable protocol, there is no guarantee that such a message will be received, thus making it possible for topological inconsistencies to persist. Although protocols such as OSPF guarantee that topology messages will be received, they do not guarantee their timeliness. As in Section 3.7, a prematurely aged LSA may arrive before legitimate copies, nullifying any effect they might have. Secondly, a malicious, undetected Bad router can announce potentially false topology changes at will. Not only can these messages be generated rapidly and in great quantity; such thrashing can potentially bring down a WATCHERS network as in Sections 3.3 and 3.4.

Possible solutions: (1) Do not allow the addition of new routers to an existing network running WATCHERS. This inhibits the creation of "ghost" routers, thus restricting the number of Good routers that can be falsely convicted as Bad. Note that in a ring network with only three nodes, if one is corrupt, the other two can still be convinced that each other is Bad (the corrupt node participates in the voting process).

(2) Account for the dynamic nature of IP's TTL and OSPF's Age and Sequence Number fields. In order to ensure no router (or group of routers) inappropriately increment or decrement such fields, all routers must be aware of the specific value, if any, each router is allowed to add or subtract. While TTL is always decremented by 1 for each hop taken, OSPF's Age field may be increased by an arbitrary amount on a per-interface basis. Once these offsets are known, all routers can compute what value a received packet's TTL and/or Age field should have, based on the path the packet is *assumed* to have taken (if we were to rely on a route recorded in the IP header, it must have been integrity-checked).

(3) An expensive alternative (or addition) might be to keep per-source per-destination counters for every TTL value, and perform a modified *Conservation-of-Flow* test. This test would take into account that TTL should be decremented by 1 at each hop, *e.g.*, when x transient packets, each with a TTL of t, where $t>2$, with a specific source/destination pair, are sent through an intermediate router, neighboring routers should see a total of x packets with that same source/destination pair leave that router with a TTL of $t-1$. (This method may need significant modification to handle OSPF's Age field, since the offset values are arbitrary.)

One potential attack remaining would be for an intermediate router to swap the TTL of two packets with the same source/destination pair. This can only be effective in causing packets to be dropped if some packets with the same source/destination pair differ in their initial TTL and the destination can be made farther away from the source than the smallest initial TTL used.

4.5 Assumption: *Conservation-of-Flow* analysis will be performed on all participating WATCHERS nodes

If a link goes down during a particular WATCHERS round, and the attached routers attempt to send traffic over it, one or both routers may conclude the other is Bad if the *Conservation-of-Flow* test is performed. For OSPF, unless a lower-level protocol informs it that a link is inoperative, RouterDeadInterval seconds must pass without hearing Hello packets before OSPF declares that link as down.

Alternatively, if the *Conservation-of-Flow* algorithm ignores traffic sent over a downed link during a round, any discrepancy between the attached nodes' counters will not be discovered. If a Bad router can intentionally down its links for a portion of a WATCHERS round, it may escape analysis.

Possible solutions: (1) As a precaution, whenever a link does down, perform the *Conservation-of-Flow* test. This may have the unfortunate side effect of causing connected Good routers to be labeled as Bad, but it ensures that a Bad router cannot continue to capitalize on downed links.

(2) If it is desirable to place the blame for a failed link where it is due, a modification to the WATCHERS model is suggested: treat all links as intermediate nodes. If a link fails, it corresponds to this intermediate node being Bad. One caveat is that these intermediate nodes would not participate in the WATCHERS protocol themselves. As such, some router misbehavior would result in an associated link being blamed instead of the guilty router. Despite this, detected Bad routers will still eventually be removed from the network if they continue to misbehave.

With this modification, all links conform exactly to the *perfect transmission condition* with respect to *all* messages, *i.e.*, all transmissions sent to a neighboring node arrive intact with no delay. This holds because we can associate any actual delay, modification, or loss of data with the intermediate node. Additionally, multi-link and multiple-interface connections are more accurately represented using an intermediate node for each interface.

