On Design and Evaluation of "Intention-Driven" ICMP Traceback

Allison Mankin, Dan Massey USC/ISI
Chien-Long Wu NCSU
S. Felix Wu UCDavis
Lixia Zhang UCLA

Contact: S. Felix Wu
(wu@cs.ucdavis.edu, fax: 1-530-752-4767, tel: 1-530-754-7070)

ABSTRACTION

Since late 1999, DDoS (Distributed Denial of Service) [1,2,3] attack has drawn many attentions from both research and industry communities. Many potential solutions (e.g., ingress filtering [6,7], packet marking [5,8,9,10,11] or tracing [4], and aggregate-based congestion control or rate limiting) have been proposed to handle this network bandwidth consumption attack. Among them, "ICMP traceback (iTrace)" is currently being considered as an industry standard by IETF (Internet Engineering Task Force). While the idea of iTrace is very clever, efficient, reasonably secure and practical, it suffers a serious statistic problem such that the chance for "useful" and "valuable" iTrace messages can be extremely small against various types of DDoS attacks. This implies that most of the network resources spent on generating and utilizing iTrace messages will be wasted. Therefore, we propose a simple enhancement called "**Intention-Driven**" iTrace, which conceptually introduces an extra bit in the routing and forwarding process. With the new "intention-bit", it is shown that, through our simulation study, the performance of iTrace improves dramatically. This work has been proposed to IETF's ICMP Trace-Back working group.

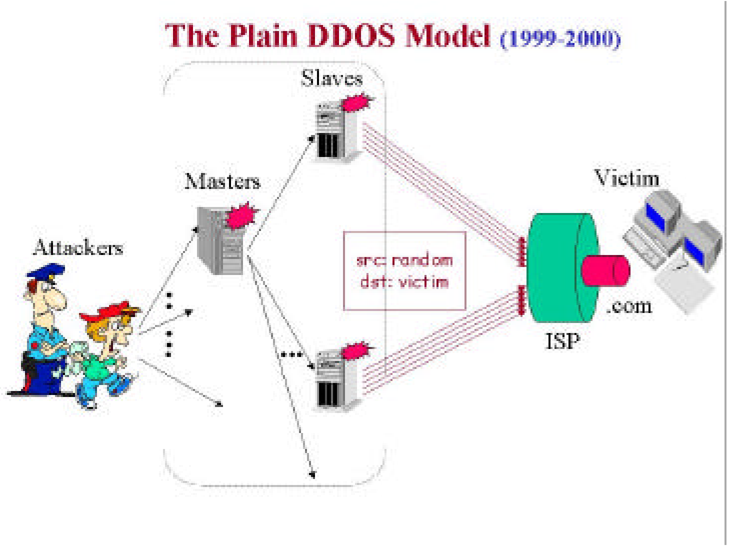
1. Introduction

A network-based intrusion detection system (IDS) might be able to detect an attack instance (either an attack packet or a sequence of attack packets) by automatically extracting and analyzing the attack signatures from a collection of incoming and outgoing data packets. However, because of the source accountability problem of today's Internet, an IDS generally can not tell where the attack packets were originated. Recently, the problem of attack source tracing has drawn a lot of attentions as many DDoS (Distributed Denial of Service) attacks (which affected many popular web sites such as Yahoo!, E-Bay, CNN among many others,) utilize IP source address spoofing.

The following are a few terms we will use to describe the problem and our proposed solutions in this paper:

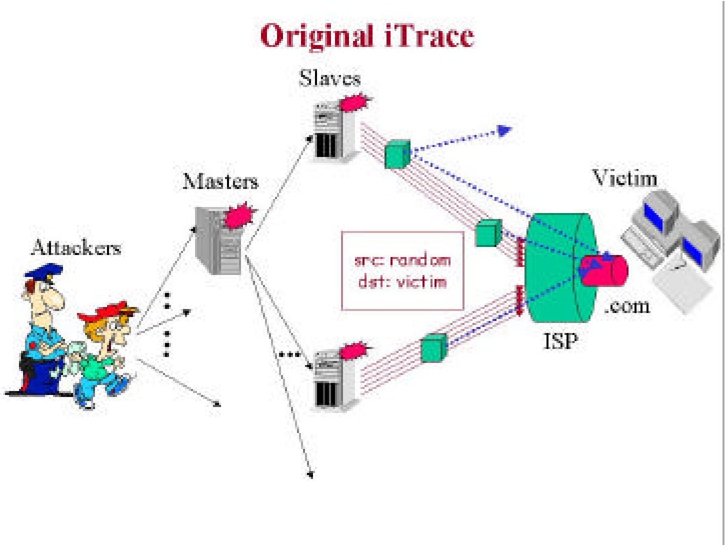
- ***DDoS Attack Infrastructure***: Hackers form their own community and they share resources among themselves. When one Internet host is compromised (a resource for the hackers), the host identity and the key to access this host is announced to all the hackers. Gradually, compromised hosts are organized and connected together as a DDoS attack infrastructure. In this infrastructure, some hosts play the role of ***masters***, while others are ***slaves***.
- ***Attacker***: An attacker is the person who utilizes the DDoS attack infrastructure to launch attack instances against a particular victim. For instance, the 15 years old Canadian boy was the attacker who launched the attack against e-bay through the plain DDoS attack infrastructure from the hacker's community.

- ***Master***: A master receives attack commands from attackers. It then sends out similar commands, through some special signaling protocol, to a set of ***slaves***. ***Masters*** are compromised hosts with "DDoS master software" installed.
- ***Slave***: A slave, after receiving the attack commands from its master, sends out lots of malicious packets toward the victim. In the case of reflective DDoS attacks, it will send lots of reflective packets toward a hugh set of reflectors. In the worst case, every single reflective attack packet is for a different reflector. ***Slaves*** are compromised hosts with "DDoS slave software" installed.
- ***Victim***: Under the DDoS attack model, the network bandwidth toward this attacked victim will be eventually taken away by the collection of DDoS or reflective DDoS packets.
- ***Reflector***: A reflector is an innocent Internet node (pretty much almost every node is possible) that will offer one or more Internet related services without verifying the source IP address before sending out the first response packet. For instance, any node with a TCP/IP protocol stack can be used as a reflector as it will send a TCP "SYN-ACK" or "ERROR" message toward the source IP address contained in the original "SYN" request packet. Please note that reflectors are innocent because, in many Internet services, there is no way to verify the source before sending out the first reply packet. We will discuss the issue of reflective DDoS attacks in a separate paper.



