A Framework for A Collaborative DDoS Defense

George Oikonomou,
Jelena Mirkovic
University of Delaware
oikonomo, sunshine@cis.udel.edu

Peter Reiher
UCLA
reiher@cs.ucla.edu

Max Robinson*
Aerospace Corporation
maxrob@gmail.com

Abstract

Increasing use of the Internet for critical services makes flooding distributed denial-of-service (DDoS) a top security threat. A distributed nature of DDoS suggests that a distributed mechanism is necessary for a successful defense. Three main DDoS defense functionalities — attack detection, rate limiting and traffic differentiation — are most effective when performed at the victim-end, core and source-end respectively. Many existing systems are successful in one aspect of defense, but none offers a comprehensive solution and none has seen a wide deployment. We propose to harvest the strengths of existing defenses by organizing them into a collaborative overlay, called DefCOM, and augmenting them with communication and collaboration functionalities. Nodes collaborate during the attack to spread alerts and protect legitimate traffic, while rate limiting the attack. DefCOM can accommodate existing defenses, provide synergistic response to attacks and naturally lead to an Internet-wide response to DDoS threat.

1. Introduction

As more critical infrastructure services and time sensitive business transactions move to the Internet, flooding distributed denial-of-service (DDoS) attacks are becoming an increasing threat. Since flooding attacks can be launched in many ways, numerous defenses have been proposed (e.g., [6, 10, 18]) that either handle a specific scenario [10] or aim to offer a comprehensive defense but at a high cost [18, 6]. None of these defenses has seen a wide deployment, yet wide deployment is necessary to combat DDoS threat. We propose to improve this situation by providing means for different defense systems to organize themselves into a collaborative framework and achieve a synergistic defense against a wide variety of DoS attacks.

We first observe that there are three critical defense functionalities: (a) accurate attack detection, (b) rate limiting of traffic to free critical resources, and (c) traffic differentiation to separate the legitimate from the attack traffic and minimize collateral damage. These functionalities are best performed at different locations in the Internet. A *victim-end* defense (e.g., [12]) maximizes detection accuracy since it can observe all the traffic reaching the victim as well as the victim's resource consumption. Some attack traffic can be distinguished from the legitimate user's traffic by *source-end* defenses (e.g., [7]), through extensive statistics gathering. This is feasible because of a low address diversity (assuming deployment of ingress filtering [4]) and low traffic rate at the source. Sophisticated (flash-crowd) attacks can be differentiated from the legitimate traffic at the victim though cooperation with good traffic sources ([18]). *Core network defenses* (e.g., [16]) are necessary to rate limit large floods that would overwhelm the victim's access links. This discussion illustrates that a complete DDoS defense must involve nodes at all three locations, that collaborate in the defense, to leverage strong points of their deployment locations and minimize their weaknesses.

The advantage of distributed over single-point defense has been recognized [18, 6, 10]. Some recently proposed defenses use collaborating source-end and victim-end nodes [10], while others deploy collaborating nodes at the victim and core networks [19]. While they perform well against a variety of attacks, they do not completely handle the flooding DDoS threat. Specifically, source/victim defenses fail to handle large attacks launched from legacy networks, while victim/core defenses inflict high collateral damage to legitimate traffic. A few defenses combine defense nodes at all three locations [6, 18]. These defenses achieve higher effectiveness, but focus on a single approach to defense (e.g., a capability mechanism in [18], victim-hiding in [6]), which ultimately discourages integration with other defenses and wide deployment.

We believe that two necessary requirements for a successful defense against flooding attacks are: (a) collaborative defense, including nodes at all three deployment loca-

*Mr Robinson performed this work during his graduate studies at UCLA

tions, and (b) the ability to accommodate existing and heterogeneous defenses (possibly deployed in non-contiguous manner) to achieve wide deployment. We propose DefCOM, a distributed collaborative framework for flooding DDoS defense. DefCOM combines the advantages of source-end, victim-end and core defenses and allows the existing heterogeneous defense systems to cooperate through an overlay. The overlay facilitates communication among non-contiguously deployed nodes. Nodes collaborate by exchanging messages, marking packets for high or low priority handling, and prioritizing marked traffic. We first described the idea and the design of the DefCOM system in [9]. In this paper we present more details about the design, we specify various mechanisms that secure DefCOM's operation from insider and outsider threats, describe a prototype implementation in a Linux router and test this implementation in live experiments.

