Collaborative Detection of DDoS Attacks over Multiple Network Domains

Yu Chen, *Member IEEE*, Kai Hwang, *Fellow IEEE*, and Wei-Shinn Ku, *Member, IEEE*

Abstract— This paper presents a new distributed approach to detecting DDoS (*distributed denial of services*) flooding attacks at the traffic flow level. The new defense system is suitable for efficient implementation over the core networks operated by *Internet service providers* (ISP). At the early stage of a DDoS attack, some traffic fluctuations are detectable at Internet routers or at gateways of edge networks. We develop a *distributed change-point detection* (DCD) architecture using *change aggregation trees* (CAT). The idea is to detect abrupt traffic changes across multiple network domains at the earliest time. Early detection of DDoS attacks minimizes the flooding damages to the victim systems serviced by the provider.

The system is built over attack-transit routers, which work together cooperatively. Each ISP domain has a CAT server to aggregate the flooding alerts reported by the routers. CAT domain servers collaborate among themselves to make the final decision. To resolve policy conflicts at different ISP domains, a new *secure infrastructure protocol* (SIP) is developed to establish the mutual trust or consensus. We simulated the DCD system up to 16 network domains on the DETER testbed, a 220-node PC cluster for Internet emulation experiments at USC Information Science Institute. Experimental results show that 4 network domains are sufficient to yield a 98% detection accuracy with only 1% false-postive alarms. Based on a 2006 Internet report on AS (*autonomous system*) domain distribution, we prove that this DDoS defense systrem can scale well to cover 84 AS domains. This security coverage is wide enough to safeguard most ISP core networks from real-life DDoS flooding attacks.

Index Terms— Cyber defense, network security, DDoS attacks, and Internet technology

1 Introduction

Today's defense systems against *distributed denial of services* (DDoS) attacks are mostly built on detecting flooding consequences rather than the causes of the traffic surges [17, 19, 31, 39]. Flooding consequence is manifested by congestions on communication links [28], overflow in half-open SYN queue, or imbalance between the input/output traffic on the gateways of edge networks [42]. Unfortunately, the damage has been done when the flooding consequence is observed. Thus, it is highly desirable to detect DDoS attacks at the earliest possible time, instead of waiting for the flood to become widespread [7, 8].

A comprehensive solution to DDoS attacks requires to cover the global effects over a wide area of *autonomous system* (AS) domains on the Internet [3, 39]. Obviously, the global-scale defense is too costly for a real-life implementation. Even the DETER testbed [4] can only emulate partial Internet activities. To implement an efficient defense system, we must leverage the network topology and use distributed traffic monitoring and detection. In reality, we build a DDoS defense system over a limited number of network domains serviced by the same *Internet service provider* (ISP). These ISP network domains cover the edge networks where the protected systems are physically connected.

In the sequel, we consider each AS a single network domain such as the core network of an ISP. According to an ISO 2006 Report [21] on AS resource allocations, there are 34,998 AS domains globally. Dimitropoulos et al. [12] identified that 67.7% of the AS domains belong to companies, organizations, or universities that run their own local-area networks, 30.3% are ISP controlled domains, the remaining 2% are Internet exchange points or network information centers.

Our DDoS defense is targeted for implementation in ISP core network domains [2]. Majority ISPs do not share their AS domains with competitors. Therefore, they are unlikely to take part in collaborative DDoS defense. However, AS domains serviced by the same ISP or owneds by the same company or organization can combat the DDoS attacks, collectively. This covers 98% of the Internet AS domains.

At the early stage of a DDoS attack, the traffic changes are difficult to detect because low traffic fluctuations are not observable. Monitoring the Internet traffic at individual flow level is cost prohibitive to cover all possible flows. Meanwhile, the global traffic in wide-area network is tremendously large to perform real-time detection of network anomalies, effectively.

In practice, it is possible to convince a small percentage, say 25% of the ISP-controlled network domains to join collective effort in combating DDoS attacks. This amounts a few hundreds of domains to form a consortium in collective DDoS defense. We will prove in Section 5.1 that it would be sufficient to involve only tens of domains to work together in coping with most DDoS flooding attacks. This defense range is certainly

- *Y. Chen is with the Department of Electrical and Computer engineering, State University of New York –Binghamton, Binghamton, NY 13902. E-mail: ychen@binghamton.edu.*
- *K. Hwang and W. Ku are with the Departments of Electrical Engineering and Computer Science, University of Southern California, Los Angles, 90089. Email: kaihwang@usc.edu, wku@usc.edu.*

Manuscript received August 14, 2006; revised Dec. 23, 2006; accepted April 10, 2007; published online June 2007. Recommended for acceptance by S. Olariu.

within the coverage of a single ISP or of a few ISPs which collaborate with each other.

To be cost-effective, we propose to monitor the traffic at a *superflow* level. A superflow contains all packets destined for the same network domain from all possible source IP addresses and applies various protocols such as TCP or UDP, etc. This detection level covers the aggregate from individual traffic flows. All packets of a superflow have the same prefix IP address of the same destination domain [15]. Motivated by using lightweight detection with low complexity [3, 9, 26, 42], we propose a *distributed change-point detection* (DCD) architecture using a new mechanism, called *change aggregation tree* (CAT). The concept of CAT was first presented in the IEEE *CTS-2006* Conference [11].

This CAT mechanism is designed at the router level for detecting abrupt changes in traffic flows. When a DDoS attack is launched, the routers observe changes in spatiotemporal distribution of traffic volumes. The domain server uses the router-reported traffic surge reports to construct the CAT tree. Usually, these changes in traffic flows present a directionality homing towards the victim system. Random fluctuations incurred with legitimate traffic flows do not present the homing effects. To benefit our readers, Table 1 summarizes the basic parameters and abbreviations used in this paper.

TABLE 1.
Notations and Abbreviations used in The Paper

Symbol	Definition	Abbr.	Meaning
α	Traffic inertia factor	ATR	Attack transit router
β	Router detection threshold	CAT	Change aggregation tree
θ	CAT detection threshold	SIP	Secure infrastructure protocol
R_d	DDoS detection rate	ISP	Internet service provider
R_{fp}	False-positive alarm rate	DCD	Distri. change-point detection
DR	Traffic deviation ratio in the I/O ports of a router	DETER	Cyber Defense Technology Experimental Research

Our DCD approach is unique and offers the very first attempt to explore distributed change-point detection over collaborative network domains. We detect the starting of DDoS flooding attacks by monitoring abnormal traffic flows. This monitory and detection is performed from router to router as the CAT tree is dynamically constructed on the fly. On the DETER testbed, we implemented the detection scheme from 4 to 16 AS domains. We carried out intensive experiments to evaluate the DCD scheme. The performance results demonstrate high detection accuracy and low false-positive alarms.

The rest of this paper is organized as follows: Section 2 briefly reviews related works. Section 3 presents the principle of the change-point detection method and the algorithms for raising attacking alerts by individual routers. Section 4 explains the CAT tree construction within a single network domain. The inter-domain change detection algorithm is presented in section 5 along with a new *secure infrastructure protocol* (SIP). Section 6 reports the DETER experiments setups and performance results. Section 7 discusses scalability issues and deployment limitations. Finally, we conclude with a summary of contributions and discuss further research needed towards eventual use of the defense system.

2 Related Work and Our Contributions

DDoS attacks often come with widespread worms [6]. The attacker often exploits the huge resource asymmetry between the Internet and the victim systems [39]. The flooding traffic is large enough to crash the victim machine by communication buffer overflow, disk exhaustion, or connection link saturation, etc. Figure 1 shows a flooding attack launched from 4 zombies. The *attack-transit routers* (ATRs) detect abnormal surge of traffic at their I/O ports. The victim is attached with the end router R0 in Fig.1. All the attack flows form the superflow homing towards the end router.

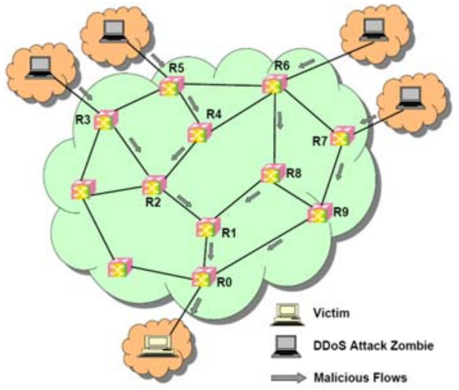


Fig.1. A traffic superflow by DDoS flooding attacks launched from a large number of Zombies towards a common victim host.

A plethora of DDoS defense and response mechanisms have been suggested in the past, including IP traceback [1, 3, 17], packet filtering [26], and flood pushback [20]. More sophisticated intrusion detection systems [19, 32] and DDoS defense schemes [10, 24, 31, 41] have been recently proposed. Researchers have attempted to combat repeated DDoS attacks [18]. Others use overlay networks [43], DDoS-resilient scheduling [36] and trust-negotiation [37] approaches to establishing trust.

MULTOPS [16] and D-WARD [29] suggested filtering and rate limiting on suspicious flows at the source end. The security managers often focus on protecting their own networks and choose a local detection approaches [7]. For instance, the COSSACK [33] and DefCOM [29] deploy detectors at the victim side and send alert to filter or to rate limiter located at the source side. Chen and Song [9] proposed a perimeter-based scheme for ISP to enable anti-DDoS services to their customers. Their scheme relies on edge routers to identify the sources of the flood of attacking packets.