DDoS attack has drawn a lot of attention lately from both the research and industry communities. At least four or five solutions have been proposed lately to handle the plain DDoS attack: probabilistic marking, IP hashing, deterministic tunnel marking, ICMP traceback, egress/ingress filtering, and finally, ACC (Aggregate-based Congestion Control) with push back. Savage and others proposed to use the IP identity field (16 bits) to include some partial route path information. Probabilistic ID-field marking requires modifications of Internet routing devices to generate such marks on the fly. On the other hand, the DECIDUOUS system [5] utilizes the IPSec protocol suite such that marks are essentially IPSec authentication headers. While both Savage's[9] and DECIDUOUS place some "routing path" marks directly on the data packets themselves, IETF iTrace working group has introduced an additional new message "ICMP trace back" [4] (or simply an iTrace message) toward the victim. Under the assumption that a hugh amount of traffic will be generated toward the victim, a small probability (1 over 20K) will be sufficient (at least in theory) to generate a useful iTrace message to help the victim or its upstream ISP to identify the possible slaves.

While many different approaches have been proposed, one practical approach proposed and evaluated under IETF is "ICMP Traceback" (or simply "iTrace"). Briefly speaking, a router under the iTrace framework will pick one packet, statistically, out of a large amount of data packets (e.g., 20,000), and generate a special ICMP packet toward the same destination of the selected packet. When a victim is under a DDoS attack, a large quantity of bandwidth-consuming packets will be sent toward the victim anyway. Thus, "hopefully" the chance for routers along the attack paths picking the right packets to generate "iTrace" is big enough to trace the location of DDoS slaves.



Under the current iTrace proposal, the number of iTrace packets generated by a router is small, which implies a low overhead (statistically, around 0.005% if we pick one out of 20,000) to the Internet. However, if each DDoS slave only contributes a small amount of attack traffic, then the probability for a nearby router picking the right attack packet can be very small. And, this small probability **is independent of** the 0.005% we mentioned earlier. Roughly speaking, the routers closer to the victims tend to have higher probability to send iTrace messages toward the true victims because most of the attack traffic has been aggregated. On the other hand, the routers closer to the true slaves might send the iTrace messages to non-victims or the slaves themselves. This fact is due to a potentially large amount of “noisy” background traffic passing through those routers at the same time. When such routers need to generate an iTrace message, it will very likely pick an innocent packet. For instance, if a slave is located in the UCDavis campus and the victim is eBay, then the victim eBay might likely receive an iTrace message from its upstream ISP, but the chance for receiving an iTrace message directly from the egress router of UCDavis can be very small. Heuristically, the iTrace messages from eBay's upstream ISP is not "very valuable" as we pretty much know that the attack traffic must be from this ISP even without these iTrace messages. On the other hand, the value of receiving an iTrace message from UCDavis, under this case, is very high.

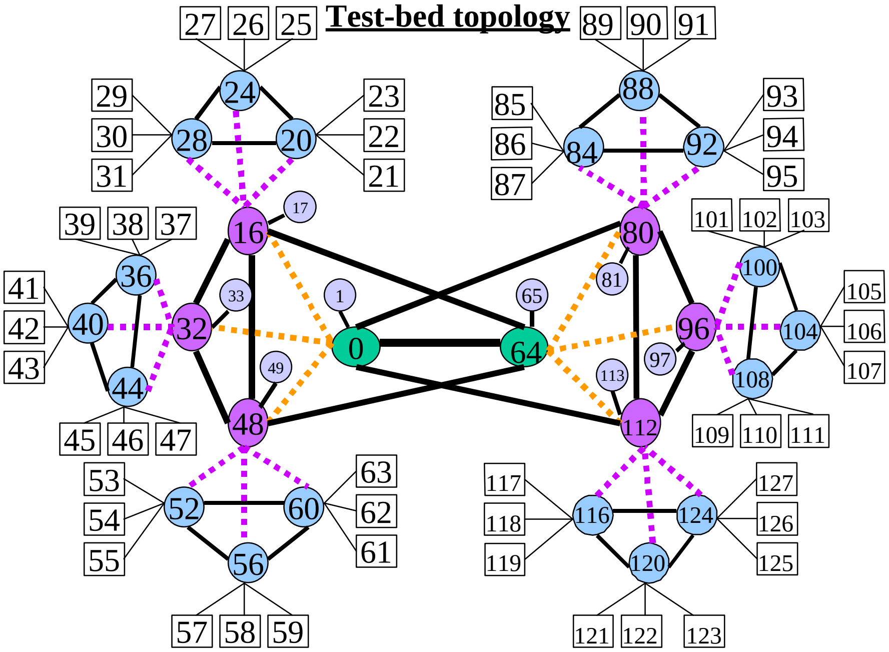
In this paper, we focus on the analysis and enhancement of the probability for a router to generate "valuable/useful" routers. We first present an evaluation model, based on the concepts of "usefulness" and “value”, for each generated iTrace messages. Second, we will quantitatively show how "valuable" iTrace messages are generated under the current iTrace proposal against two different DDoS attack models: plain DDoS, and, multi-layer DDoS Then, we will propose a simple and practical enhancement for iTrace, "Intention Driven" iTrace, which improves the values of generated iTrace messages dramatically.

2. DDoS: Simulation Testbed and Experiments

For studying the behavior of DDoS attacks as well as evaluating various different ICMP traceback proposals, we construct a network simulation test-bed (based on NS-2 simulator [12]) with 128 routers as shown below. In this section, we only discuss the topology of our testbed as well as different DDoS attack strategies. The simulation results will appear later.

Our DDoS simulation testbed has been used to performed a set of experiments:

1. *SS-SV:* *single-slave* *versus* *single-victim*
2. *MS-SV:* *multiple-slaves* *versus* *single-victim*
3. *MS-MV:* *multiple-slaves* *versus* *multiple-victims*
4. *ML-SV:* *multi-layer slaves* *versus* *single-victim*
5. *ML-MV:* *multi-layer slaves* *versus* *multiple-victims*

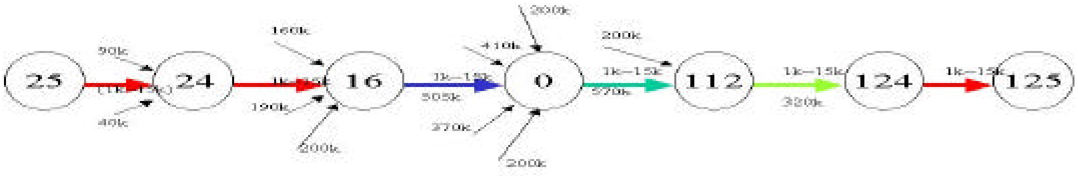


Test-bed topology

Example 1:

For instance, in the first case, SS-SV, network node 25 is the only slave, and node 125 is the only victim, while the attack path is 25→24→16→0→112→124→125. The rate of attack packets is between 1,000 and 50,000 packets per second per slave. Also, each network node has been assigned different amounts of background traffic (two examples are shown):

Node 24			%
Entry not for victim	(normal traffic)	410k packets/sec	0.9976~0.8936
Entry for victim	Normal traffic	10k packets/sec	
	Attack traffic	{1k~50k packets/sec}	0.00237~0.10638
Node 16			%
Entry not for victim	(normal traffic)	710k packets/sec	0.9986~0.93506
Entry for victim	Normal traffic	10k packets/sec	
	Attack traffic	{1k~50k packets/sec}	0.001387~0.0649



For the multiple slaves and single victim case, MS-SV, we have three slaves: 25, 95, and 117 against the single slave 125. And three attack paths are:

(1) 025→024→016→000→112→124→125
(2) 095→092→080→112→124→125
(3) 117→116→124→125.

Furthermore, for the multiple victim case, we add another victim, 41 and thus three more attack paths:

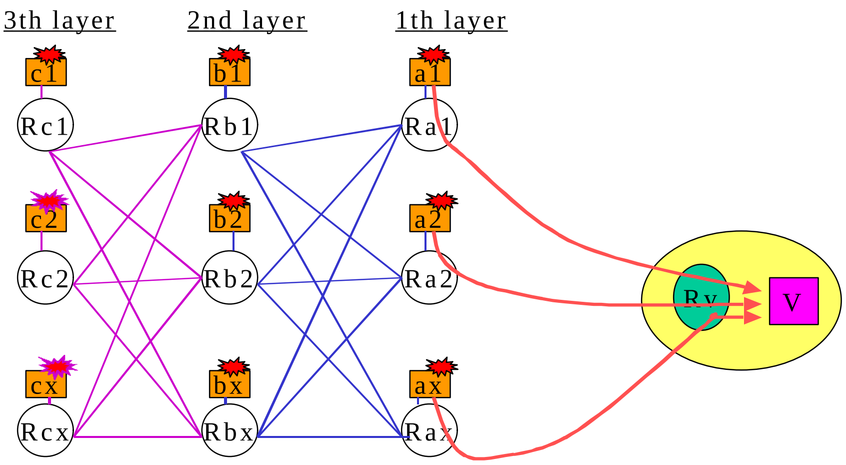
(1) 122→112→000→032→040→041
(2) 087→084→000→032→040→041
(3) 023→020→016→032→040→041.

3. Multiple-Layer DDoS Attack

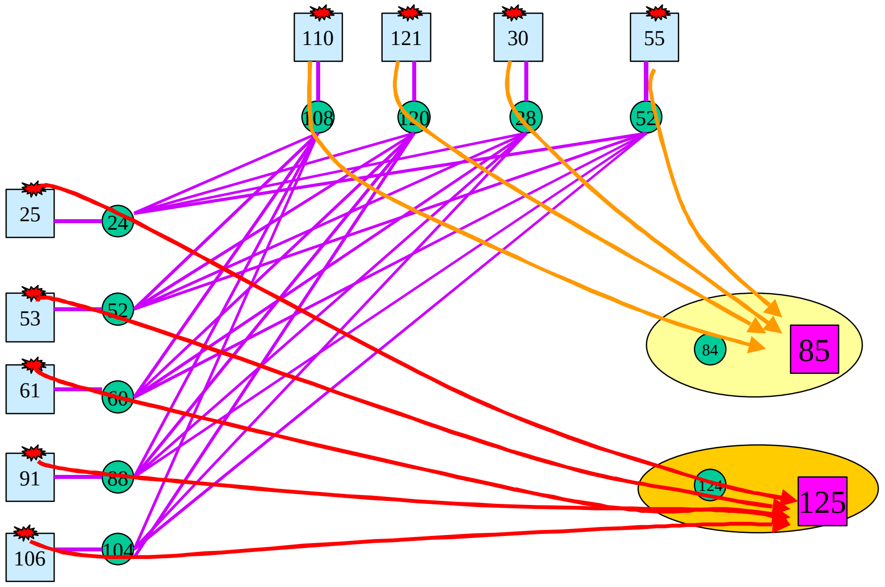
In order to demonstrate the weakness of the original iTrace proposal, we have developed a new DDoS attack model called “multiple-layer DDoS attack (ml-DDoS)”. In ml-DDoS, an attacker can use multiple layers of slaves to statistically cover each other’s attack path to some victim. The basic idea is to generate enough “artificial background traffic” for each other such that the iTrace messages will be generated for the wrong (or useless) destinations.

For example, we have a couple slaves across several UC campuses. First, we let a slave in UCSB attack e-Bay at the rate of 1,000 packets per second. If the background traffic at the UCSB gateway router is another 1,000 packet per second, then the chance to catch one of the slave packets for e-Bay is 50%. On the other hand, if two slaves in UCDavis and UCLA each send a small chunk of traffic (about 1,000 packets per second – might not be DDoS traffic) toward some destinations located in UCSB, while the slave in UCSB attacks e-Bay. Then, the chance of being caught from UCSB to e-Bay is reduced to about 25%. Furthermore, with a 50% chance, iTrace messages will be generated for the artificial noisy traffic, but only from the UCSB gateway to some destinations within UCSB itself.

One might ask: “Why not just having these three slaves each sending about 333 packets toward e-Bay?” Then, we still have about 1,000 packets per second for e-Bay while the probability of being caught is still around 25%. Our assumptions here are that either the attacker can find a way to generate traffic between two campuses without being suspected (such as ftp, web download or others) or the campus network does not process iTrace messages. ***Please note that our objective here is NOT to demonstrate that we have a new good way to DDoS. Our main objective here is to***



Multiple-layer/multiple-victim attack model



show that if the attacker manages to find a network environment that either of our assumptions holds, then the statistical tracing game might be even tougher as we will demonstrate later.

More specifically, in ml-DDoS, an attacker compromises N slaves and try to attack M victims (M could be 1 or more). These N slaves can be divided into several groups: A(a1, a2, a3, …, ax), B(b1, b2, …, by), and so on.. We can use A's slaves as the first layer to directly attack victims and use B as second layer to control the background traffic into the first layer. With this multiple-layer attack model, we will show later that routers closer to the first layer slaves will generate iTrace messages, not for the real victims, but for the slave network themselves. In our ml-DDoS simulation study, we use (25, 53, 61, 91, 106) as first layer and (110, 121, 30, 55) as second layer to collaboratively attack victim 125. For victim 85, we use (110, 121, 30, 55) as first layer, and (25, 53, 61, 91, 106) as second layer.

