A Source Identification Scheme against DDoS Attacks in Cluster Interconnects

Manhee Lee* Eun Jung Kim* Cheol Won Lee†
*Department of Computer Science
Texas A&M University
College Station, TX-77840
manheelee@tamu.edu, ejkim@cs.tamu.edu

†National Security Research Institute
Daejeon Korea, 305-350
cheolee@etri.re.kr

Abstract

Designing secure clusters has recently become a critical issue to make these systems robust to attacks from the Internet. The Distributed Denial of Service (DDoS) attack is one of the most serious problems in the current Internet. To defend against DDoS attacks, clusters usually depend on firewalls or Intrusion Detection Systems (IDS). However, once firewall and IDS are breached, the impact of DDoS attack within a cluster can be severe. That is because one infected system or one malicious user, which is believed to be trustworthy, may instantly paralyze the whole cluster through the high speed network. In this paper, we present a deterministic distance packet marking (DDPM) scheme to identify the source nodes generating spoofed IP packets in cluster interconnects. The scheme can be applied to many cluster interconnects such as mesh, torus and hypercube, which are popular in many commercial systems. Our scheme is practically attractive since it is scalable to large networks and does not incur much processing overhead on both switches and nodes.

1. Introduction

After a series of system-downs of Yahoo, eBay, and Amazon in February 2000, people realized that DDoS attack is a very serious problem. Moreover, after experiencing world-wide network-slowdown during CodeRed worm and Nimda worms attacks in 2001, people were surprised at how powerful enough DDoS attacks could be to crash the whole Internet [5, 11, 12].

DDoS attacks deplete network or system resources, thereby disabling those services. The first generation DDoS attacks dump huge number of packets to a specific target system by using DDoS attack tools such as Tribe Flood Network (TFN) and *trinoo* [7, 9]. Aggregated traffic causes system-slowdown or even breakdown because of too large amount of traffic to handle. The second generation DDoS attacks are by worms or viruses [4, 15, 16, 18]. For example, if a user opens an infected email, the email virus starts to send the infected emails to the recipients whose addresses are stored in the mail client of the infected user. Even though these attacks do not target a specific system, it can use up system and network resources because its total traffic increases exponentially.

Wide spread use of cluster systems in a diverse set of applications has spurred significant interest in designing such servers, considering performance, scalability, and quality-of-service (QoS). In addition to these design objectives, designing secure cluster has recently become a critical issue since numerous security loopholes of cluster servers come to the forefront. To defend against attacks exploiting those vulnerabilities, cluster systems usually depend on *firewalls* or the Intrusion Detection Systems (IDS). Even though they defend against various attacks from outside, they cannot prevent all possible threats such as password leakage by users [21]. Once a hacker breaks in the cluster, the impact of DDoS attack within a cluster would be even severe since one infected system, which is believed to be trustworthy, may instantly paralyze the whole cluster through the high speed network.

Researchers have proposed new methods to improve security inside clusters. Ian *et al.* proposed a communication library allowing programmers to communicate securely in the geographically distributed computing environment [13]. Connelly and Chien focused on the tightly-coupled components. They applied traditional security functions such as transposition, substitution, and data padding on the marshalling layer [6]. Dmitriov and Gleeson presented security enhancement methods in three levels: network host interfaces, System Area Networks (SAN), and protocols for interconnecting many SANs [8]. However, to our knowledge, there has been no prior research on defending against DDoS attack from within a cluster.

There are two main challenges in defending against DDoS attacks. The first challenge is that a DDoS attack usually camouflages itself as normal traffic, hence making it difficult to detect the attack. For example, TCP

SYN flooding attack makes as many TCP half-open connections as the victim host is limited to receive. However, the individual connection has nothing wrong except that the connection does not complete three-way handshaking. In addition to detection, we have identification problem as the second challenge. Even after detecting the occurrence of DDoS attack, it is difficult to identify originators of the attack because DDoS attacks often use spoofed IP addresses, meaning that an attacker uses a fake IP addresses instead of the real source IP address. To solve this problem, researchers have proposed several approaches including packet marking algorithms such as Probabilistic Packet Marking (PPM) and Deterministic Packet Marking (DPM) [17, 19, 20]. In PPM, each router randomly selects a packet and writes its encoded IP address. After receiving many packets marked by routers on the path, the victim can reconstruct the whole path of a DoS attack. DPM, on the other hand, marks all packets with just one bit of hash value of routers' IP addresses. A series of bits marked by all intermediate routers represents a signature. Once a source or a path is identified, we can protect our system by blocking packets from that source or that path.