Many researchers use change-point detection theory to detect abnormal Internet traffic caused by DDoS attacks [5, 11, 34, 42]. Lacking accurate statistics to describe the pre-change and post-change traffic distributions, a nonparametric CUSUM scheme was developed for its low computational complexity [5]. The scheme monitors the short-term behavior shifting from a long-term behavior. Once the cumulative difference reaches certain threshold, an attack alert is raised. Shin and associates [42] have suggested a centralized DDoS defense scheme to monitor the change points at the gateway level. Peng et al. [34] took a similar approach to monitoring the source IP addresses.

In this paper, we propose a new distributed aggregation scheme based on change-point detection across multiple network domains. This scheme is extended from the single-domain change-detection scheme reported in [11]. We establish the cooperation among communicating network domains. This enables the building of an early warning system for DDoS defense across multiple ISP domains. Our DCD scheme is capable of tracing back automatically, once the detection is successfully carried our. The global CAT tree detects the network anomalies incurred on the fly.

In summary, our contributions are highlighted below in four technical aspects. The details and proofs are given in subsequent sections:

(a). Traffic anomaly detection at superflow level:

Monitoring Internet traffic at routers on individual flows is identified by a 5-tuple: {*source IP, destination IP, source port, destination port, protocol applied*}. The superflow consists of those traffic flows destined for the same network domain and applied the same protocol. This level of traffic monitoring and anomaly detection is more cost-effective for DDoS defense in real-life Internet environments.

***(b). Distributed change-point detection*:**

Considering the directionality and homing effects of a DDoS flooding attack, we propose to use collaborative routers for distributed change-point detection and use the domain servers for alert correlation and aggregation.

***(c). Hierarchical alerts and detection decision making*:**

Our system adopts a hierarchical architecture at the router and domain levels. This simplifies the alert correlation and global detection procedures and enables the DCD system implementation in ISP networks.

(d). Novelty of SIP (secure infrastructure protocol):

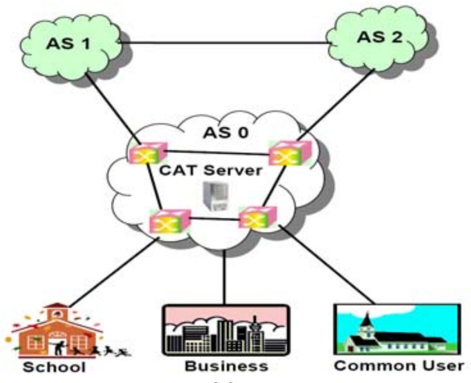
We propose a new trust-negotiating SIP protocol to secure inter-server communications. The SIP has removed some of the shortcomings of the existing IPSec and application-layer multicasting protocols [25, 43]. SIP appeals for implementation on VPN tunnels or over an overlay network built on top of all domain servers.

3. Distributed Change-Point Detection

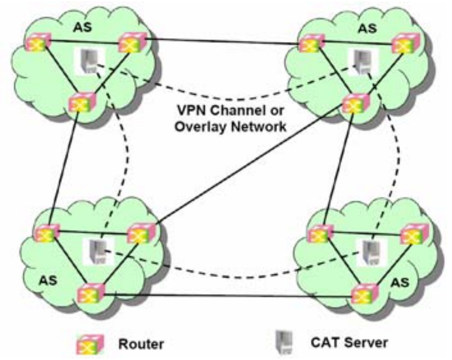
The DCD scheme detects DDoS flooding attacks by monitoring the propagation patterns of abrupt traffic changes at distributed network points. Once a sufficiently large CAT tree is constructed to exceed a preset threshold, an attack is declared. This section presents the principles behind the DCD system. We focus on traffic pattern change detection at the router level.

3.1 The DCD System Architecture

Figure 2 presents the system architecture of the DCD scheme. The system is deployed over multiple AS domains. There is a central CAT server in each domain. The system detects traffic changes, checks flow propagation patterns, aggregates suspicious alerts, and merge CAT subtrees from collaborative servers into a global CAT tree. The root of the global CAT tree is at the victim end. Each tree node corresponds to an ATR. Each tree edge corresponds to a link between the attack-transit routers.



(a)



(b)

Fig.2. Distributed change detection of DDoS attacks over multiple AS domains. (a) Multi-domain DDoS defense system; (b) Inter-domain communication via VPN tunnels or overlay network atop the CAT servers in 4 domains.

Our system has a hierarchical detection architecture. There are three layers in this architecture. At the lowest layer, individual router functions as a sensor to monitor local traffic fluctuations. A change-point detection program (Algorithm 1) is executed on each router. Considering the directionality and homing effects in a DDoS flooding attack, routers check how the wavefront changes. A router raises an alert and reports an anomalous traffic pattern to the CAT server.

The second layer is at each network domain level. The CAT server constructs a CAT subtree according to alerts collected. The subtree displays a spatiotemporal vision of the attack superflow in the domain. At the highest layer, the CAT servers at different domains form an overlay network. For security precaution, they communicate with each other through *virtual private network* (VPN) channels.

All CAT servers send their locally-generated CAT subtrees to the edge server in the destination domain, where the victim is attached. By merging CAT subtrees from cooperative domains, the destination server has a global picture of the attack. The larger is the global CAT tree so constructed, the higher is the threat experienced.

The CAT detection scheme does not need to specify an absolute threshold on traffic volume. The detection is done by checking the number of nodes (routers) raising the alerts from the CAT subtree. Figure 3 illustrates how a CAT subtree rooted at the end router is constructed by merging the alert reports from 9 ATRs. The upstream and downstream ATRs report to the CAT server during each monitory cycle.

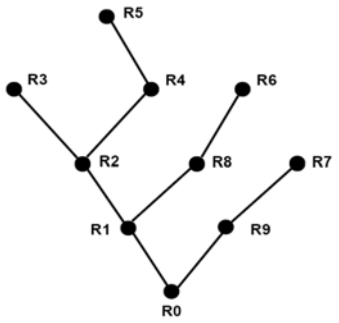


Fig.3. Construction of a change aggregation tree for the flooding pattern reported by 9 attack-transit routers in Fig.1, where the victim host is connected to the end router R0

Using Algorithm 2, the server constructs a CAT rooted at the end router R0. The server recursively scans through all upstream routers to construct the tree. The CAT presents a traffic-flow tree pattern rooted at the router connected to the edge network, where the victim is attached. With sufficient exchange of alert information from related domains, the system can detect the DDoS flooding attack at very early launching stage, before the attacking flows hit the victim network.

The flow-level detection can distinguish among several DDoS attacks. We monitor the traffic change based on the homing effects of the victim address. Each CAT tree is uniquely constructed for flooding streams towards the same destination in an edge network. When multiple DDoS attacks are launched concurrently against multiple victims, there are multiple CAT trees to be constructed and they are completely distinguishable. The shape of the CAT tree corresponds to th attacking traffic paths.

Surely the attacker can randomly choose zombies in an attack. In addition, the group of zombies can be changed dynamically during the attack. However, the random selection of zombies will not impact our detection results, because the CAT tree is constructed on the fly. Essentially, different distribution of zombies results in different CAT tree. Since the detection criterion is not the shape but the size of the CAT tree, changing zombie distribution will not weaken our detection capability.

3.2 Principles of Change-Point Detection

In change-detection problems, if pre-change and post-change distributions are known, the CUSUM statistic has been suggested to solve the problem [5]. We adopt a non-parametric approach for its simplicity. Let t_1, t_2, …, t_m be discrete time instants and x(t_m, i) be the number of packets received by a router during time slot m at port i. The historical estimate of the average number of packets is defined iteratively by:

$$\overline{X}(t_m,i) = (1-\alpha)\cdot\overline{X}(t_{m-1},i) + \alpha\cdot x(t_m,i) \qquad \textbf{(1)}$$

where 0 < α < 1 is an inertia factor showing the sensitivity of the long-term average behavior to the current traffic variation. Higher α implies more dependence on the current variation. We define below $\boldsymbol{S_{in}(t_m, i)}$ as the deviation of input traffic from the average at time slot $\boldsymbol{t_m}$.

$$S_{in}(t_m,i) = \max\{0, S_{in}(t_{m-1},i) + x(t_m,i) - \overline{X}(t_m,i)\} \qquad \textbf{(2)}$$

The subscript in indicates that this is the statistics of the incoming traffic. While a DDoS flooding attack is launched, the cumulative deviation is noticeably higher than the random fluctuations. Since $\boldsymbol{S_{in}(t_m, i)}$ is sensitive to the changes in the average of the monitored traffic [5], we measure the abnormal deviation from historical average as follows. Let the *deviation from average* (DFA) be the indicator of such an attack. The incoming traffic DFA is defined below at port $\boldsymbol{I}$ at time $\boldsymbol{t_m}$.

$$DFA_{in}(t_m,i) = S_{in}(t_m,i)\big/\overline{X}(t_m,i) \qquad \textbf{(3)}$$

If the DFA exceeds a router threshold β, the measured traffic surge is considered a suspicious attack. The threshold β measures the magnitude of traffic surge over the average traffic value. This parameter is preset based on previous router use experience. In a monitoring window of 100 ms to 1 s, a normal superflow is rather smooth due to statistical multiplexing of all independent flows heading for the same destination [22]. If there is no DDoS attack, we expect a small deviation rate far below β. In general, we work in the range $2 \le \beta \le 5$.