DefCOM does not contain a novel attack detection or response mechanism. Instead, a lightweight communication and traffic policing capability of DefCOM is designed to be coupled with existing defenses, to facilitate their collaborative action. We use several existing defense systems in our prototype implementation, but a variety of other defenses could be integrated with DefCOM in real-world deployment. The novelty of DefCOM is in defining collaborative mechanisms usable by a variety of existing defense systems deployed at distributed participants. To our best knowledge, DefCOM is currently the only collaborative framework that can accommodate heterogeneous defense nodes.

2. DefCOM Overview

DefCOM provides added functionality to existing defenses so they can collaborate in DDoS detection and response though a dynamically-built overlay. There are three types of DefCOM functionalities that can be added to existing routers or defense nodes. A single physical node may host more than one DefCOM functionality: (1) A *classifier* functionality is added to existing defenses that are capable of differentiating the legitimate from the attack traffic. A classifier marks packets recognized as legitimate with a HIGH-priority mark that guarantees priority handling by downstream DefCOM nodes. (2) A *rate limiter* functionality is deployed by routers. During an attack, a rate limiter runs a weighted fair share algorithm (WFSA) to prioritize traffic it forwards to the victim, and it rate limits this traffic to preserve victim's resources. (3) An *alert generator* functionality is added to defenses that can detect a DoS attack. An alert generator propagates the attack alert to other DefCOM nodes using the overlay. The alert contains the IP address of the attack's victim and specifies a desired rate limit, e.g., the size of the victim's bottleneck link.

Classifiers and rate limiters need to be deployed inline since they manipulate the traffic; alert generators could be deployed inline or as passive monitors. All DefCOM nodes that forward traffic to the victim are expected to obey the rate limit advertised in attack alert messages. This means that all *routers* or inline defenses that join DefCOM must deploy a rate limiter.

Nodes that are direct neighbors in the overlay are called *peers*. Peering links are built dynamically, using traffic flow information, as described in Section 2.1. Alert generator nodes are always active, examining traffic for signs of attack, while classifiers and rate limiters are quiescent during normal operation and become active only during an attack. Activation is triggered by an alarm message generated by an alert generator, and flooded to all overlay nodes. Active nodes start their classifier or rate limiter functionality, and mark packets they forward to the victim with a stamp periodically negotiated with their peers. There are two types of stamps: (1) HIGH priority stamps are initially used by classifiers to mark packets that have passed their legitimacy tests, and (2) LOW priority stamps are used by classifiers and rate limiters to denote traffic that is below a victim-specified rate limit but whose legitimacy cannot be verified. Packets marked for HIGH priority handling receive much better service that LOW-marked and unmarked packets, and are isolated from the attack using WFSA. Since stamps are only valid between two DefCOM peers, every DefCOM node restamps the packets that pass the rate limit with its own stamps.

Fig. 1 illustrates DefCOM operation using a simple network topology. Router A deploys a classifier and a rate limiter functionality, and hosts some source-end defense. Routers B and F deploy only a rate limiter functionality. Router H deploys a rate limiter and an alert generator functionality, and hosts some victim-end defense. Thin lines represent physical connections between nodes, and routers C, D, E and G are legacy routers.

2.1. Dynamic Overlay Constuction

We use traffic flows to dynamically build DefCOM peering relationships between nodes that are deployed inline. The resulting overlay is used only for DefCOM control message exchange while data packets flow on the routes defined by Internet routing protocols.

A DefCOM node advertises itself by generating a DEFJOIN message with a small probability p_{JOIN} for a packet sniffed from its forwarding path. The message has its destination IP copied from the packet, and carries the source IP address of the DefCOM node and a certificate that binds the node's identity with its public key, and grants the node a permission to join the overlay.