In this paper, we attempt to solve the second challenge, the source identification of DDoS attacks, in the direct networks where two neighboring devices are communicating with a direct link. When we use existing marking algorithms in the direct networks, we find two drawbacks. First, in PPM, the overhead is proportional to the number of hops, the average of which is mostly less than 20 in the Internet. However, since the number of hops of cluster interconnects is often larger than 20, the overhead increases too much to apply PPM to cluster interconnects. Similarly, in DPM, ambiguity increases when the number of hops is longer than 16, i.e. at the point where marking bits is overwritten by other routers. Second, the previous algorithms assume that routing path is stable. However, the direct networks often use adaptive routing schemes in which routing paths are not fixed but variable depending on the current network condition. Therefore, we propose a new method, Deterministic Distance Packet Marking (DDPM), which finds a source directly without identifying paths. DDPM records only the relative distance from current position to source node. This can be easily implemented in regular networks such as Mesh, torus, and hypercube where each node is indexed by its regular topology. The victim needs only one packet to identify the source.

The remainder of the paper is organized as follows: Section 2 introduces the key idea of packet marking scheme and describes the PPM and DPM. We review the interconnection networks and explain their distinct properties in section 3. In section 4, we explain basic assumptions of our approach and show how the existing packet marking algorithms can be applied in the direct networks. Section 5 describes our algorithm and shows pseudo codes. Finally we discuss our algorithm's limitation and the future research in section 6, and conclude in section 7.

2. Related Work

Researchers have proposed various schemes against DDoS attacks in the Internet. Ferguson and Senie proposed an ingress filtering scheme which blocks all packets with bogus source addresses [10]. This scheme is possible only when a filtering router is aware of all legitimate source addresses. It is effective to block DDoS attacks in small networks because routers are aware of all source IP addresses. However, in large networks it is impossible to have all the IP information. Burch and Cheswick proposed a controlled flooding method, which can identify the DDoS attack path by selectively flooding incoming links [2]. Their idea is based on the fact that flooding a link DDoS traffic will change the amount of DDoS traffic noticeably. This approach is possible only during ongoing attacks. Also, it cannot find the paths when the attack traffic comes from many links. In addition, it can further worsen the situation by flooding more traffic into the already congested networks.

After Burch and Cheswick noted the possibility of tracing back source nodes by packet marking, Savage was the first to fully explore the probabilistic packet marking scheme by the help of intermediate routers [19]. The Probabilistic Packet Marking (PPM) scheme chooses a packet probabilistically at every edge and writes traceback information in the Marking Field (MF). The information consists of an edge identifier and distance from the current edge to the victim. A straightforward edge identifier can be two 32-bit IP addresses of two neighboring routers. For example, if a traffic flows through three routers, 165.91.133.25, 128.194.1.26, and 128.194.103.177, the marking information from the first edge will be (165.91.133.25, 128.194.1.26, 1), and the second will be (128.194.1.26, 128.194.103.177, 0). When the destination receives sufficient marking information, it can reconstruct the whole path of the traffic. This is very simple but two 32-bit marking information and distance field is too long to fit into the 16-bit MF. To store sufficient trace back information in the 16-bit IP identification field, they proposed an encoding scheme which hashes IP addresses and writes a fraction of it. With less packet length overhead, the expected number of packets for the victim to receive before reconstructing a path of length of d is roughly less than $k \cdot ln(kd)/p(1-p)^{d-1}$, where k is the number of fraction and p is the marking probability.