For outgoing traffic, we define $\boldsymbol{y(t_m, i)}$ as the number of packets at time tm leaving at port $\boldsymbol{i}$ and be the historical average of departed packets. Similarly, we have:

$$\overline{Y}(t_m,i) = (1-\alpha)\cdot\overline{Y}(t_{m-1},i) + \alpha\cdot y(t_m,i) \qquad \textbf{(4)}$$

$$S_{out}(t_m,i) = \max\{0, S_{out}(t_{m-1},i) + y(t_m,i) - \overline{Y}(t_m,i)\} \qquad \textbf{(5)}$$

The above equations will be used to specify the change-detection algorithms in subsequent sections.

3.3 Traffic Surge Detection in Routers

Each router monitors traffic variation and counts the packet number within a monitory window at each I/O port. We use the term *traffic pattern* to refer to the combination of traffic surges at all I/O ports of a router. In general, a router with m-input ports and n-output ports may have 2^{m+n} possible traffic patterns. The height of the black boxes in Fig.4 signifies the magnitude of traffic volume at I/O links. The raised block height indicates a surge detected and the lower boxes for normal traffic.

All packets of a superflow must be homing towards the same destination network. Before entering the destination domain, the flow paths present a converging tree pattern. Only at the destination, the superflow scatters packets towards a particular edge network. There exist 16 possible traffic patterns from a 2 x 2 router. For simplicity, we illustrate in Fig.4 only 4 basic traffic patterns at a 2 x 2 router with $m = n = 2$. The remaining 12 traffic patterns can be specified, similarly.

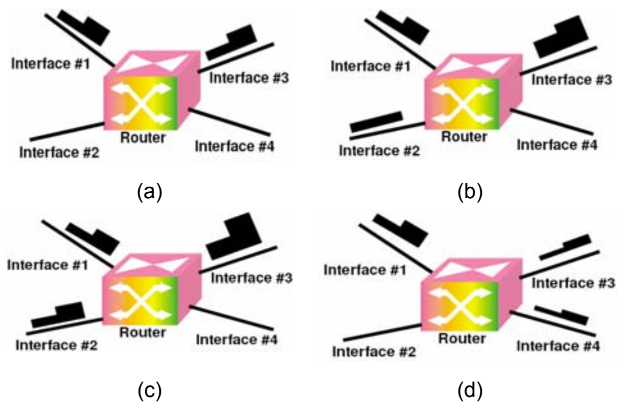


Fig.4. Four basic patterns of traffic changes at a 2 x 2 router I/O ports:(a) Flow through.(b) Partial aggregation. (c) Full aggregation. (d) Scatter pattern

a. ***Flow-through pattern:*** This traffic pattern is shown in Fig.4(a). The router forwards the traffic flow from an input port to a selected output port without subdividing or diverting the traffic to other ports.

b. ***Partial aggregation pattern:*** All the incoming flows are merged at one outgoing port iout, not all incoming flows contain traffic surges as shown in Fig. 4(b).

c. ***Full aggregation pattern:***The outgoing flow merges multiple incoming flows, all containing traffic surges exceeding the threshold β. This router is considered a merge point on the attacking path (Fig.4(c)).

d. ***Scatter pattern:*** The incoming flow scatters at this router. This is not part of a DDoS attack (Fig. 4(d)). This pattern is observed in the destination domain

Another statistical parameter, *deviation ratio* (DR), is defined below to measure the ratio of incoming packets from port i_{in} have propagated to output port i_{out}. DR is the ratio of traffic deviations between I/O ports.

$$DR(i_{in}, i_{out}) = S_{out}(t_m, i_{out}) / S_{in}(t_m, i_{in}) \qquad \textbf{(6)}$$

If DR > 1, the router amplifies the input deviation. This corresponds to a full surge of traffic volume. DR ≈ 1 implies the router merely plays the role of a forwarder. This phenomenon is observed in the partial surge at one input port. The case of DR < 1 indicates that the incoming wave is scattered to multiple ports. It is not part of the convergence traffic of DDoS attacks. Therefore, by checking the DR value, a router determines whether the pattern is part of the traffic from a DDoS attack.

When a router detects that a DFA_{in} exceeds the deviation threshold β, it calculates the deviation rate between the outgoing and incoming ports. If DR is close to one, the traffic aggregation pattern is considered suspicious. The router generates an alert message and reports the pattern to the CAT server. Otherwise, the router sends a regular status message indicating no anomaly observed.

Below is a pseudo code of Algorithm 1 for local change detection at the router level.

```
Algorithm 1: Traffic surge detection at router level
Input:  x(t, i) and y(t, i): Incoming and outgoing packets at
        time t and port I, respectively
        $\overline{X}(t_{m-1}, i)$ : Historical average of packet arrivals
                up to time m-1 at port i
        $\overline{Y}(t_{m-1}, i)$ : Historical average of outgoing packets
                up to time m-1 at port i
        Router detection threshold $\beta$ based on past experience
Output: Alert messages sent to the central CAT server.
Procedure:
01: Update historical average of I/O packets in a flow
02: Calculate $DFA_{in}$ using Eq. (3)
03: If $DFA_{in} \ge \beta$ Then  Calculate DR using Eq. (7)
04:     If DR ≈ 1.0 Then  Suspicious pattern detected.
            Send out an alert message to CAT server.
05:   Else  Nothing suspicious.
            Send out a regular status message to CAT server.
```

Algorithm 1 demands light-weigh computing power at ATRs. For an *m*x*n* ATR at an intermediate node of the CAT subtree, there are 2^{m+n} combinations of traffic surge patterns at the I/O ports. For each superflow, the router needs to calculate the DR at most m times. Even if a large DFA value is detected at an input port, the router uses only one output port to release the traffic surge. This will simplify the routing decision at the ATR significantly.

At the victim domain, the end router decision is more complicated. If two or more attacks are launched concurrently towards the same destination domain, the traffic surges will pass through the end router at multiple ports. For an *m*x*n* end router, the worse case is that m input surges scatter to n output ports. The end router calculates the DR *mn* times. This burden is lowered by splitting the superflow to multiple destination addresses.

4. Constructing Subtrees at Domain Servers

This section describes the CAT subtree construction at each CAT server in a single network domain. Different subtrees are generated in multiple network domains. The global CAT tree is generated by merging all subtrees. While the flooding traffic merges at the victim end, the routers along the paths capture suspicious traffic patterns.

The router reports the identifier of a superflow causing the traffic surge. Since all routers are under the same ISP authority and work cooperatively, each router knows their immediate neighbors. Using the reported status information, the domain server detects the traffic flood based on the CAT tree constructed.

The alert message provides the upstream and downstream router identifiers. Since all routers are under the same authority and they work cooperatively, each router knows their immediate neighbors. The alert message provides information for CAT server to include the routers in the CAT subtree. The main purpose of sendingt then flow status message reports where the suspicious flows are captured.

To indicate the location of a suspicious flow, the router identifier must send. We need to identify the superflow identifier of the *n*-bit prefix of the destination IP addresses. To construct the CAT, the status report provides the upstream and downstream router identifiers instead of router I/O port numbers. Using the reported status information, the domain server constructs the CAT tree gradually after receiving the alert reports from the ATRs.

Table 2 summarizes the information carried in a typical alert message from an ATR. The output of Algorithm 2 is a single-domain CAT subtree similar to the one shown in Fig.3. The CAT tree is specified by a hierarchical data structure. The root node carries the superflow ID, the number of routers involved, root node ID, and the count of child nodes at the next level.