4.6 Assumption: Realistic disagreements in the *Conservation-of-Flow* analysis stage can be resolved through setting appropriate threshold levels

Network congestion, unreliable transport, message latency, and Bad routers can all contribute to discrepancies in the counters used in the WATCHERS analysis. While Good networks should only experience minor problems, when an undetected Bad router exists on the network, it can exploit the problems above to generate false positives as in Section 3.4. By induction, we must either set the thresholds high enough to ignore such noise, thus missing actual problems, or we accept false positives, possibly overlooking misbehavior.

Possible solutions: (1) Guarantee that Administrative messages are not dropped. TCP/IP currently allows a saturated node to drop packets. Selectively drop lower-priority packets and guarantee bandwidth sufficient for maximal high-priority usage. This scheme may be conducive to Denial of Service attacks in providing a means by which Bad routers may squeeze out lower-priority packets by saturating the network with Administrative messages; however, such behavior may be discovered by intrusion or anomaly detection systems.

(2) Use a reliable transport mechanism, *e.g.*, TCP, for all Administrative communication. Neither of these solutions solves the latency problem; rather, each focuses on limiting the damage potential of Bad routers with respect to network congestion and transport reliability.

4.7 Assumption: Messages are not passed simultaneously, and routers have no associated delay in WATCHERS' proof of correctness

Even if WATCHERS' four required conditions hold, plus the two additional proof requirements, namely the *perfect transmission condition* and *neighbor agreement condition*, a Good router may still incorrectly diagnose another Good router as Bad.

If messages can be exchanged simultaneously, one router may have just taken a snapshot of its WATCHERS counters, while it exchanges the *request* message it just received with a datagram from an adjacent router. That adjacent router, upon receipt of the *request* message (just happening to be the one message necessary to accumulate *requests* from a majority of routers in the AS), also takes a snapshot of its counters. Because the *request* message and a datagram were exchanged concurrently, these two routers' counters will disagree, forcing each to declare the other as Bad.

Additionally, while the *perfect transmission condition* requires that there be no delay between sending and receiving a WATCHERS message, the routers themselves may still delay in placing messages on the network. If they do, when a WATCHERS round begins and counter snapshots are taken, a node may still have a transient packet waiting to be sent, in which case it appears as though this packet is missing to the *Conservation-of-Flow* test. Again, a Good router may potentially be incorrectly labeled as Bad.

Possible Solution: If we must guarantee that Good routers are never falsely diagnosed as Bad, we must ensure either the thresholds are sufficiently high or that no transient packets remain in the AS while snapshots are taken of WATCHERS' counters. The latter can be accomplished by additional synchronization among the participating routers, however, because the network would need to effectively shut down for a period of time, depending on the frequency of the WATCHERS rounds, this may result in unacceptable delays.

5. Discussion

The original WATCHERS specification is elegant in its simplicity. However, the domain it intends to model, namely network routing, is inherently complex. This disparity reveals itself when we attempt to apply WATCHERS to real-world networks.

The underlying problem is that the Conservation of Flow equations do not take discarded packets into account. In particular, if a router drops a packet, it is assumed to be malicious. But IP packets may be discarded for a variety of reasons, many of which are behaviorally correct (such as the TTL expiring).

For an unreliable protocol like IP, Conservation of Flow inherently fails. Unreliability is a hallmark of IP. Protocols that use it either provide reliability at a higher layer (like TCP, which underlies *telnet*), or accept unreliability because reliability adds expense (like UDP, which underlies *ntp*).

One possible solution introduced in Section 4.4 is to modify WATCHERS to accept unreliability by augmenting the counters with other counters for discarded packets. This balances the Conservation of Flow equations and accounts for legitimately dropped packets.

Each host maintaining one such counter, keeping track of how many packets it drops, would be insufficient. A Bad router could simply claim a packet's TTL expired when it did not, and therefore drop valid packets undetected. Hence, each host must keep a counter for those packets that will be dropped at the next router. But two Bad routers in succession could defeat the counting. A quick induction shows that each router must track the TTL of all packets passing through it, and these numbers must be correlated at the verification time with the discard counts of the routers at appropriate distances.