The primary drawback of the PPM is that it is not robust to distributed attacks. To identify the distributed attacks, it has to receive too many packets to analyze the whole traffic effectively. Another problem is vulnerability to the compromised router. To solve these problems, Song and Perrig proposed an advanced and authenticated marking scheme [17]. With an assumption that a victim has a complete router map, it can trace back by receiving less than one eighth of the packets than the PPM scheme, with robustness to the compromised routers.

In contrast to PPM, Yarr *et al.* proposed Deterministic Packet Marking (DPM) scheme in which each edge marks every packet. DPM writes just one bit in the MF, where the position is decided by its TTL. To decrease the probability that different paths have the same MF, DPM hashes two IP addresses of neighboring routers and records the last bit. All the edges down the path write their bits sequentially because TTL is decreased by one at every hop. So, the MF value of a distinct path is almost unique. If we assume the route is stable, all the traffic coming from the same source will have the same MF value. The victim can block all following traffic with that marking value, without additional computing complexity.

3. Interconnection Networks

Interconnection networks are classified into four categories: *shared-medium networks*, *direct networks* (Router-based networks), *indirect networks* (Switch-based networks), and *hybrid networks*. In shared-medium networks, devices communicate using a common shared medium such as a *bus* or a *token ring*. Alternative to the shared medium, a point-to-point link can be used between two neighboring devices. Those interconnects are called as direct networks. Typical examples are *Mesh*, *torus*, and *hypercube*. On the contrary, indirect networks connect devices via one or more switches. Crossbar and Multistage Interconnection Networks (MIN) are examples of these networks. Multiple backbone buses and cluster-based networks are examples of hybrid networks.

The popular direct networks are n-dimensional mesh, torus, and hypercube. All networks can be modeled as a graph, with processors as nodes and wires between processors as edges. *Degree* is the maximum number of edge incidents on any node. *Diameter* is the largest number of edge in the shortest path between any pair of nodes.

An n-dimensional mesh has $k_0 \times k_1 \times \ldots \times k_{n-1}$ nodes. k_i is the number of nodes in ith dimension. $(x_0,x_1,\ldots,x_{n-1})$ and $(y_0,y_1,\ldots,y_{n-1})$ are coordinates of node X and Y respectively. X and Y are neighboring if and only if the two indexes are same except only one dimension such that $x_i = y_i \pm 1$, where $0 \le i \le n-1$. The degree and the diameter of n-dimensional mesh is $2n$ and $\sum k_i - n$, where $0 \le i \le n-1$ respectively. In Figure 1 (a), the network's degree is four and its diameter six. Torus or k-ary n-cube is similar to n-dimensional mesh. The only difference is that two nodes, X and Y are neighboring if and only if the two coordinates are the same except only one dimension such that $x_i = (y_i \pm 1) \bmod k$. The modular arithmetic puts additional wraparound channels into n-dimensional mesh in Figure 1 (b), making k-ary n-cube. Its degree is $2n$ and diameter is $\sum k_i/2$ if k_i is even. Another popular direct network is the hypercube. An n-cube hypercube, also referred to as an n-cube hypercube (Figure 1 (c)), is an n-dimensional mesh where $k_i=2$ for $0 \le i \le n-1$. Its degree and diameter is n.

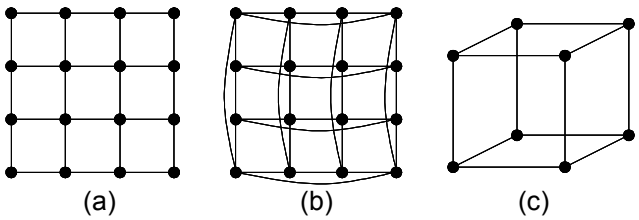


Figure 1. (a) 2-dimensional mesh, (b) 4-ary 2-cube, and c) 3-cube hypercube.

Routing algorithms decide the path each packet takes to arrive at its destination. Since routing algorithms in the Internet are different from those in the interconnection networks, we provide a brief introduction. If a routing algorithm decides on only one path between the source and the destination nodes, it is called deterministic. However, in adaptive routing, packets can move through different paths. In other words, an adaptive routing can give flexible routing paths depending on the current condition of the network. Depending on the adaptivity, an algorithm is called partially or fully adaptive.