TABLE 2.
Alert Message Reported by a Router

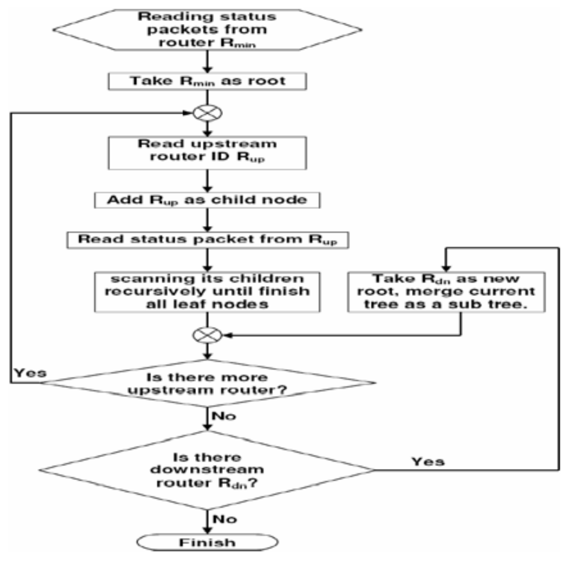
Parameter	Brief Definition
nd_id	The router ID,
fl_id	The superflow ID
up_num	Number of upstream nodes
dn_num	Number of downstream nodes
up_id	node ID of upstream node
dn_id	node ID of downstream node
Router status	Suspicious attack or normal traffic

Algorithm 2: CAT Subtree Construction in a Single Domain Server
Input: Traffic alert messages received from all routers In the same AS domain
Output: A data structure describing the CAT subtree constructed in this domain

Procedure:
```
01: Read all suspicious patterns in and arrange them according to router ID
02: Start from the suspicious node with minimum ID $R_{min}$
03: root ← $R_{min}$
04: read the upstream node number up_num
05: read the downstream node number dn_num
06: node_number ← node_number + up_num - 1
07: While up_num > 0
08:     Read in one upstream node $R_{up}$
09:     Add $R_{up}$ as a leaf node
10:     scan through its upstream nodes
11:     up_num ← up_num – 1
12: End While
13: While dn_num = 1
14:     Read the downstream node $R_{dn}$;
15:     root ← $R_{dn}$
16:     node_number ← node_number + 1
17:     Scan through other upstream nodes of new root;
18:     dn_num ← dn_num of the new root
19: End While
```

To clarify the control flow, this construction process is specified by a flowchart in Fig. 5. The next level lists the pair of information {L1 node ID, count of children at next level L2}. This process continues until reaching the leave nodes of the tree. The CAT subtree is sent to the CAT server of the destination domain. In Algorithm 2, the domain server constructs the CAT subtree based on collected status reports from the routers. Routers detected no attacks are not involved in the tree construction.

Starting from the node R_{min} with a minimum ID in Fig.5, the CAT server takes it as the root node. The server scans through upstream child nodes identified by up_id. This descendent search is performed iteratively until the leaf nodes are reached. If there is a downstream router R_{dn}, we take router R_{dn} as the new root and repeat the procedure. Meanwhile, the descendent search procedure is repeated for all upstream routers of root R_{dn}. Then we check the downstream router of Rdn and repeat the procedure until the downstream router is out of the domain boundary.


Reading status packets from router R_{min}
Take R_{min} as root
Read upstream router ID R_{up}
Add R_{up} as child node
Read status packet from R_{up}
scanning its children recursively until finish all leaf nodes
Take R_{dn} as new root, merge current tree as a sub tree.
Yes
Is there more upstream router?
No
Is there downstream router R_{dn}?
Yes
No
Finish
Fig.5. Control flow in Algorithm 2 to construct a CAT subtree.

5. Merging Traffic in Multiple Domains

This section describes the extension of the single-domain detection scheme to work on multiple network domains. First, we analyze the complexity of the global CAT tree growth in real-life Internet domains. Then we present the mechanisms for cross-domain attack detection. In addition, we introduce a new protocol that supports inter-domain communications, trust negotiation, and collaborative detection.

5.1 CAT Growth with Domain Distribution

The complexity of the CAT growth is analyzed below based on Internet topology data available from open literature [21, 38]. Figure 6 illustrates the process of the CAT tree growth out of merging subtrees from attack-transit domains. Let ***r*** be the number of hops from an AS domain to the destination domain. The server checks the received subtrees in increasing order of distance ***r***.

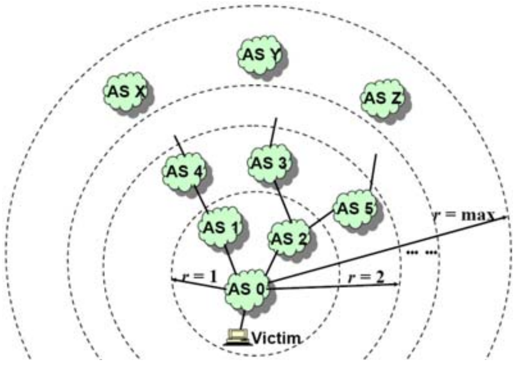


Fig.6 Merging CAT subtrees from neighboring AS domains to outer domains to build a global CAT tree, where AS_0 is the victim domain and $\boldsymbol{r_{max}}$ = 6 hops

The system first merges the subtrees from ASs located in 1-hop (***r*** = 1) distance to form a partial global tree. Next, it merges the subtrees from domains at 2-hop distance. The merging process repeats with distances ***r*** = 3, 4 until all subtrees are merged into the final global CAT tree. We analyze below the complexity of global CAT tree growth at intra-domain and inter-domain levels. The routers monitor traffic conditions and report anomalies to their domain CAT server, periodically. The local setting parameters α and *β* affect the size of the local CAT subtrees constructed.

Given a domain consisting of *N* routers, the number of alerts that CAT server receives is proportional to *N*. The threshold used in CAT subtrees construction (Algorithm 2) equals to the number of alerts received by the final CAT server. Therefore, the detection time is estimated by O(*N*) within each domain. Of course, different domain sizes (*N*) may requie a variable subtree generation time.

At the inter-domain level, the complexity of global CAT tree merging is highly dependent on the network topology. We model the Internet domains as an undirected graph of *M* nodes and *E* edges. The diameter of the graph is denoted by *δ*.

Siganos et al. [38] models the Internet neighborhood as an *H*-dimensional sphere with a diameter ***δ***. The parameter *H* is the dimension of the network topology [14]. For example, *H* = 1 specifies a ring topology and *H* = 2 for a 2-dimensional mesh. Any two nodes are within an effective diameter, $\boldsymbol{\delta_{ef}}$ hops away from each other. Faloutsos estimated the magnitude of $\boldsymbol{\delta_{ef}}$ by the following expression:

$$\delta_{ef} = \left(\frac{M^2}{M + 2E}\right)^{1/H} \qquad \textbf{(7)}$$

In 2002, the dimension of Internet was calculated as *H* = 5.7 in an average sense. The ceiling of this diameter $\boldsymbol{\delta_{ef}}$ is thus set to be 6. Let *NN*(***h***) be the number of domains located at distance ***h*** from a typical domain in the Internet. Table 3 gives the domain distribution – the probability of an AS domain residing exactly ***h*** hops away from a reference domain. The numbers of domains in various distance ranges are given in the second row. It is interesting to note that most communicating domains are within 3 or 4 hops, almost following a normal distribution centered arouind an average hop count of 3.5.

TABLE 3.
Internet Domain Distribution Reported in Feb. 28, 2006 [21]

Hop Count, ***h***	1	2	3	4	5	6	≥ 7
Domain Distribution,$\boldsymbol{p_h}$	0.04%	8.05%	38.55%	38.12%	12.7%	2.25%	0.29%
Domain Count, *NN*(*h*)	14	2,818	13,493	13,342	4,445	788	102

The number of Internet AS domains keeps increasing in time, the Faloutsos reports [14, 38] indicates that this AS distribution is pretty stable over time. This implies that a packet can reach almost all domains in the Internet by traversing through 6 hops. Therefore, we set the maximum hop count $\boldsymbol{r_{max}}$ = 6 in Fig.6.

Let $\boldsymbol{p_h}$ be the probability of having an AS domain located at distance ***h*** from the reference domain Therefore, the average number of domains used to build a global CAT tree is upper bounded by:

$$T = \sum_{h=1}^{r_{\max}} NN(h) \times p_h \qquad \textbf{(8)}$$

Substituting the entries in Table 2 into Eq.(9), we obtain the expected domain count *T* = 14×0.004 + 2818×0.0805 + 13493×0.3855 + 13342×0.3812 + 4445×0.127 +788×0.0225+102×0.0029=11,097 domains used in average Internet applications. This domain count posts a loose upper bound on the expected number of ISP domains involved in building a global CAT tree.

In reality, only a few ISP-controlled AS domains may commit to defend DDoS attacks, collaboratively. On the conservative side, consider 5% of ISP AS domains are committed. Thus the above upper bound could be reduced to only 168 ISP domains, provided that they conform to the domain distribution in Table 3.

5.2 Global Tree Construction at Victim End

In a DDoS flooding attack, the attacker often recruits many zombies distributed over the Internet. The flooding traffic may travel through multiple AS domains before reaching the edge network, where the victim is physically attached. Routers at the upstream domains observe the suspicious traffic flows earlier than routers at the downstream networks.

Our DCP detection system was designed to have strong collaborations among all domain servers along the superflow paths. Algorithm 3 specifies the merge of CAT subtrees for detecting DDoS attacks across multiple network domains. The CAT subtrees constructed at all traversed domains must be merged to yield a global CAT tree at the destination domain.

Algorithm 3: Global CAT tree construction and detection decision

Input: CAT subtree reports from participating domain servers, the server detection threshold θ.

Output: The global CAT tree over multiple AS domains. Raise the alert for an imminent DDoS attack.

Procedure:

```
01: Construct the local CAT sub-tree (Algorithm 2) periodically
02: Receiving sub-trees from other CAT servers
03: If local subtree exists, Then  Check the superflow ID,
04:         If this domain is the destination domain, Then  Set distance $r = 1$
05:         Merge subtrees from domains at distance $r$ to the current global tree
06:         $r \leftarrow r+1$
07:             While { there are un-checked sub-trees }, generate the CAT profile
08:                 If CAT profile ≥ $\theta$ Then  DDoS attack is detected and raise an alert
09:         Else   Check the root router position
10:             If root router is connected to other domain
11:             Then Sent the global CAT tree to the destination domain server
12: Else  Raise an attack alert based on the global tree merged
```

The final declaration of a DDoS attack is the result of threshold detection using the global CAT tree. Not only the victim network launches appropriate counter-measures, but also some traceback actions are to be taken by all attack-transit routers along the superflow paths. The actions include dropping of suspicious packets or rate limiting against the flows.