4. On "Usefulness" and "Value" of iTrace Messages

Potentially, the design of the original iTrace proposal might have one critical disadvantage: the routers closer to the true slaves will be unlikely to generate "useful" iTrace messages toward some victims who really desperately need those iTrace messages. In order to verify this potential weakness, in this section, we define two different measures to evaluate the effectiveness of a particular tracing technique: “usefulness” and “value”.

Definition 1: “iTrace”

An iTrace message (iTr) contains mainly three pieces of information: the identity of the router generating this iTrace message (iTr.rtr-ID), a destination address that will receive this iTrace message (iTr.dst-ID), and the packet picked by the router (iTr.pkt).

In the original iTrace proposal, the concept of “intention” does not exist, and the iTrace box is merely a statistic packet generation box without the concern about the “intention” of the receiver. We found the concept of “intention” very critical in tracing DDoS attack sources. In today’s Internet environment, there are so many possible receivers for iTrace messages, and therefore, for efficiency, we need to know which destinations are indeed interested in utilizing information from the iTrace framework. In fact, adding the concept of “intention” into iTrace is the key contribution of this paper.

Definition 2: “Intention”

Each destination node has an “intention” value associated with it. If the intention value is 1, then it implies that this network node is interested in receiving iTrace messages. Otherwise, the intention value should be zero.

An iTrace message (iTr) is not very “useful”, if (iTr.pkt) is not an attack packet, or (iTr.dst-ID) is not interested in receiving iTrace messages. In other words, if a network address is not under attacks (or the IDS does not detect it) or it does not care about tracing (even it is under a DDoS attack), the iTrace message will not make any difference.

Definition 3: “Usefulness”

If the packet contained in an iTrace message (iTr.pkt) is an attack packet and the node iTr.dst-ID is interested in receiving iTr (or Intention(iTr.dst-ID) = 1), then we call this iTrace message “useful”. Therefore, an iTrace message will be useless if it contains an innocent packet or the destination does not want it.

Based on the definition of “usefulness”, we run simulation for SS-SV (as described in Example #1 earlier), and obtained the following simulation results on router 24, which is very close to the true slave, router 25:

node 24's itraced_table   Total itraced pkts= 1200

Src	Dst	count	%	intention	usefulness
25	125	5	0.004167	1	1
126	26	1195	0.995833	0	0

It is not surprised that, because of the amount of background traffic, for the first 1200 iTrace messages generated by router 24, most (99.58%) of the iTrace messages was NOT toward the victim. After examining closely on the simulation log, we found that the generation sequence numbers of those five useful iTrace messages by router 24 were #293, #429, #707, #817, and #1107. I.e., the first useful iTrace message was generated after router 24 generated 292 useless iTrace messages toward other destinations. If one iTrace message represents 20,000 data packets, this implies that the first useful iTrace message is generated after router 24 forwarded about 6 million data packets.

While the first measure, "usefulness", is simple, but it does not catch up certain important aspects of iTrace behavior. First, for instance, in example #1, an iTrace message generated by router 24 will be as "useful" as an iTrace message from router 124, according to our simple definition of usefulness (Definition #3). In reality, this is certainly not true. Second, in Definition #3, for the same victim, receiving two iTrace messages from the same router is as useful as from two different routers. We argue that this should not be the case as duplicated iTrace messages, in theory (i.e., we neglect the fact that some messages might be dropped during the DDoS attack), offer no more information about the slave's location than the first iTrace message. Third, we observe that, in responding to DDoS attacks, in order for a system administrator to quickly recover from the damage, it is highly desirable to receive iTrace messages immediately after the attack. For instance, in example #1, the first iTrace message from router 24 toward the true victim 125 is generated AFTER router 24 generated 292 useless iTrace messages (roughly 2 million packets). If router 24's throughput is 10,000 packets per second, it will take about 600 seconds before we can identify router 24. Clearly, it will be better if we can receive this first iTrace message in an earlier stage.

With the above three observations, we add three more attribute into our "value" evaluation of iTrace messages:

An iTrace message (iTr) is not very "valuable", if iTr[rtr-ID] has sent iTr[dst-ID] at least N iTrace messages within the past K seconds, while N and K will depend on parameters such as how many iTrace messages will be dropped. This observation implies that duplicate iTrace messages over a short period of time might not be very efficient in iTrace message generation. In other words, if the Internet has multiple DDoS victims simultaneously, then each router in the network should try to generate iTrace messages for ALL the victims. It is NOT very valuable if a router only generates iTrace messages for a single victim over and over again.

Furthermore, an iTrace message is less "valuable," if iTr[rtr-ID] is only X (e.g., X <= 3) hops away from the victim, iTr[dst-ID]. For instance, if iTr[rtr-ID] is only 1 or 2 hops away from iTr[dst-ID], then this iTrace message does not provide too much "new" information about the attack source. If the victim has only one (or two) ISP, then we can derive pretty much the same conclusion without the iTrace messages. However, it is much more useful, if we can receive iTrace messages from routers that are, for example, 10 or more hops away from the victim.

Finally, an iTrace message is more "valuable" if the iTrace message is generated right after the attack begins, which gives the intrusion response system more time to handle the problem. Therefore, in our evaluation, we give more "value" to iTrace messages being generated earlier in the attack cycle.

Definition 4: "Value"

The information value of an iTrace message iTr is determine by the following five parameters (the first two have been covered in the definition of "usefulness".):

Value(iTr) =

F(Attack[iTr.pkt], Intention[iTr.dst-ID], HopCount[iTr.rtr-ID→iTr.dst-ID],
Received[iTr.rtr-ID→iTr.dst-ID], Generated[iTr.rtr-ID]),
where Received[iTr.rtr-ID→iTr.dst-ID] represents how many iTrace messages from iTr.rtr-ID to iTr.dst-ID have been received before, and Generated[iTr.rtr-ID] represents the number of iTrace messages already generated by iTr.rtr-ID for all destinations.