For example, in Figure 2, we have drawn three routing paths on a 2-dimensional (4×4) mesh, each of which uses deterministic routing (XY routing), partially adaptive routing (west-first routing), and fully adaptive routing respectively. XY routing forwards packets along rows first and then along columns later. Just one turn is allowed. In (a), packets from S1 arrives at D by moving along the third row and then moving up along the third column. Packets from S2 move along the first row and then move down along the third column until they reach D. In (b), there are two small blocks on the right side of sources, meaning that those links failed for some reasons. In this condition, XY routing cannot forward any packets because it cannot use the right-side links first. However, west-first routing can forward packets successfully as in (b). West-first routing forwards packets west first, if

necessary, and then forwards east, south and north adaptively. In (b), since D is located east of sources and the east links fail, packets from S1 and S2 move south first (Packets from S1 can move north first) and then move toward D until they arrive in D. In (c), since a lot of links fail, there are a few paths from sources to the destination. Note that all paths should turn west at the right side node of D. West-first routing cannot route in this situation because packets should turn west at the last turn, not first. Fully adaptive routing does not have such restrictions, so it can forward all the packets successfully as shown Figure 2 (c).

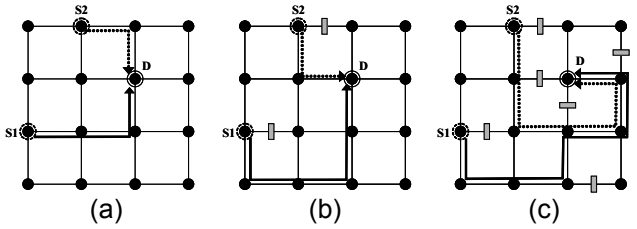


Figure 2. Routing algorithms in 2-dim mesh: (a) XY routing; (b) west-first routing; (c) fully adaptive routing.

4. Simple Marking Algorithms

In this section we describe basic assumptions and present how the packet marking algorithms described above can be applied to some direct networks. We show that because of routing adaptivity, existing marking algorithms cannot defend against DDoS attacks efficiently.

4.1. Basic Assumptions

Before applying the packet marking algorithms to direct networks, we identify assumptions that motivate and constrain our application:

- attacking traffic is generated and consumed inside direct networks,

- direct networks use IP,

- attackers generate packets with spoofed IP addresses,

- one node consists of a switch and a computing node, but they are separate entities,

- switches cannot be compromised, and

- the route is not stable.

The first assumption narrows down our research scope. Since we assume that IDS or firewall can keep cluster networks safe from outside security threats, we concentrate on the inside traffic of the network.

The second assumption allows us to apply packet marking algorithms in the cluster interconnects because the MF is located in the IP header. In most processor-level interconnection networks, they don't have to use IP address. However, in many cluster-level networks, to be connected to the Internet, they should use IP address. Even though only a front-end system uses a real IP address and other systems use private IP addresses, each IP address should be unique inside the network. In addition to the IP address, these machines may have additional indexes because they are connected in a regular way. After establishing a mapping table between IP addresses and indexes, switches look for this index alone. But every packet still contains IP header. Therefore, we can feasibly use the IP header for storing marking information. For the same reason, we can assume that an attacker can spoof the IP address.

Until now, we have considered the switch and computing node as one entity (Even though direct networks are router-based networks, we use *switch* rather than *router* to differentiate from the Internet router). But in a real scenario, they are separate entities. It is possible to integrate switching functions into its computing node by installing multiple Network Interface Cards (NIC). The primary issue of this interconnection is that it cannot get a high performance. For a packet to be switched, it should traverse each system's NIC. In a large cluster with a few thousand systems, total traversing distance will be at least hundreds of meters because a packet may have to traverse hundreds of systems. Alternatively, one can put together all switches in one rack, or even in a board. Such interconnection is easer to manage and has very low latency. Thus separation is more preferable and we henceforth hold that assumption.