The global CAT tree corresponds to the flooding attack flows. The leaf nodes are directly related to the zombies used. The height of the global CAT tree corresponds to the superflow hop count. Some highly distributed attacks may recruit hundreds of zombies, the global CAT tree may cover a wide area on the Internet. Therefore, we use the global CAT tree profile θ as a global detection threshold. The CAT profile indicates how many routers observed abnormal traffic surges. Thus θ is an integer bounded by the number of ATRs in a domain.

The tree width and height thus reveal the scope of the DDoS attack. Through experiments on DETER testbed, we obtain the global detection threshold value by training from some attack datasets. Theses threshold values have yielded the highest detection rate and lowest false positive rate during the training period.

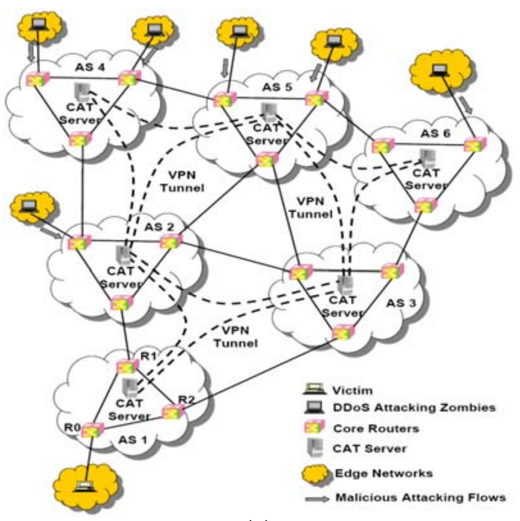
On receiving subtrees from upstream CAT servers, the CAT server in the destination domain builds the global CAT tree from its local subtree. Once the global CAT tree is formed, the server compares the CAT profile with the global detection threshold θ to decide on a DDoS attack. An alert is raised and necessary countermeasure is triggered, accordingly. Figure 7 shows an example network environment involving six AS domains. The victim system is located in the AS1 domain. Zombies are scattered widely in Internet outside the illustrated domains. By detecting abnormal traffic changes in each domain, the CAT server creates a CAT subtree locally at each domain using Algorithm 2.

Figure 7(b) shows three steps taken to merge the 6 subtrees generated by 6 CAT servers of 6 AS domain. All 6 subtrees are resulted from checking the packets belonging to the same superflow traffic destined for the same domain AS1. Five subtrees generated at AS 2, AS3, AS4, AS5, and AS6 at upstream domains are sent to AS1 at Step 2. Then, the concatenated CAT subtrees are connected to the downstream subtree at AS1. Thus the global CAT tree is finally rooted at the last hop router to an edge network R0 that is attached to the victim system.

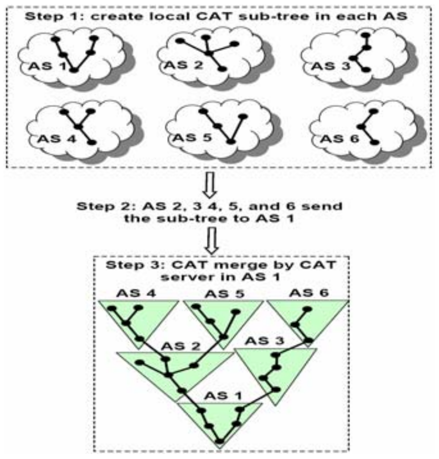
5.3 Secure Infrastructure Protocol (SIP)

To support global CAT tree construction across multiple domains, we propose a new *secure infrastructure protocol* (SIP) protocol by extending from ICMP [35]. The SIP is designed as an integral part of IPv4 standard in a layer closer to the physical network. To provide a secure information exchange platform, the protocol supports three levels of communication as illustrated in Fig.8.

The lowest level enables routers in the same domain to share information in status monitoring. The second level supports communication between routers and CAT server in each domain. Routers periodically report local traffic detection results to the domain server. The highest level supports inter-domain communication among the CAT servers in trust negotiation to resolve conflicts in security polices in different domains.



(a)


Step 1: create local CAT sub-tree in each AS
AS 1
AS 2
AS 3
AS 4
AS 5
AS 6
Step 2: AS 2, 3 4, 5, and 6 send the sub-tree to AS 1
Step 3: CAT merge by CAT server in AS 1
AS 4
AS 5
AS 6
AS 2
AS 3
AS 1

(b)

Fig. 7. An example 6-domain global CAT tree construction environment. (a) DCD system architecture over 6 domaion. (b) Mergeing 6 CAT subtrees to yield a global CAT tree

Due to privacy and security concerns, ISPs are reluctant to reveal inside information to competitors, such as topology, bandwidth configuration, or capabilities. Hence, aside from managing the information exchange, servers are also in charge of trust negotiation. We adopted the trust negotiation of multilateral security suggested by Ryutov et al. [37].

Using their *Generic Authorization and Access-control* (GAA) interface, SIP can help establish trust among AS domains and enter a collaborative DDoS defense system agreement. By trust negotiating, ISPs can determine how much private information is allowed to be shared with others.

When a domain server enters a collaborative defense agreement with another domain server, the administrator sets up the trust levels required. Each server needs to perform a trust negotiation with a peer server only once with each new domain joining. The trust level determines how much sensitive information can be disclosed when exchanging detected anomalies information.

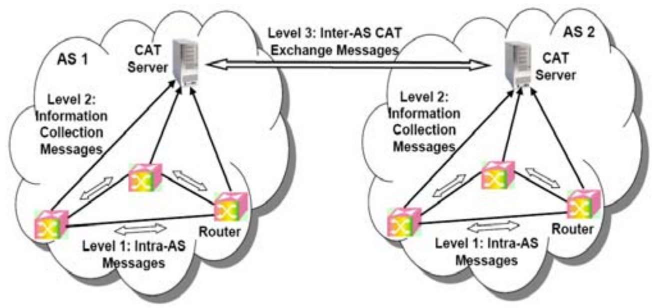

AS 1
CAT Server
Level 3: Inter-AS CAT Exchange Messages
AS 2
CAT Server
Level 2: Information Collection Messages
Level 2: Information Collection Messages
Router
Router
Level 1: Intra-AS Messages
Level 1: Intra-AS Messages

Fig.8 Three communication levels in using the SIP (secure infrastructure protocol) between two CAT servers in two AS domains.

The SIP is designed with three trust levels : *full trust* (FT), *basic trust* (BT), and *no trust* (NT). With full trust, the SIP server will provide all necessary information that describes the characteristics of the detected anomaly. With basic trust, the SIP server only sends some statistics containing no private and sensitive data. When no trust is set, no cooperation takes place between the CAT servers.

Existing security protocols like IPSec and ICMP focus on information security through strong cryptography to avoid eavesdropping, strict access control to block illegal access, and using digital signatures, etc. The ICMP is used to control error, when a network device or host requires to report an error in datagram processing [35].

In contrast, SIP is designed to monitoring the status of network infrastructure, such as link bandwidth utility, fluctuation of traffic, etc. Although both IPsec and SIP are implemented at the IP layer, the SIP is totally deployed inside the intermediate network and does not need support from the end hosts.

6. Results from DETER Experiments

We verify the performance of the newly proposed DDoS detection scheme with network attack experiments on a 220-node DETER testbed [4] at USC Information Sciences Institute. The experimental settings and performance results are reported below.

6.1 Settings on DETER Testbed

To evaluate the performance of the CAT-based DDoS detection system, we performed experiments using variant network topology, background traffic, and attack generation. We adopt the real-world ISP topologies downloaded from the Rocketfuel Project at University of Washington [2]. The DETER testbed is essentially an Internet emulator built over a cluster of PCs to perform a broad range of network security experiments.

We report below the DETER results on 2, 4, 8, and 16 collaborative domains. Each domain typically has 7 to 12 routers. Figure 9 shows the DETER cluster configuration built at USC Information Science Institute. Due to limited 220 nodes in DETER, we choose the smallest ISP configuration topology from the Rocketfuel dataset.

In the DETER testbed experiments, 34 routers were simulated over 4 AS domains. The link bandwidth among the network domains was set at 100 MB/s. To generate the background traffic closer to reality, we use the OC48 trace dataset from the CAIDA project [30] to regenerate

Internet traces using the Harpoon traffic generator [40]. To generate DDoS attacks, we use the toolkit Stacheldraht (version 4.0) [13]. Stacheldraht generates the ICMP, UDP, TCP SYN flooding and Smurf attacks.