As with these TTL counters, a similar set must be created to monitor the Age of topological updates sent via OSPF. Finally, additional counters must be created to balance the Conservation of Flow equations if packets can be dropped legitimately for other reasons, *e.g.*, saturation. This area requires more study, and repairing the equations for Conservation of Flow to account for realistic networks requires additional research.

Assuming Conservation of Flow equations can be made to account for all messaging behaviors, a number of prerequisites must still be met. Even the when the four conditions that WATCHERS requires hold, numerous problems remain. Among the more costly are the needs for integrity and authentication (asymmetric message encryption), a static network topology, and guaranteed, timely receipt of Administrative messages. Each of these alone would adversely affect the efficiency of any WATCHERS implementation

5.1 IPv6

Although IPv6 promises greater reliability and security, its use still leaves WATCHERS vulnerable to many of the attacks in Section 3. Specifically, IPv6 can detect both header and payload modification, but only at the destination, and only when the Authentication Header and

Encrypted Security Payload are used. The result is that modified packets may cause intermediate WATCHERS routers to incorrectly increment their counters. This is consistent with the "steel tunnel" analogy: only the source and destination can be confident of message authenticity and integrity. It is suggested that intermediate source address verification may be accomplished by some variant of the Authentication Header, but is currently left as an open question [3].

Introducing another potential attack, IPv6 routers will not fragment packets already on the network, but will instead drop them and return an ICMP message if they are too large for the next hop. Again, because WATCHERS does not consider this ICMP message as compensation for the dropped packet, the *Conservation-of-Flow* test will fail.

Another compounding aspect of IPv6 is that its OSPF link-state database will not be shared with the IPv4 database; IPv6 OSPF and IPv4 will run in parallel, significantly increasing WATCHERS' memory and computational requirements on an AS supporting both IPv4 and IPv6. One offsetting simplification IPv6 offers is a single 32-bit identifier for each router, independent of its network addresses.

5.2 Miscellaneous

Note that while TCP/IP scenarios have been discussed exclusively in this paper, WATCHERS suffers similar shortcomings when applied to IPX/SPX, SNA, and other network protocols, due to their similarities to TCP/IP.

It should also be noted that WATCHERS tries to have the good routers agree on values for counters. In this sense, it is an attempt to solve the Byzantine agreement problem in a specific context. Unfortunately, adapting WATCHERS to use a solution to that problem would require more restrictive assumptions (such as no more than *N*/3 Bad routers), and the cost would be prohibitive.

6. Conclusion

This paper reviewed the WATCHERS protocol, and showed that its assumptions do not accurately model the existing network. For example, networks drop packets for legitimate reasons. Hence the issues highlighted by applying an idealized law (the law of Conservation of Flow) to a realistic situation (network routing) render an excellent idealized algorithm ineffective in practice.

7. References

[1] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson, "Detecting Disruptive Routers: A Distributed Network Monitoring Approach," *Proceedings of the 1998 IEEE Symposium on Security and Privacy* (May 1998). pp. 115-124.
[2] J. Davidson, *An Introduction to TCP/IP*, Springer-Verlag, New York, NY (1988).
[3] Christian Huitema, *IPv6: The New Internet Protocol*, Second Edition, Prentice-Hall, Inc., Upper Saddle River, NJ (1998).
[4] J. Moy, *OSPF Version 2*, RFC 2328 (April 1998).
[5] R. Hauser, T. Przygienda, and G. Tsudik, "Reducing the Cost of Security in Link-State Routing," *Proceedings of the 1997 Symposium on Networked and Distributed System Security* (Feb. 1997).
[6] D. Qu, B. M. Vetter, F. Wang, R. Narayan, S. F. Wu, Y. F. Jou, F. Gong, and C. Sargor, "Statistical Anomaly Detection for Link-State Routing Protocols," *Proceedings of the 1998 International Conference on Network Protocols* (October 1998). pp. 62-70.
[7] K. E. Sirois and S. T. Kent, "Securing the Nimrod Routing Architecture," *Proceedings of the 1997 Symposium on Networked and Distributed System Security* (Feb. 1997).