Switches provide very limited service and switches are separate from computing nodes. This makes them very less unlikely to be compromised. To prevent even the small probability of compromising switch, we should add an authentication function working on the switching layer. Before putting this function into a switch, rigorous research is required to consider a trade-off between performance and security. We will discuss more about this in the discussion section. For now, we assume switches are not compromised.

Finally, a route from an attacker to a victim is not stable due to the adaptive routing. As we mentioned earlier, adaptive routings allow each packet to select the next link according to the current network state. A route is guaranteed to be stable only in deterministic routing.

Moreover, many adaptive routing algorithms allow a packet to revisit the same node. To prevent livelock caused by revisiting, adaptive routing algorithms on the direct networks provide livelock avoidance (or, recovery) schemes.

4.2. Probabilistic Packet Marking on the Direct Networks

In a probabilistic edge sampling, each switch randomly selects a packet, writes its own index and sets the distance to zero. When the next switch finds out a zero in the distance field, it writes its index next to the previous switch's index, and then increments the distance. All switches should perform one of two operations: random marking or deterministic incrementing. Figure 3 (a) shows an example of the simple marking scheme in a 4×4 mesh with deterministic routing. 1110 receives the following four different MFs, (0001, 0011, 3), (0011, 0010, 2), (0010, 0110, 1), and (0110, 1110, 0) from 0001. It receives (0101, 0111, 2), (0111, 0110, 1), and (0110, 1110, 0) from 0101. It is not ambiguous to reconstruct two distinct paths. The 16-bit MF is sufficient to store two indexes and one distance field because the length of two indexes is 2*log*16 bits and the length of distance is *log*8 bits. Total number of bits is 11, which is smaller than 16-bit MF. However, this scheme is not feasible in large networks (Table 1).

Table 1. Scalability of simple PPM

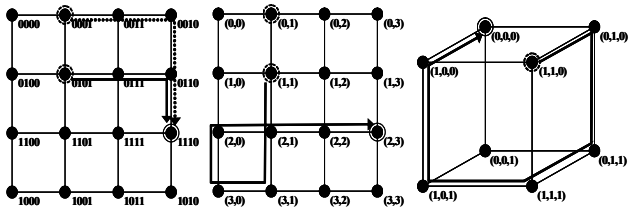
Topology	Required Field	Max Cluster Size
$n \times n$ mesh, torus	$logn^2 + logn^2 + log2n$	8×8 nodes
n-cube hypercube	$2log2^n + loglog2^n$	2^6 nodes

In an eXclusive-OR (XOR) scheme, instead of storing two indexes of neighboring nodes, switches write an XOR value of two nodes' indexes. While this scheme is not feasible in the Internet due to the long IP address, this is not practical in the direct networks due to the ambiguity in reconstructing attack paths. Since there is only one bit difference between neighboring nodes, the XOR value always has only one bit set to one, and other bits are set to zero. In an $n \times n$ mesh, during reconstruction, one XOR value is mapped into average $n(n\text{-}1)/logn$ edges because there are $n(n\text{-}1)$ edges in the mesh. Consequently, as the mesh size increases, the ambiguity also increases. Any encoding method decreasing the length of the edge identification field will end up increasing the reconstruction ambiguity. This holds true for other direct networks, too.

To remove the ambiguity, we can use bit difference position. Instead of storing two neighboring indexes or only the XOR value, this scheme stores one index and a bit difference position as well as distance. When the least significant bit is different, we say the 0th bit is different, and similarly when the most significant bit is different, $loglogn^2$th bit is different. For example, in Figure 3 (a), 1110 receives (0001, 1, 3), (0011, 0, 2), (0010, 2, 1), (0110, 2, 0) from 0001 and (0101, 1, 2), (0111, 0, 1), (0110, 2, 0) from 0101. However, this scheme is also not suitable for large networks because the maximum size is not as large as current clusters with thousands nodes (Table 2).