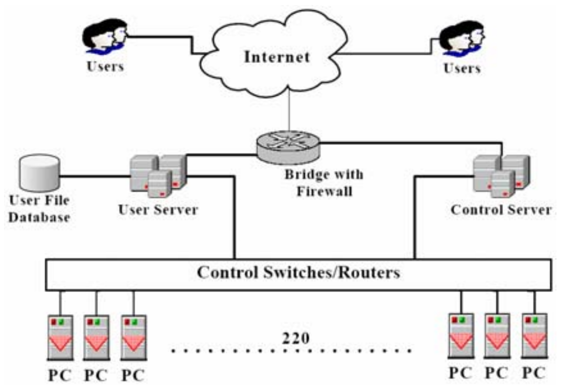


Fig.9 Multi-PC cluster architecture of the DETER Testbed at USC Information Science Institute.

6.2 Performance Metrics Used

The performance of our DCD detection scheme is evaluated with three metrics: *detection rate, false-positive alarms,* and *system overhead.* All the metrics are measured under different DDoS attacks using TCP, UDP, and ICMP protocols. The *detection rate* R_d of DDoS attacks is defined by the following raio:

$$R_d = a / n \qquad (9)$$

where a is the number of DDoS attacks detected in the simulation experiments and n is the total number of attacks generated by the Stacheldraht toolkit during the experiments.

In addition, we are interested in the performance of our DCD scheme under normal traffic without DDoS attacks. A traffic superflow is called a *false-poisitive alarm*, if an attack is detected out of normal traffic without attcks. Let p be the number of false positive alarms raised by the CAT server and m be the total number of normal traffic flow events checked by the simulator. Therefore, the ratio p/m defines the *false positive alarm rate*:

$$R_{fp} = p / m \qquad (10)$$

The ROC (*receiver operating characteristic*) curve shows the tradeoff between the detection rate and false-positive rate. Section 6.4 reports the detection accuracy measured under different detection thresholds. Another critical issue is the time overhead to detect the launch of DDoS attacks. The average detection time measures from the start of a DDoS attack to the time of raising an alarm. The monitoring window should be chosen greater than this detection time.

6.3 DETER Experimental Results

To evaluate the effectiveness of the DCD detection scheme, we report the alerts raised at routers and analyze the CAT subtree properties. The DETER experiments choose the inertia factor a in the range (0.1, 0.5) and the router detection threshold β in the range (2, 5).

A. Alert Magnitude and Router Threshold

We used 34 routers in 4 AS domains in the DETER testbed. When the traffic flow exceeds the router threshold β, the router raises alerts. No alert will be raised, if no anomaly is detected at routers. Figure 10 illustrates the total numbers of alerts raised by the routers with SYN flooding compared with the case of no attack. More alerts are raised with attacks comparing with alerts from no attacks.

Two left bars of each group correspond to a = 0.1, the case of heavy dependence on the past average traffic to raise alert. The leftmost bar stays around 20 alerts, which is insensitive to increasing threshold β. The second (gray) bar reduces to 5 alerts as β increases. This implies that β = 3.5 is an ideal threshold to distinguish attacks from no attack. The two right bars (fabric and black) demonstrate a higher inertia value setting a = 0.3. The fabric bars with attacks are much higher than the black bars without attacks. For $a \geq 0.3$ and $\beta \geq 3.5$, the alert bars reduce to zero, meaning the attack is not detectable.

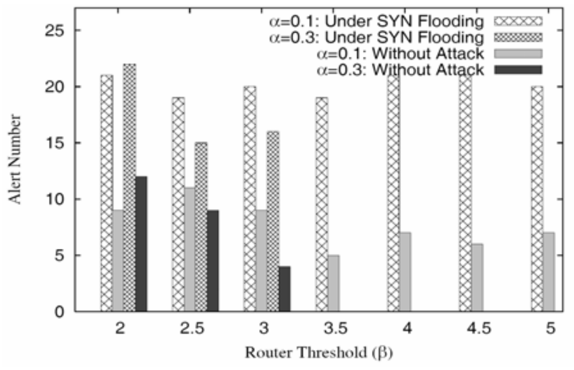


Fig.10. Total alerts plotted against the router threshold in DETER experiments with 34 routers in 4 AS domains using a 500 ms monitoring window.

B. The Global CAT Profiles

Figure 11 displays the global CAT tree profile, defined by the tree size or the number of routers that have raised the alert signals. We plot the tree profile the against the router detection threshold β. For small traffic inertia α = 0.1, the SYN flood has an average of 20 routers being alerted to yiels the CAT subtree. Without attack, this tree profile reduces to less than 5 nodes.

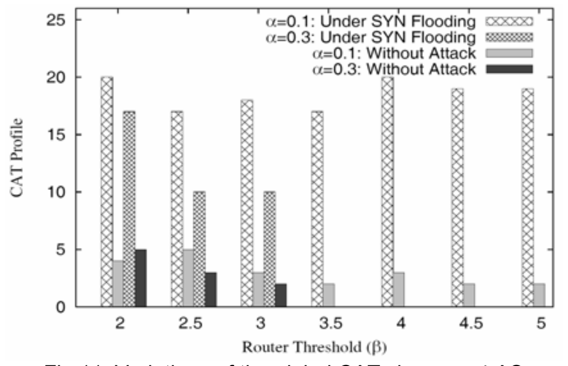


Fig.11. Variations of the global CAT size over 4 AS domains in 500 ms monitory window

With higher traffic inertia α = 0.3, the SYN attack results in tree profle with around 10 alerts. Without attack, the alert profile reduces to zero after the threshold β exceeds 3. Based on these results, we discover an optimal router threshold setting $\beta \geq 3.5$ with an inertia ratio α = 0.1. With 20 out of 34 routers launched alerts, the router utilization is 20/34 = 58%.

C. Effects of Monitoring Window Size

The size of the monitoring window affects the number of alerts raised in multiple AS domains. Through experiments on DETER testbed, we observed the optimal monitoring window size as 100 ms. The false positive alarm number increases steadily with increasing monitoring window size. However, the alert of real SYN attacks keeps about the same number for all monitoring window sizes as shown in Fig. 12.

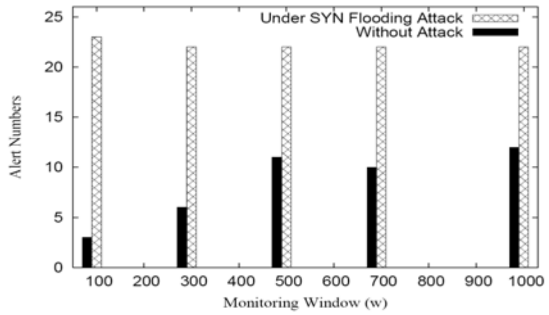


Fig.12. The router alert number with and without DDoS attacks monitored using various window size

In Stacheldraht [13], the UDP and ICMP packet rate for each zombie is adjustable through setting different UDP and ICMP packet sizes – the longer is the packet length, the lower is the packet rate. The TCP SYN attacks use fixed packet size of 64 bytes with a fixed packet rate. The maximum UDP and ICMP packet size is limited to 1024 bytes in Stacheldraht. We observed similar detection rate of TCP SYN and UDP/ICMP attacks with 128-byte packets.

6.4 Detection Accuracy and False Alarms

In this section, we report the detection accuracy of the DCD scheme under TCP SYN, UDP, and ICMP attacks with different packet rates. The reported results correspond to a = 0.1, β = 2.0, and w = 500 ms. The detection accuracy is reflected two factors – the detection rate and the false positive rate. In order to achieve high detection accuracy, we have to increase the detection rate and decrease the false positive rate. However, tradeoff exists between the two factors as discussed in the following paragraphs.

A. Detection Rate of DDoS Attacks

Figure 13 illustrates the variances of the detection rate (Eq.7) with respect to different server detection thresholds (θ). The TCP SYN attack has the highest detection rate which is close to 100% with $\theta \leq 12$. The low-rate UDP attacks have lower detection rate than TCP attacks.

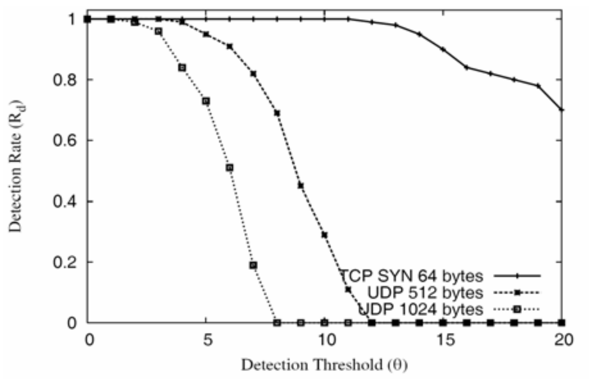


Fig.13. Effects of server threshold on the detection rate of 3 DDoS attack types.

For UDP attacks of 512-byte packets, the detection rate can be kept above 80% with $\theta \leq 9$. When the packet size increases to 1024 bytes, the detection rate drops to zero with θ ≥ 7. These results demonstrates that in order to maintain high detection rate on TCP and UDP SYN attacks, we need to set θ with a small value, such as θ = 5 and adjust the packet size to 1024 bytes.

B. False-Positive Alarms

Figure 14 shows the false positive alarm rate against the CAT server threshold θ. The number of alert generated by random fluctuation in normal traffic is small and negligible. With a server detection threshold θ = 4, the false positive rate drops to less than 1%. However, the real challenge lies in the fact that highly distributed attacks may use low packet rates to avoid from being detected [39]. Only after sufficient attack flows are merged, the deviation is detected by the routers. Hence, a small detection threshold value is required to achieve high detection accuracy with a low false positive rate.