The DEFJOIN messages are generated to a currently unassigned UDP port. If they hit a DefCOM node en route

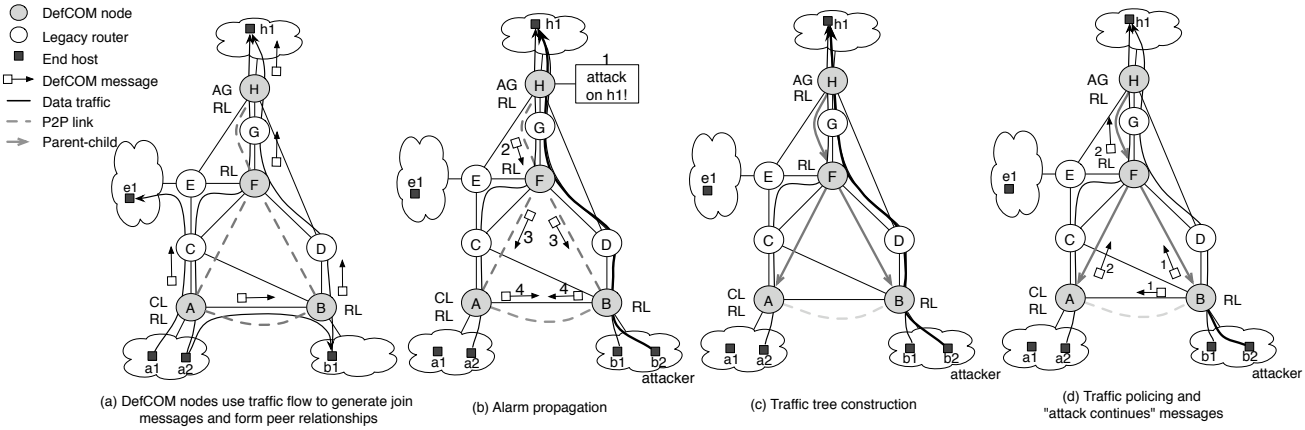


Figure 1. Illustration of DefCOM operation.

to the destination, they will be intercepted and the receiving node will verify the certificate, add the originator's IP address to its peer list, and generate a DEFREPLY to the source IP from the DEFJOIN message. DEFREPLY messages are processed in the similar manner as DEFJOIN messages. A DEFJOIN message that does not hit a DefCOM node will be silently dropped at the destination.

For security, a DEFJOIN message includes a nonce which should be returned in the DEFREPLY message, to prevent a denial-of-service attack with non-solicited DEFREPLY messages. A session key is also exchanged via DEFJOIN messages, and used for encryption of future control messages between two peers.

While traffic flows on a given path, periodic DEFJOIN messages will refresh the corresponding peer relationships. If traffic subsides, each node will remove stale peers after some set timeout (60 seconds in our prototype). In Figure 1 (a) we show a few traffic flows with thicker, solid lines, and we illustrate DEFJOIN messages with squares and an arrow line. Dashed lines show resulting peering relationships.

Control messages are exchanged only between peers and encrypted with the session key. Messages use the UDP protocol to avoid congestion response but we implement a reliable delivery mechanism at the application layer. All the messages are acknowledged by the receiver; the sender retransmits a message some fixed number of times if an acknowledgment has not been received. Table 1 lists all the control messages, which we describe in the following sections.

Control Messages	Usage
DEFJOIN / DEFREPLY	Formation & reinforcement of the P2P link
STMP	Mark exchange
ALRM	New attack alert
ATCK-CONT	Ongoing attack alert

Table 1. DefCOM Messages

2.2. Packet Marking Mechanism

DefCOM's packet marking mechanism ensures that packets verified as legitimate by a classifier will receive high priority treatment by downstream nodes. Every DefCOM node has a HIGH and a LOW stamp, which is a number to be placed in clear in a packet's IP header. Packet marking is done only by active nodes, that lie on the path of the traffic going to the victim of an attack. We use two stamps because many defenses [7, 18, 6] can only ascertain if a packet is legitimate, and such packets will be marked with the HIGH stamp. If a defense could provide a higher granularity of traffic separation, more stamps would be needed to take advantage of this for traffic prioritization.