Table 2. Scalability of Simple Bit Difference PPM

Topology	Required Field	Max Cluster Size
$n \times n$ mesh, torus	$logn^2 + loglogn^2 + log2n$	16×16 nodes
n-cube hypercube	$2log2^n + loglog2n + loglog2n$	2^8 nodes



(a) (b) (c)

Figure 3. Simple and Deterministic Distance Packet Marking

It would be possible to store the edge information in the IP additional option. This is only possible when we can manipulate the IP option in a packet that is in flight. To do this, switches would have to check the IP option of every packet and then write marking information in the appropriate position. This large overhead is not preferable to high performance clusters.

Even though we can manage to use the IP option effectively, PPM is not applicable due to traffic overhead and reconstruction ambiguity. The expected overhead for the victim to reconstruct an attack path of length d is less than $ln(d)/p(1\text{-}p)^{d-1}$, where p is the marking probability [19]. In case of $n \times n$ mesh and $n \times n$ torus, diameter is $2n$-2 and n respectively. In a middle size cluster with a mesh of about 1024 nodes, the diameter is 62. This is far larger than average hops, around 15, in the Internet [1, 3, 14]. Long distance incurs large traffic overhead on the victim. However, the above analysis is possible only under the assumption that a route is stable. Depending on the

network's state and the adaptivity of the routing, packets with the same source and the same destination may take very different paths. This makes it impossible to use any other probabilistic packet marking approaches including PPM.

4.3. Deterministic Packet Marking on Direct Networks

In contrast to PPM, in DPM, every switch should mark all the packets. Rather than writing an index value or an XOR value, every switch writes the last bit of the hash value of the switch index. The marking position is decided by TTL mod 16. Definitely, a path with longer than 16 hops will get marking bits overwritten by other switches located farther than 16 hops away from the source. If we use the node index for the hash value, 1110 will receive 11000 from 0001 and 1100 from 0101 in Figure 3 (a). Therefore, if we detect that both traffic are DDoS attacks, we can block all traffic having 11000 or 1100 in the MF.

After the 16th hop, the MF starts to lose information of paths farther than 16 hops. Therefore, it can not find source effectively in networks with diameter larger than 16, which is very relatively small. In addition, this scheme happens to be ambiguous. On an average, two out of four neighbors in the 2-D mesh have the same last bit. It is highly probable to trace back non-attacking sources. Considering the adaptive routing, the ambiguity becomes much larger. This is because one attack may have different MF values and different length and thus DPM is not suitable for the direct networks.

5. Deterministic Distance Packet Marking

All the previous marking schemes try to record all path information that packets traverse. That is why they are not robust to adaptive routing. To solve this problem, we propose Deterministic Distance Packet Marking (DDPM) scheme. This marking scheme records only relative distance from the current position of a packet to the source.

For recording a relative distance in each dimension, we use the indexing scheme that we mentioned in section 2. For an n-dimensional mesh, we can define a distance from X $(x_0,x_1,...,x_{n-1})$ to Y $(y_0,y_1,...,y_{n-1})$ as vector V $(v_0,v_1,...,v_{n-1})$ where each $v_i = y_i - x_i$. This vector is the coordinate difference, so there is a unique distance vector between any two nodes in the network. This means that a distance vector V arriving at node Y identifies a unique source node, X. For each packet, V is set to a zero vector when the packet first enters a switch from a computing node. After the switch decides the next channel to forward the packet, the switch computes a distance from the next node to the current node. The new distance vector is calculated by adding original distance and the distance to next node. After the switch stores the new distance in the MF, the switch forwards the packet. The full algorithm is described in Figure 4. *Routing*() is a function that returns the next node coordinate. *Extract_MF*() returns a distance value from the IP header, and *Store_MF*() stores it in the header.

Regardless of the routing algorithm used, the final distance vector V should be the exact difference from the source to the destination for a packet to arrive a destination. Therefore, the destination can identify the source directly even though it does not know intermediate routes.

In the hypercube algorithm, each d_i denotes whether the value of dimension i of current node is the same with that of the source node. That is, if d_i is 0, the values of the ith dimension of current node and source node are the same and vice versa. The only difference is that it uses XOR rather than addition and subtraction. In the hypercube, a switch toggles just one dimension at each hop, so V' is always one bit different from V. Therefore, DDPM is applicable to hypercube interconnections.