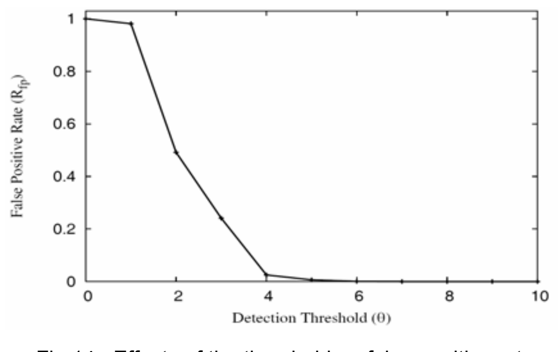


Fig.14 Effects of the threshold on false-positive rate in detecting TCP SYN attacks.

The ROC curve in Fig.15 explains the tradeoff between the detection rate and false positive rate under various attacks. Our detection scheme achieves a detection rate as high as 99% with less than 1% false positive rate for high-rate DDoS attacks. All three curves support this observation. Even for low-rate UDP attacks, our choice of low CAT threshold (θ) accomplishes a detection rate of 91% at a false-positive rate of 23%. This result proves the effectiveness of the DCD detection mechanism

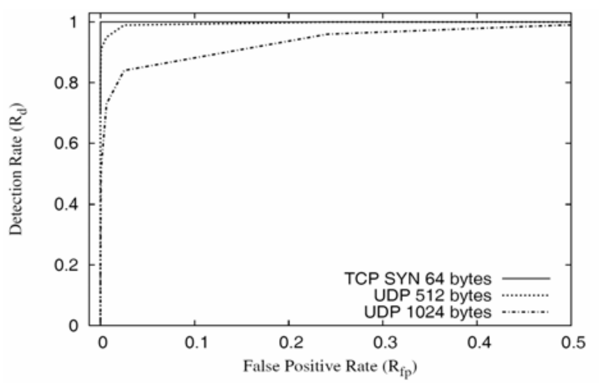


Fig.15 ROC curves showing the tradeoff between the detection rate and false-positive rate of DDoS attacks.

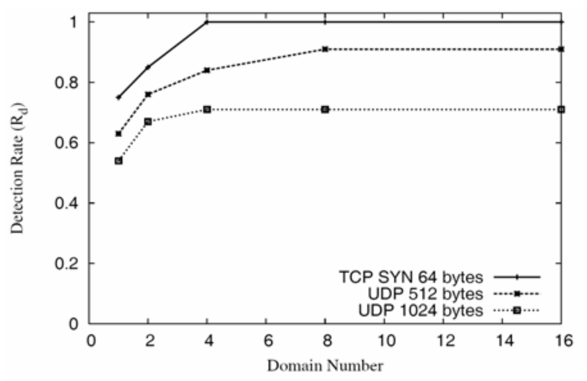


Fig.16 Scalability in using the DCD defense system for DDoS defense, where 4 AS domains are sufficient to yield 68% – 98% detection rate based on DETER simulation results

7. Scalability and Deployment Limitations

To deploy a distributed security scheme in ISP core networks, the scalability is related to the network size, domain number, data rate, link capacity, or router number involved. This section studies the scalability of the DCD scheme in terms of detection performance and system overhead experienced. Then we discuss the issues of flash crowd, security holes, and implementation limitations of the DCD system.

7.1 Domain Scalability in ISP Coverage

The study of domain scalability is driven by reducing the costs in system implementation and maintenance. One advantage of collaborative detection is its enlarged area of protection coverage. We have to use the CAT subtrees constructed by upstream domains to assess the earlier impact of a superflow generated by a DDoS attack. Even before the target network is overwhelmed, early warning can prevent catastrophic disasters.

Figure 16 plots the detection rates of three DDoS attack types against the number of domains used. Through experiments on DETER testbed, we studied the effectiveness of cross-domain cooperation up to 16 AS domains. The detection rate becomes saturated after a sufficient number of AS domains are involved. The results are obtained under the system settings: $\alpha = 0.1$, $\beta = 2.0$, $w = 500$ ms, and $\theta = 5$.

Recall, we simulated 8 to 10 routers per domain in scalable DETER experiments. With a small AS domain containing 8 routers, $\theta = 5$ implies that more than half of the routers generated alerts as the attacking flows aooroaching the root of the CAT tree. For 64-byte attacks, the optimal domain size is 4. For UDP 512-byte packets, the detection rate saturates at 8 domains. For UDP 1024-byte packets, four AS domains are sufficient to achieve the saturated detection rate.

The above analysis implies that 25% (4 out of 16) to 50% (8 out of 16) of participating network domains would be sufficient to detect a DDoS attack. Based on this level of domain participation, we find from Table 2 that the DCD system can scale well to cover T×30.3%×5%×(25% to 50%) = 42 to 84 domains in commercial ISP core networks. This result is rather encouraging in practical applications.

This number is manageable, considering the added monitoring burden of routers and the role of the CAT server in each AS domain. In reality, the decision process is ended much earlier by using a lower threshold θ. This leads to the conclusion that the DCD system scales well to cover wider area of flooding DDoS attacks.

7.2 Implementation and Limitations

It is a big challenge to discriminate DDoS attacks from fluctuation of legitimate traffic patterns, called *shrew attacks* [10] and *flash crowds* [23]. When a flash crowd happens, the CAT server creates a similar tree and could raise a false alarm. We suggest adding a few new features to separate the real DDoS attack traffic from the flash crowd.

The idea is to check newly appeared source IP addresses. For each superflow, in addition to traffic volume, we need to monitor the distribution of source IP addresses. Packet content matching offers another potential solution. However, this option is limited by packet payload being encrypted.

Compromised insiders are often the most difficult problem to solve in security control. Infected routers may hide suspicious traffic patterns or send false alarms to the CAT server. These false alarms can weaken the use of the CAT tree as a means of attack detection.

We can make the DCD system more robust by introducing a topology verification procedure. Knowing the network topology, the CAT server is capable of rectifying the falsely reported traffic patterns according to the upstream and downstream relationship. Single or limited number of Byzantine defectors could be blocked this way.

The CAT server in the destination domain merges all received CAT subtrees to make a global detection. The global CAT tree provides useful information for traceback or pushback. In the literature, packet marking offers another option to trace the path of attack flows [1, 20]. The victim launches traceback or pushback operations only after DDoS attack is fully detected. In contrast, our DCP system finishes the traceback task as soon as the merged CAT tree becomes sufficiently large. Our system pushes the defense line to upstream domains, where the traffic surges are first observed.

Internet AS resource distribution [21] suggests the use of up to 84 domains to cope with TCP SYN and UDP flooding attacks. This corresponds to a saturated detection rate (98%) and low false alarm rate (below 1%). If we lower to 65% detection rate with 15% false alarms, the server detection threshold (θ) can be further lowered. This implies that fewer AS domains could be used to make a decision. Based on our DETER experiments, it is sufficient to involve only 28 ISP domains in detecting the most of the known DDoS flooding attacks.

8. Conclusions and Further Research

It is crucial to detect the DDoS flooding attacks at their early launching stage before widespread damages done to legitimate applications on the victim system. We develop a distributed DDoS detection system based on a new CAT mechanism. In conclusion, we elaborate on potential impacts and applications of the system.

A. Detecting traffic changes at attack-transit routers:

Based on the anomaly pattern detected in related network domains, our scheme detects a DDoS attack before the victim is overwhelmed. This approach captures the abrupt traffic changes at attack-transit routers. The high detection rate of DDoS attacks is expected with very low false-positive alarms.

B. Scalable performance over multiple ISP domains:

Our DCD detection scheme is suitable for deployment at the ISP core networks. The provider-level cooperation eliminates the need of intervention by edge networks. Our FETER experimental results prove that 4 to 8 domains are sufficient to yield 98% detection rate of TCP SYN and UDP flooding attacks. Based on recently reported Internet AS domain distribution, we expect that the defense scheme to scale well to cover up to 84 ISP-controlled domains in a real-life Internet environment.

C. SIP protocol for inter-domain trust negotiation:

To support inter-AS collaboration, the SIP protocol is proposed to resolve policy conflicts and regulate alert message format. Our SIP protocol is part of USC/ISI effort in securing Internet infrastructure against DDoS or worm attacks [19, 37] that threaten the availability, reliability, and dependability of Internet services.

D. Valuable parameters from DETER experiments:

We have verified the effectiveness of the DCD scheme through emulation experiments on the DETER testbed. The engineering data on the traffic inertia factor α, router threshold β, global detection threshold θ, and monitory window size w are very useful design parameters for building real DCD prototype or production systems against DDoS attacks in the future.

Our distributed detection scheme automatically performs the traceback during the detection of suspicious traffic flows. Once a DDoS flooding attack is detected, we know the exact router or network domain that the anomaly was observed. In related projects, we have developed a spectral analysis method [10] to filter out *shrew* DDoS attacks with low attacking rate. Network worm containment helps the defense of DDoS attacks as reported in [6].