A node changes its stamps periodically (every T_{change}, currently 5 seconds) to reduce the danger of a sniffing or guessing attack, and communicates these stamps to all its peers in an encrypted STMP message. When the message is acknowledged by all its peers, or after a timeout, the node starts using new stamps.

We place the stamp in the ID field of the IPv4 header, which is normally used for fragmented packet identification, and we drop fragmented traffic going to the victim during an attack. We believe that the damage to fragmented traffic will be minimal because: (1) fragmented traffic makes a very small portion (0.25%) of the Internet's traffic [11], and (2) DefCOM only marks traffic *going to the victim during the attack*, so the fragmented traffic loss is limited. We could place the stamps in the IP options field instead, but because some routers process such packets on the slow path this would jeopardize performance.

2.3. Operation

Alarm propagation. When a defense coupled with an alert generator detects the attack, the alert generator creates ALRM message with the victim's IP address and resource limits (RLM). Our current prototype focuses on defense against bandwidth attacks, so we express RLM in units of bandwidth. Alarm message is flooded on the overlay, in a controlled manner to suppress duplicate messages. A node floods an ALRM message to all its peers and remembers it for T_r (currently 6) seconds. Duplicate messages will refresh the timer and will be silently dropped. A node that receives an ALRM message becomes active and starts to classify or rate-limit traffic, according to its functionality. In Figure 1 (b) we illustrate ALRM message flooding (squares with arrows, numbers denote the message order) that occurs when attacker b2 launches an attack against victim h1.

Traffic tree construction. Active nodes organize themselves into a *traffic tree* (subgraph of the overlay) containing only the nodes that forward any traffic to the victim. Nodes use the traffic flow to the victim to discover parent-child (upstream-downstream) relationships with their peers. Traffic tree is used by a node to keep track of each child's traffic output, and identify malicious insiders.

A parent-child relationship is discovered using packet marking. An active classifier or rate limiter marks all forwarded packets with its HIGH or LOW stamp. When a node observes traffic with its peer's stamp it flags this peer as a "child". This process leads to a distributed formation of a traffic tree. For example, in Fig. 1 (c), node F observes stamped packets from nodes A and B and becomes their parent, while node H becomes parent of node F. The traffic tree is represented with thick gray lines, and arrows denote parent-child relationships.

Traffic policing. Excess traffic to the victim is controlled by rate limiter nodes. A rate limiter first reclassifies each incoming packet based on its current stamp and the aggressiveness of the child that forwarded this packet, and then rate limits all outgoing traffic to RLM. Reclassification rules are the following: (1) A rate limiter keeps byte count of all traffic received from each child in the past T_{Mal} seconds (currently $T_{Mal} = 5$). A child whose average output is more than RLM will be considered aggressive and all its packets will be reclassified as unstamped. Since each active node should rate limit the traffic forwarded to the victim during an attack, nodes that violate this requirement are clearly malicious. Packets from non-aggressive children will preserve their HIGH or LOW classification and will be marked with rate limiter's stamps. (2) A rate limiter keeps byte count of all unstamped traffic received in the past T_{Mal} seconds. If its average is lower than RLM all unstamped packets will be marked with a LOW stamp. This rule helps identify legacy traffic from networks that do not host attackers but also do not deploy classifiers. LOW classification helps improve chances of such traffic in competition with attack, while ensuring better service for verified-legitimate traffic marked with HIGH stamp.

Resource allocation and rate-limiting are performed after reclassification using a weighted fair sharing algorithm (WFSA) we describe in Section 5. In real deployment a rate limiter could use a weighted fair queueing module available in many commercial routers. In our experiments we use the following weights: $w_{HIGH} = 0.9$ and $w_{LOW} = 0.1$, and we drop unstamped traffic.