Algorithm: Deterministic Distance Packet Marking for Direct Networks

Inputs: Destination D $(d_0,d_1,...,d_{n-1})$
Current X $(x_0,x_1,...,x_{n-1})$,
Output: Next Y $(y_0,y_1,...,y_{n-1})$
Source S $(s_0,s_1,...,s_{n-1})$
Variables: Distance V $(v_0,v_1,...,v_{n-1})$
New distance V' $(v'_0,v'_1,...,v'_{n-1})$
Difference Δ $(\delta_0,\delta_1,...,\delta_{n-1})$

Procedure:

```
if X = D then
        V := Extract_MF();
        S := X – V;
endif
Y := Routing(V);
V:= Extract_MF();
Δ := Y – X;
V' := V + Δ;
Store_MF(V');
```

Figure 4. Deterministic Distance Packet Marking

For example, Figure 3 (b) shows a packet traverses 2-D mesh adaptively from (1,) and (2,3). The distance vector changes as following: (1,0), (2,0), (2,-1), (1,-1), (1,0), (1,1), and (1,2). When (2,3) node receives the distance vector (1,2), it can subtract (1,2) from (2,3) and

quickly identify the source (1,1). In the hypercube in Figure 3 (c), the distance vector changes as following: (1,0,0), (1,0,1), (0,0,1), (0,1,1), (0,1,0), and (1,1,0). (0,0,0) can identify the source (1,1,0) by XORing its coordinate (0,0,0) and the distance vector (1,1,0).

DDPM can support larger number of nodes compared to the simple marking algorithms. To support $n \times n$ 2-D mesh and torus, each half of the MF contains the distance in one dimension. The distance can be negative, so half of MF can represent 2^7 nodes in one dimension. Therefore, DDPM can mark up to 128×128 mesh and torus (16384 nodes cluster). For a 3-D mesh and torus, DDPM can mark $16 \times 16 \times 32$ nodes by splitting the MF into two five-bits and one six-bits (8192 nodes cluster). For the hypercube, the whole MF can be used for the distance vector, so DDPM can mark 16-cube hypercube (65536 nodes cluster). Table 3 summarizes these facts.

Table 3. Scalability of DDPM

Topology	Required Field	Max Cluster Size
$n \times n$ mesh, torus	$2\log n - 2$	128×128 nodes
n-cube hypercube	$\log 2^n$	2^{16} nodes

6. Discussions

In this section, we discuss our algorithms' limitations and future research. First, we discuss DDoS detection and its difficulties. Second, we explain the importance of research on relationship between performance and security. Finally, we describe future research on diverse cluster interconnects.

6.1. Detection

Before identifying sources, we have to decide which traffic is malicious. That is because to identify all sources of entire traffic will be useless if we cannot differentiate malicious traffic from normal traffic. Therefore, in this paper, we assumed there exists an efficient DDoS detection method in cluster interconnects. In the Internet, it is possible to filter or wiretap all traffic because there are some points at which the Internet is connecting or most traffic aggregates. Firewalls or IDS can analyze all the traffic and find DDoS attacks.

In the cluster networks, however, it is not easy to filter or wiretap all traffic. The first reason is that gathering all traffic is topologically difficult. Nodes and switches can be distributed and traffic does not aggregate at several points. The second reason is that a cluster generally has a high speed network, making real time analysis difficult. For the source identification to be effective, detection should be fast enough to block or trace back the source node in real-time. To solve these problems, one can consider to find a minimal set of trusted switches for detection and identification, which requires more extensive research.

6.2. Performance vs. Security

There is a trade-off between system performance and security. If we put more functions on switches, cluster interconnects would be more secure. For example, switches can block packets with spoofed IP addresses by looking up a mapping table. However, it will increase the processing time of switch, and hence overall degrade the system performance to some extent. In our approach, a switch performs only simple functions such as addition, subtraction, and XOR, so we expect they would not affect overall performance. Hence, we need to find the relationship between performance degradation and security functions as the future research.