We suggest a hardware approach to implementing the CAT mechanism and the SIP protocol using network processors or reconfigurable FPGA devices. This may demand the integration of signature-based IDS with anomaly detection systems [19]. The ultimate goal is to promote real-time detection and response against DDoS attacks with automatic signature generation.

Acknowledgements: This work was supported by NSF ITR Grant ACI-0325409 at USC Internet and Grid Research Laboratory. We thank Cliff Neuman and Kevin Leahy at USC/ISI for assisting us to carry out the large-scale DETER experiments reported here.

References

[1] H. Aljifri, "IP Traceback: A New Denial-of- Service Deterrent," *IEEE Security and Privacy* , May/June 2003, pp. 24-31.

[2] T. Anderson, et al, "Rocketfuel: An ISP Topology Mapping Engine"http://www.cs.washington.edu/research/networking/rocketfuel/, 2006.

[3] S. Bellovin, J. Schiller, and C. Kaufman, "Security Mechanism for the Internet", *RFC 3631*, Internet Eng. Task Force, 2003.

[4] T. Benzel, et al, "Experience with DETER: A Testbed for Security Research", *Second IEEE Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities* (TridentCom2006).

[5] R. Blazek, et al, "A Novel Approach to Detection of DoS Attacks via Adaptive Sequential and Batch-sequential Change-Point Detection Methods," *Proc. of IEEE Workshop on Information Assurance and Security*, June 2001.

[6] M. Cai, K. Hwang, J. Pan and C. Papadupolous, "WormShield: Fast Worm Signature Generation with Distributed Fingerprint Aggregation", *IEEE Trans. of Dependable and Secure Computing,* Vol.4, No. 2, April/June 2007.

[7] G. Carl, G. Kesidis, R. Brooks, and S. Rai, "Denial-of-Service Attack Detection Techniques," *IEEE Internet Computing,* January/February 2006.

[8] A. Chakrabarti and G. Manimaran, "Internet Infrastructure Security: A Taxonomy", *IEEE Network*, November 2002.

[9] S. Chen and Q. Song, "Perimeter-Based Defense against High Bandwidth DDoS Attacks," *IEEE Trans. on Parallel and Distributed Systems,* Vol. 16, No. 6, June 2005.

[10] Y. Chen and K. Hwang, "Collaborative Detection and Filtering of Shrew DDoS Attacks using Spectral Analysis," *Journal of Parallel and Distributed Computing*, Special Issue on Security in Grids and Distributed Systems, Sept. 2006, pp.1137-1151.

[11] Y. Chen and K. Hwang, "Collaborative Change Detection of DDoS Attacks on Community and ISP Networks", *IEEE Int'l Symp. on Collaborative Technologies and Systems* (CTS 2006), Las Vegas, May 15-17, 2006.

[12] X. Dimitropoulos, D. Krioukov, G. Riley, and K. Claffy, "Revealing the Autonomous System Taxonomy: The Machine Learning Approach," the *Passive and Active Measurement (PAM) Workshop* in 2006.

[13] D. Dittrich, "The 'Stacheldraft' Distributed Denial of Service Attack Tool," http://staff.washington. edu/dittrich/, 2000.

[14] M. Faloutsos, C. Faloutsos, and P. Faloutsos, "On Power-law Relationships of the Internet Topology," *Proc. of ACM SIGCOMM,* Aug. 1999.

[15] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and

Aggregation Strategy," Network Working Group, *RFC 1519*, Internet Engineering Task Force,, 1993.

[16] T. Gil and M. Poletto, "MULTOPS: a Data-Structure for Bandwidth Attack Detection," *Proceedings of 10th USENIX Security Symposium,* August 2001.

[17] K. Houle, et al, "Trends in Denial of Service Attack Technology", 2001, www.cert.org/archive/pdf/.

[18] A. Hussain, J. Heidemann, and C. Papadopoulos, "Identification of Repeated Denial of Service Attacks," *IEEE INFOCOM* 2006, Barcelona, Spain, April 23-29, 2006.

[19] K. Hwang, M. Cai, Y. Chen, and M. Qin, "Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes", *IEEE Trans. on Dependable and Secure Computing,* Vol.4, No.1, January-March 2007, pp. 41-55.

[20] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense against DDoS Attacks," *Network and Distributed System Security Symposium.* (NDSS), San Diego, CA. Feb. 6-8, 2002.

[21] ISO 3166 Report,"AS Resource Allocations", 2006, http://bgp.potaroo.net/iso3166/ascc.html

[22] H. Jiang and C. Dovrolis, "Why is The Internet Traffic Bursty in Short Time Scales?" *SIGMETRICS'05,* June 6-10, 2005, Banff, Alberta, Canada.

[23] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial-of-Service Attacks: Characterization and Implications for CDNs and Web Sites", *Proceedings of Int'l World Wide Web Conference,* ACM Press, 2002.

[24] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds", *Second Symposium on Networked Systems Design and Implementation* (NSDI), Boston, MA, May 2005.

[25] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," *RFC 2401*, Internet Engineering Task Force, 1998.

[26] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: Statistics-Based Overload Control Against Distributed Denial of-Service Attacks," *Proc. INFOCOM,* 2004.

[27] T. Law, J. Lui, and D. Yau, "You can run, but you can't hide: an effective statistical methodology to trace back DDoS attackers," *IEEE Trans. on Parallel and Distributed Systems*, Sept. 2005.

[28] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network," *Computer Comm. Review*, July 2002.

[29] J. Mirkovic and P. Reiher, "D-WARD: A Source-End Defense Against Flooding DoS Attacks," *IEEE Trans. on Dependable and Secure Computing,* July 2005, pp. 216-232.

[30] Monk and K. Claffy, "Cooperation in Internet Data Acquisition and Analysis," Coordination and Administration of the Internet Workshop, Cambridge, MA., Sept. 8-10, 1996, (CAIDA Project), http://www.caida.org/.

[31] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," *Proc. of the 10th USENIX Security Symposium,* 2001.

[32] P. Ning, S. Jajodia, and X. S. Wang, "Abstraction-based Intrusion Detection in Distributed Environment", *ACM Trans. On Information and System Security*, Nov. 2001, pp. 407-452.

[33] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, R. Govindan, "COSSACK: Coordinated Suppression of Simultaneous Attacks," *Proc. of DISCEX III*, 2003, pp. 2—13.

[34] T. Peng, C. Leckie, and K. Ramamohanarao, "Detecting Distributed Denial of Service Attacks by Sharing Distributed Beliefs," *The Eighth Australasian Conference on Information Security and Privacy* (ACISP 2003), Australia, July 9-11, 2003.

[35] J. Postel, "Internet Control Message Protocol," Network Working Group RFC 792, 1981.

[36] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, "DDoS-resilient Scheduling to Counter Application Layer Attacks under Imperfect Detection," *IEEE INFOCOM* 2006, Barcelona, Spain, April 23-29, 2006.

[37] T. Ryutov, L. Zhou, C. Neuman, T. Leithead, and K. E. Seamons, "Adaptive Trust Negotiation and Access Control," *ACM Symposium on Access Control Models and Technologies* (SACMAT'05), Stockholm, Sweden, June 1-3, 2005.

[38] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos, "Power-Laws and the AS-level Internet Topology", *ACM/IEEE Trans. on Networking*, pp. 514-524, Aug. 2003.

[39] S. M. Specht and R. B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures," *Proc. of Parallel and Dist. Comp. Systems,* S. F., Sept. 2004.

[40] J. Sommers and P. Barford, "Self-Configuring Network Traffic Generation," in Proc. of ACM Internet Measurement Conference, Taormina, Sicily, Italy, Oct. 25-27, 2004.

[41] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker "DDoS Defense by Offense," *ACM SIGCOMM* 2006, Pisa, Italy, September 2006.

[42] H. Wang, D. Zhang, and K. Shin, "Change-Point Monitoring for the Detection of DoS Attacks," *IEEE Trans. on Dependable and Secure Computing,* Vol. 1, Oct.-Dec., 2004.

[43] X. Wang, S. Chellappan, P. Boyer, and D. Xuan, "On the Effectiveness of Secure Overlay Forwarding Systems under Intelligent Distributed DoS Attacks," *IEEE Trans. on Parallel and Distributed Systems,* Vol. 17, July 2006.

Yu Chen received the Ph.D. in Computer Engineering at University of Southern California (USC) in 2006. He is an Assistant Professor of Electrical and Computer Engineering,State Uniuversity of New York at Binghamton. His research interest includes network security, DDoS defense, and Internet technology. He is a member of the IEEE and ACM. He can be reached at ychen@binghamton.edu.

Kai Hwang received the Ph.D. from the University of California, Berkeley in 1972. He is a Professor of Electrical Engineering and Computer Science at University of Southern California. An IEEE Fellow, he specializes in computer architecture, Internet security, distributed and peer-to-peer computing. Contact him at kaihwang@usc.edu or visit http://GridSec.usc.edu/Hwang.html.

Wei-Shinn Ku received his B.S. from National Taiwan Normal University in 1999 and the M.S. and Ph.D. in Computer Science from USC in 2003 and 2007, respectively. His research interests include spatial and temporal data management, geographical information systems, and peer-to-peer systems. He is a member of ACM and IEEE. Contact him at wku@usc.edu.