Please note that, while the function “F” can be possibly defined in many different formats, in this paper, we use a very simple form of “F” for the purpose of evaluation:
Value(iTr) =
(Attack[iTr.pkt] * Intention[iTr.dst-ID] * HopCount[iTr.rtr-ID→iTr.dst-ID]) /
((Received[iTr.rtr-ID→iTr.dst-ID] + 1) * (Generated[iTr.rtr-ID] + 1)),

For example, if an iTrace message iTr (generated by an UCDavis router) contains an attack packet toward e-Bay, then:
Value(iTr) = (1 * 1 * 8) / ((0+1) * (0+1)) = 4.0
I.e., this iTrace message is “useful”, the hope count between rtr-ID and dst-ID is 8, it is the first iTrace message from rtr-ID to dst-ID, and finally it is the second iTrace message rtr-ID has ever generated. For the same example:
Value(iTr) = (1 * 1 * 8) / ((1+1) * (7+1)) = 0.5
except this is the second iTrace message from rtr-ID to dst-ID, while this is the eighth iTrace message the router rtr-ID ever generated.

According to Definition #4, most of the first 1200 iTrace messages generated by router 24 in Example #1 are value-less except:
Value (iTr-r24-0293) = (1 * 1 * 5) / ((0+1) * (293)) = 0.0170648
Value (iTr-r24-0429) = (1 * 1 * 5) / ((1+1) * (429)) = 0.0058275
Value (iTr-r24-0707) = (1 * 1 * 5) / ((2+1) * (707)) = 0.0023573
Value (iTr-r24-0817) = (1 * 1 * 5) / ((3+1) * (817)) = 0.0015300
Value (iTr-r24-1107) = (1 * 1 * 5) / ((4+1) * (1107)) = 0.0009033
The value of the duplicated iTrace messages decrease according to the “F” function defined in Definition #4. The accumulated value (for all the useful or useless iTrace messages) generated by router 24 is 0.028, based on the “Value” measure. We will show in the next section that a simple enhancement to the iTrace proposal will boost the value to about 8.12.

5. Intention-Driven iTrace

According to the design of the original iTrace proposal, the Internet will dedicate a fix portion of network resources/bandwidth for generating iTrace messages. However, we have demonstrated that, in some cases, many of such iTrace messages are wasted. Therefore, we present a new (but very simple) mechanism called "intention-driven" iTrace to resolve the statistic problem of the original iTrace proposal. Our design objectives are:

- The probability for generating "useful/valuable" iTrace messages should be significantly higher than the original static iTrace scheme. In the worst case, the new scheme should just fall back to the original scheme, but not worse.
- The number of iTrace messages generated by each router remains the same, i.e., roughly 1 over 20,000 data packets.
- The new mechanism is compatible with the current iTrace scheme such that we do not require every router to support the new mechanism.

- The design and implementation of the new mechanism is simple, efficient, secure and scaleable.

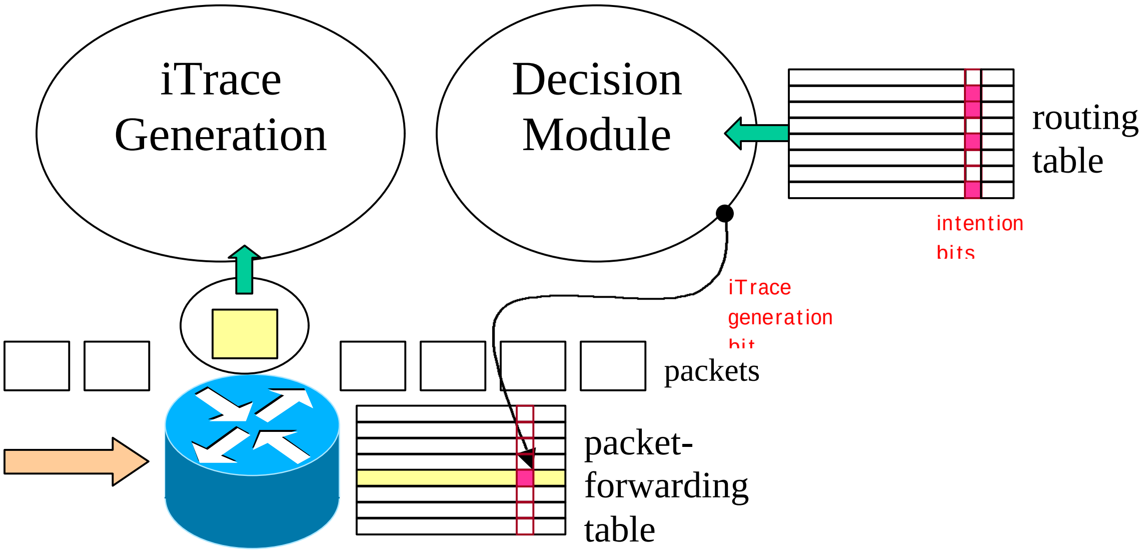
5.1. Routing Table and Packet-Forwarding Table

Since our proposed solution heavily depends on the BGP routing protocol, we will first clarify the difference between a routing table and a packet-forwarding table in a network router.

On a BGP router, we have two tables: a routing table and a packet-forwarding table. The former table is maintained by the BGP protocol implementation and it contains information such as which route (including both next-hop and AS path information) should be used for a particular destination address prefix. The content of the BGP routing table is passed into the kernel's packet forwarding table. The forwarding table is the key component for the packet forwarding process in a router. Today, a typical high-end router will have about 1 quarter million entries in its forwarding table. Usually, adding a new attribute to the forwarding table or changing the pipeline of the packet-forwarding process (usually done in hardware) is more difficult than changing the BGP protocol itself (in firmware/software). Therefore, it is also our objective to minimize any changes to the forwarding process or table.

5.2. Architecture of Intention-Driven iTrace

From a high-level point of view, "intention-driven" iTrace separates the functions of iTrace into two different modules: decision module and iTrace generation module. The decision module will determine, based on the information provided in the "routing table", which kind of iTrace messages should this router generate next? More specifically, which "entry" in the "packet-forwarding" table should be the target for iTrace generation? Based on this decision, one special bit in the packet-forwarding table (iTrace generation bit, see below) will be set to 1, and the very next data packet using this particular forwarding entry will be chosen as an iTrace message. Next, this chosen packet will be processed by the "iTrace generation" module, and a new iTrace message will be sent.