6.3. Network Topology

Our approach is limited to direct networks. A lot of cluster systems employ indirect networks or hybrid networks. Since the properties of the networks are different, a new approach may be necessary to solve the source identification problem in such networks. Moreover, hybrid networks and irregular networks do not have a universal regularity and it may need a completely different approach. Hence, rigorous research on various network topologies is necessary.

7. Conclusion

In this paper, we have argued that DDoS attacks inside the cluster will be a serious problem in near future. We have shown that two existing marking algorithms, Probabilistic Packet Marking and Deterministic Packet Marking, cannot trace back the source efficiently due to the routing adaptivity in the direct networks. Instead, we have proposed a new packet marking scheme, Deterministic Distance Packet Marking, which records only distance from the victim to the source, thus identifying the source directly. Our technique is robust to routing algorithms and applicable to mesh, torus, and hypercube. Moreover, our approach can mark very large networks and does not incur much overhead on switches or the victim, making it highly scalable. Finally, we have discussed some considerations such as attack detection, trade-off between performance and security, and multiple topologies.

8. References

[1] H. Burch and B. Cheswick, "Internet watch: Mapping the Internet," *Computer*, 32(4):97–98, Apr. 1999.

[2] H. Burch and B. Cheswick, "Tracing Anonymous Packets to Their Approximate Source," Unpublished paper, Dec. 1999.

[3] Caida. Skitter. http://www.caida.org/tools/measurement/-skitter/ , 2000.

[4] "CERT/CC, CERT Advisory CA-2001-26 Nimda Worm," http://www.cert.org/advisories/CA-2001-26.html, Sept. 2001.

[5] "Computer emergency response team, cert advisory ca-2000-01: Denial-of-service developments," http://www.cert.org/ advisories/ CA-2000-01. html, 2000.

[6] Kay Connelly and Andrew A. Chien, "Breaking the Barriers: High Performance Security for HighPerformance Computing," in *New Security Paradigms Workshop '02*, 2000.

[7] Sven Dietrich, Neil Long, and David Dittrich, "Analyzing distributed denial of service attack tools: The shaft case," in *14th Systems Administration Conference, LISA 2000*, 2000.

[8] Rossen Dimitrov and Matthew Gleeson, "Challenges and New Technologies for Addressing Security in High Performance Distributed Environments," in *Proceedings of the 21st National Information Systems Security Conference* 457-468.

[9] Dave Dittrich, "Distributed Denial of Service (DDoS) attacks/tools resource," http://staff.washington.edu/ittrich/ misc/ddos/, 2000.

[10] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing," Internet RFC 2267, Jan. 1998.

[11] L. Garber, "Denial-of-Service Attack Rip the Internet," *Computer*, April 2000.

[12] K. Houle and G. Weaver, "Trends in Denial of Service Attack Technology," http://www.cert.org/, October 2001.

[13] Foster Ian, Nicholas Karonis, Carl Kesselman and Steven Tuecke, "Managing Security in High- Performance Distributed Computations," *Cluster Computing*, Vol 1, issue 1, pages 95-107, 1998.

[14] Internet mapping. http://research.lumeta.com/ches/map/, 2002.

[15] A. Machie, J. Roculan, R. Russell, and M. V. Velzen, "Nimda Worm Analysis," Tech. Rep., Incident Analysis, SecurityFocus, Sept. 2001.

[16] D. Song, R. Malan, and R. Stone, "A Snapshot of Global Internet Worm Activity," Tech. Rep., Arbor Networks, Nov.2001.

[17] Dawn X. Song and Adrian Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," in *Proceedings of INFOCOM* 2001, 2001.

[18] R. Russell and A. Machie, "Code Red II Worm," Tech. Rep., Incident Analysis, SecurityFocus, Aug. 2001.

[19] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson, "Practical Network Support for IP Traceback," in *Proceedings of SIGCOMM 2000*, 2000.

[20] Abraham Yaar, Adrian Perrig, and Dawn Song, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," in *Proceedings of the IEEE Symposium on Security and Privacy 2003*, 2003

[21] "Attackers infiltrating supercomputer networks," http:// news.com.com/2100-7349-191024.html?tag=sas.email, 2004.