5.3. iTrace Generation Bit in the Packet-Forwarding Table

We introduce a new bit, an iTrace bit, into each entry in the packet-forwarding table**. In this paper, at any moment, at most one iTrace bit will be set to 1,** while it is possible to extend our scheme to allow multiple bits set '1' simultaneously. When the forwarding engine forwards a data packet, it will perform a table look-up to find the right entry to forward the packet. If the forwarding process sees a ‘1’ in the entry for this packet, it will send a copy of this packet to the iTrace generation module and reset that bit to ‘0’. Furthermore, The modified packet-forwarding process is described as the following piece of pseudo-code:

```
int
perDataPacketProcess
(IP *p)
{
      ForwardEntry *fe = findForwardEntry(p->destinationIPaddr);
      if (fe == NULL) return -1;
      if (fe->iTrGenBit == 1)
      {
            sendiTrace(p);
            fe->iTrGenBit = 0;
      }
      regularPacketForwarding(p, fe);
      return 0;
}
```

Please note that, while the design here requires an extra bit in the forwarding table, in practice, we have considered a similar design without requiring any changes to the packet forwarding process. However, the router needs to have some packet logging facility available for us to choose a packet from the log. We presented this design to IETF’s ICMP Trace-Back working group, and some router venders voiced their concern about adding a new bit, while some others feel that this should be possible in their commercial routers.

5.4. Intention Bit in the BGP Routing Table

We associate each routing table entry with one extra bit called "intention bit". If the "intention bit" for a particular route entry is 1, then the network destination under this route entry indicates its desire in receiving iTrace messages. **Please note that zero or more routing entries might simultaneously have intention bits on.** For instance, multiple sites might be simultaneously under serious DDoS attacks and they have running applications that will utilize the information being carried by iTrace messages.

On the other hand, if the intention bit is 0, then it indicates that either the destination's intrusion detection system does not believe that its network is under DDoS attacks or it believes that iTrace messages would not help to handle the attacks it observed. For instance, if an intrusion detection system detects that its network is under reflective DDoS attacks, then the normal (so called) "forward" iTrace messages will not help because iTrace messages will only lead to a hugh number of reflectors, but not to the real DDoS slaves.

A side issue we would like to mention here is that, at this point of time, it is not clear whether the option of "backward iTrace messages" will be adopted by the working group or not. But, if the "backward iTrace option" is adopted, then we need to introduce another bit to express the intention of receiving backward iTrace messages. In other words, in the current draft, the intention bits are only for controlling the "forward iTrace option".

5.5. Decision Module

After a router forwarding about 20,000 packets, it should emit an “iTrace trigger” to indicate the need for generating an iTrace message. This trigger will be delivered to the decision module, and the decision module will then need to make a decision: which “entry in the packet-forwarding table” needs a 1 in their iTrace generation bit. We have implemented two very simple strategies:

- Generate a random number to select one routing table entry from all the routing entries with intention bit ‘1’. And, all such entries will have exactly the same probability.
- Generate a random number to select one entry. In addition to (1), we add a statistical bias toward entries having longer ASPath toward the final destination. This heuristic is to prefer the destinations with a longer “distance” from the generation router.

After deciding which routing/forwarding entry should be the next iTrace target, the corresponding iTrace generation bit will be set. Also, if none of the entries has intention to receive iTrace, then no iTrace message will ever be sent. Therefore, our design does not introduce any more (if not less) iTrace messages than the original proposal does. The following is the pseudo code for the Decision module handling each iTrace trigger:

```
int
periTraceTriggerProcess
(RouteEntry *RETable, int RETableSize)
{
      int i;
      RouteEntry *re;
      for(i = 0; i < RETableSize; i++)
      {
            re = &(RETable[i]);
            if (re->intention == 1)
#ifdef   INTENTION_WITH_DISTANCE
                  enterSelection(re, re->ASPathLength);
#else    INTENTION_WITH_DISTANCE
                  enterSelection(re, 1);
#endif   INTENTION_WITH_DISTANCE
      }
      re = finalSelection();
      if (re != NULL)
            re->fe->iTrGenBit = 1;
      return 0;
}
```

5.6. How to Set and Distribute the Intention Bits?

So far we have discussed, based on the intention bits in the routing table, we can decide the forwarding entry for the next iTrace message. In this section, we will present our design about “how to set and distribute the intention bits?”

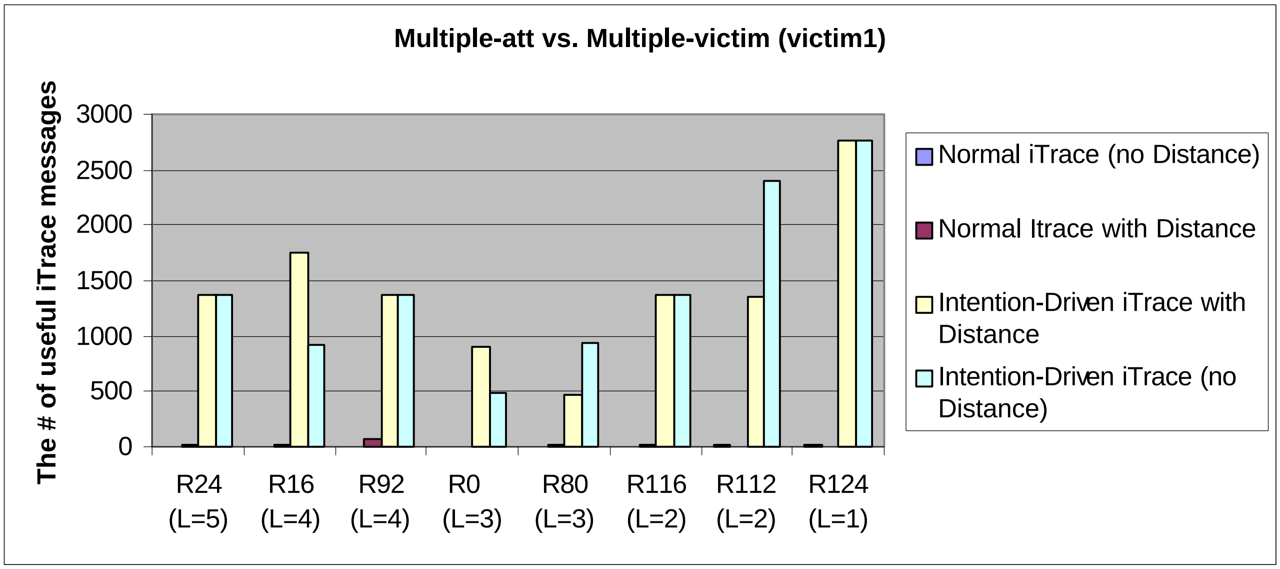
We propose to utilize the BGP routing information exchange protocol as the vehicle to distribute the intention bit values. When a BGP router advertises its reachability to some network addresses, it will also attach the “Intention bit” value for that particular network address prefix. Therefore, when this BGP update is propagated to some downstream BGP routers, the intention bit value of the corresponding route entry will be updated. Furthermore, if BGP updates are aggregated, the intention bit values will also be aggregated.

Specifically, we utilize the “community attribute” in BGP to carry the intention bit value. The community attribute in BGP is a 32 bit unsigned integer, and we introduce two new community string values, one for “Intend to receive iTrace” and the other for “Not intend to receive iTrace”. (If our proposal is accepted by IETF, we will determine the exact values for these two strings later.) Furthermore, the community attribute is “transitive” and “optional”, which means that if a router does not understand these new community values (such as an old BGP router does not support “intention-driven” iTrace), it will still forward it to the next hop. Eventually, a downstream BGP router can pick it up, update its intention bit values, and maybe perform proxy aggregation for some old upstream routers. We believe that our design is practical as it doesn’t require changes to all the routers. And, even for the new routers, the change is fairly small as we do not introduce a new BGP attribute, but two new values for an existing BGP attribute.

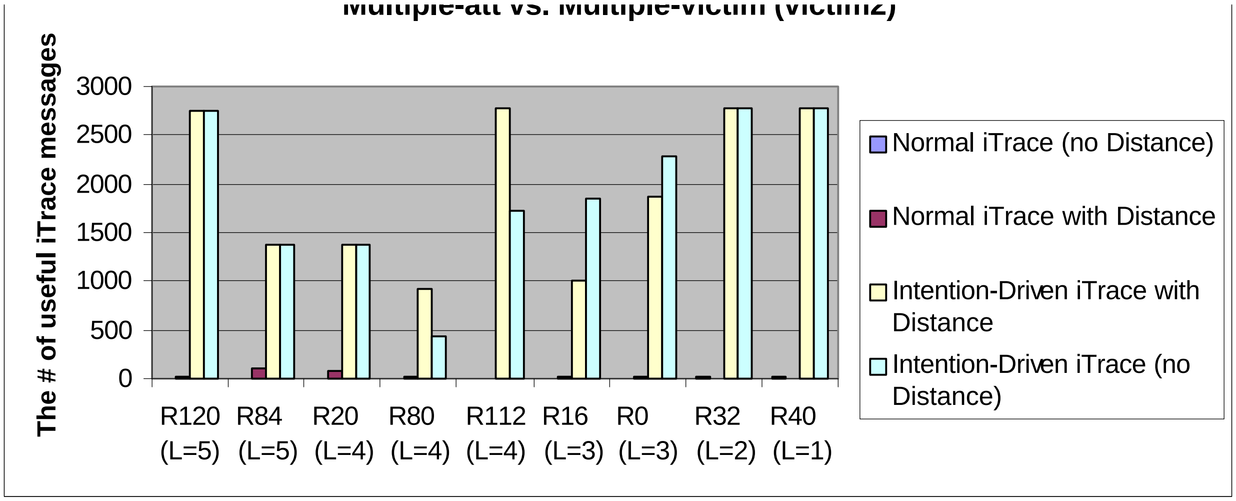
Therefore, when a particular network is under DDoS attack, the intrusion detection system will inform the BGP router to include the iTrace intention value in the next update. Since the rate of our updates can be limited by the BGP keepALive interval, our design will not introduce extra BGP traffic, while the intention bit value can be distributed through the whole Internet. Currently, it will take a few seconds (up to maybe 10~15 seconds) for BGP to propagate to the whole Internet. Finally, because of BGP route aggregation, our design will not increase the number of entries in the routing or forwarding tables.

6. Performance Evaluation

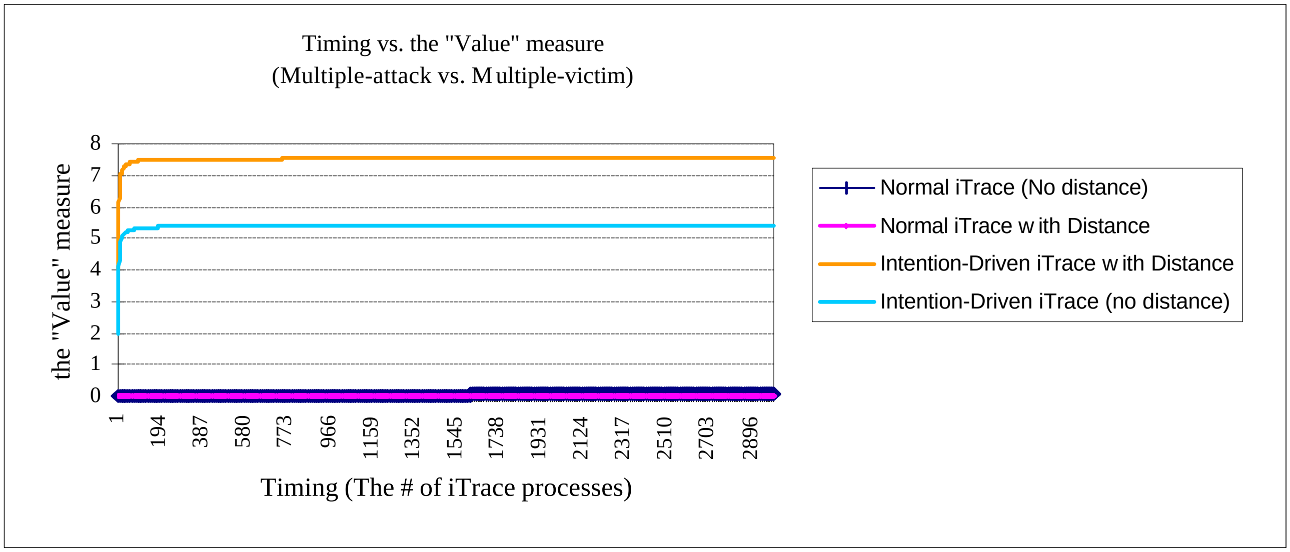
In this section, we compare four different iTrace schemes: the original one with or without distance and intention-driven iTrace, again with or without distance. Please note that, the results presented here do not include the delay introduced by the BGP propagation itself. We use two different performance measures: “usefulness” and “value” as defined in Section 4. Due to the space limitation, we will only present two cases: Multiple Slaves versus Multiple Victims (MS-MV) and Multiple Layers versus Multiple Victims (ML-MV). The following figure shows the “usefulness” of iTrace messages along “one attack path” against victim 125:



The following figure represents the same measure but for a different path against victim 41:

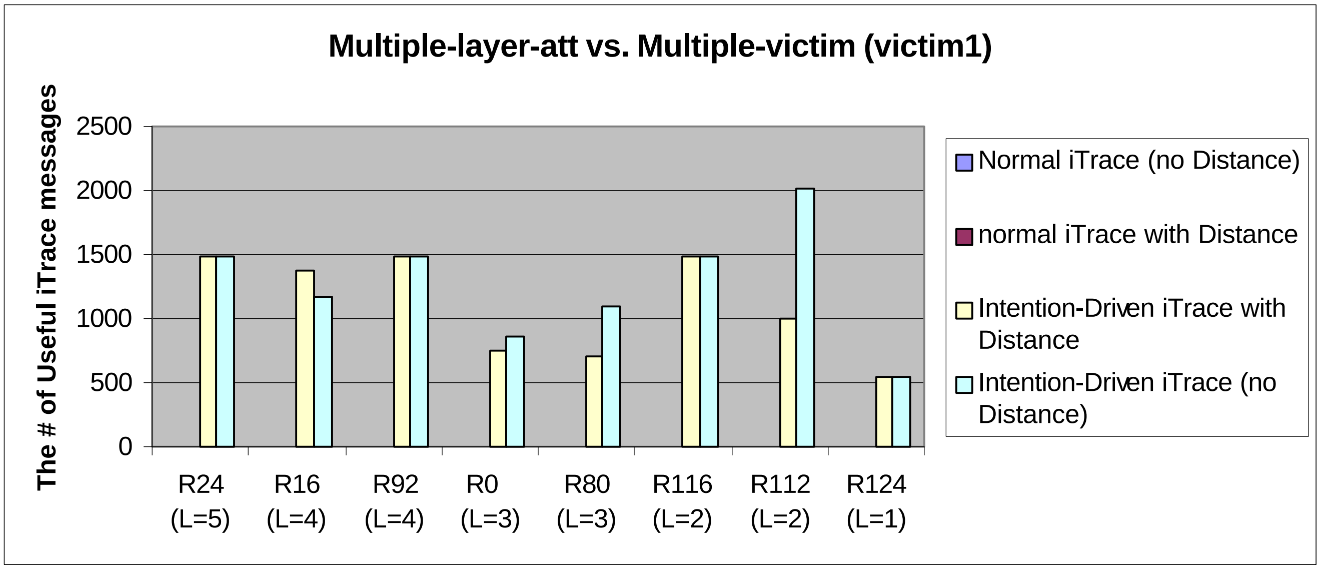


Clearly, we can observe that, under our simulation study, most of the iTrace messages for the original iTrace proposal are wasted, while it is not really clear whether the heuristic of "distance" really contributes significantly – under the "usefulness" measure. However, if we look at the "value" measure, the story is different:



From our result of "accumlated" iTrace message value over time, we conclude that:
(1). With intention-driven iTrace (regardless of distance heuristics or not), the value of iTrace information increases very fast to its maximum in the first few iTrace triggers. This result is very encouraging as it implies that we can identify the sources (even under the MS-MV case) very quickly after the attack starts. Furthermore, this also indicates that maybe one iTrace per 20,000 is an "over-kill".
(2). With the distance heuristics, the value of information is about 40% better, based on the value measure, which is biased toward "long shot" iTrace messages. We feel that we need to consider other possible measures as well as to get inputs from the operational folks (e.g., ISP) to validate whether the "value" measure we are using here is fair or not.

The following is the "usefulness" measure for ML-MV (the "value" measure is very similar to MS-MV). The interesting observation is that, even under the "usefulness" measure, the normal iTrace hardly generates "ANY" useful iTrace messages. After examining the log file, we found all the routers on the path generated at most two "useful" iTrace messages while many actually scored zero.



7. Security Consideration

Since our scheme will introduce exactly the same amount of iTrace messages as the original iTrace proposal, our proposal will not introduce any new vulnerability related to denial of service attacks based on the iTrace messages themselves.

Since we propose using BGP to distribute the intention values, our scheme is subject to the same security risks as BGP. The risks with respect to intention values would be that an attacker who can tamper with the BGP contents could modify the behavior of iTrace to divert iTrace away from the attacker's location. This attack seems as if it would be very difficult to accomplish, but the issue should be considered in more detail in the future.

8. Remarks

In this paper, we study the problem of DDoS attack source tracing, and in particular, we examine the weakness of one proposed solution, ICMP Trace-Back. We have defined two different measures for source tracing performance and performed simulation on four iTrace schemes against 5 different types of DDoS attacks. From the results we obtained, we believe that the statistic problem of iTrace is very critical and we are convinced that “intention-driven” iTrace is an important extension in order to achieve a much better tracing performance. Furthermore, our proposed solution is very practical and simple: it requires very few changes to the routing infrastructure and it does not require a global replacement.

We are currently developing a new simulation testbed to include the delay of BGP propagation. We also start to design and implement an “intention-driven” iTrace prototype on Linux to further validate our proposed solution here.

9. References

1. John Elliott, "Distributed Denial of Service attack and the zombie ant effect," IP professional, March/April 2000.
2. Computer Emergency Response Team (CERT), "CERT Advisory CA-2000-01 Denial-of-service developments," January, 2000, http://www.cert.org/advisories/CA-2000-01.html
3. Dave Ditrich, "Distributed Denial of Service (DDoS) attacks/tools resource page," http://staff.washington.edu/dittrich/misc/ddos/, 2000.
4. Steven M. Bellovin, "ICMP Traceback Messages", Internet Draft, March, 2001.
5. H.Y. Chang, S. F. Wu, and et al. "Deciduous: Decentralized source identification for network- based intrusions," in 6th IFIP/IEEE International Symposium on Integrated Network Management. IEEE Communication Society Press, May 1999.
6. P. Ferguson and D. Senie. RFC2267 Network Igress Filtering: Defeating Denial of Service Attacks which employ IP source address spoofing, January 1998
7. Cisco Systems. Configuring TCP Intercept (prevent Denial-of-Service attacks). CISCO IOS Documentation, March 1999.
8. Robert Stone, "CenterTrack: An IP overlay network for tracking DoS flooding,", October 1999. http://www.nanog.org/mtg-9910/robert.html
9. Stefan avage, David Wetherall, Anna Karlin, and Tom Anderson, "Pratical Network Support for IP traceback," Technical Report UW-CSE-00-02-01, Department of Computor Science and Engineering, University of Washington, Febrary 2000
10. Dawn Song and Adrian Perrig, "Advanced and authenticated marking scheme for IP traceback," IEEE INFOCOM 2001.
11. K. Park, and H. Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack," IEEE INFOCOM 2001.
12. Network Simulator ns-2 < http://www.isi.edu/nsnam/ns/ >

10. Acknowledgements

Our research is sponsored by DARPA under the fault tolerant networking program.