Deactivation. A node that drops traffic due to rate limiting generates an ATCK-CONT message every T_{end} seconds (currently $T_{end} = 6$) and floods it on the overlay, using the same rules as for the ALRM message flooding to control the overhead. A node detects the end of the attack *locally* if it drops no traffic due to rate limiting in the last T_{end} seconds. It then stops generating ATCK-CONT messages but still forwards these messages sent by other nodes. The global end of the attack is detected when a node does not receive or generate any ATCK-CONT message in the last $2 * T_{end}$ seconds, which means that all drops due to rate limiting have stopped. Figure 1 (d) illustrates traffic policing and ATCK-CONT messages (squares with arrows, arrows, numbers denote the message order).

We selected the values for various time intervals (e.g. T_{Mal}, T_{end}, T_r) empirically, to balance the reaction speed with the accuracy. Higher values slow down DefCOM's response and lower values make the system overreact to bursty traffic. The weights w_{HIGH} and w_{LOW} should be selected based on the confidence of the legitimacy tests. Accurate tests warrant larger w_{HIGH} (and smaller w_{LOW}) values.

3. DefCOM security

We prevent malicious nodes from joining DefCOM by requiring DEFJOIN and DEFREPLY messages to carry a valid certificate, issued through human channels after a node shows that it meets some required security criteria. Certificates could be issued by some global certification authority, or current DefCOM nodes could vouch for the trustworthiness of a new node and cast a vote in its favor.

Fabrication and replay of control messages is prevented by signing each message by the originator's private key, encrypting it with the session key and attaching a sequence number. Since all control messages are exchanged between peers, and are not frequent, the cryptographic and key exchange cost is small.

An attacker could deny DefCOM's service by flooding a node with bogus messages and forcing it to pay the price of cryptographic verification. A DefCOM node defends against this attack by requesting that each control message carries a valid peer stamp, that serves as a nonce.

Ensuring security against malicious participants is difficult for any distributed protocol. An insider could interfere with DefCOM's operation in various ways, fabricating or suppressing messages. For space reasons, we discuss here only the most harmful attacks.

Sending excessive messages to a peer. This attack can be handled by limiting the rate of messages a node is willing to receive from each peer.

Lying about the attack via false ALRM messages. DefCOM alert generators must possess an authorization to issue alerts for a given victim. One solution would be to bind specific networks to alert generators that are allowed to issue alerts for them, using a DefCOM certificate. For instance, an ISP's alert generator could be authorized to issue alerts for this ISP and its customers.

Marking attack traffic with HIGH priority mark. Falsely-marked traffic competes with legitimate traffic for resources because both traffic flows are marked for HIGH-priority handling. An attacker can generate high-rate attacks by either compromising a few classifiers (sparse attack), and sending at a high rate, or by compromising many classifiers (diffuse attack) and sending at a low rate from each one. We counter sparse attacks via non-aggressive checks and reclassification in section 2.3. To counter diffuse attacks, we enhance the rate limiter functionality with *active testing*, aimed to identify malicious children. Note that these attacks are difficult for the attacker to create, since he must compromise many classifier nodes that are also well distributed to minimize rate-limiting.

Active testing is performed by a rate limiter, that receives total HIGH-stamped traffic at the average above RLM over period of T_{Mal} seconds. Testing consists of forwarding and dropping phase, each T_{test} (currently 5) seconds long. The goal of the testing is only to confirm that traffic marked HIGH by a child is congestion responsive, which verifies the legitimacy of this child. In the forwarding phase, the node chooses a set of children at random, so that the sum of their HIGH-stamped traffic is lower than RLM. It then forwards all the traffic from these children and records the average arriving rate of each child's HIGH-stamped traffic, R_1. Traffic from other children is dropped to ensure that the tested traffic experiences minimal congestion. During the dropping phase, all tested children's traffic is dropped and the average arriving rate of each child's HIGH-stamped traffic, R_2 is recorded. If $R_2 < 0.2 \cdot R_1$, in response to packet drops, this verifies that the traffic marked for HIGH priority handling by the tested child is congestion responsive. The rate limiter then marks this child as "confirmed-legitimate". Otherwise the child is marked as "malicious" and its traffic will be dropped by the end of the attack. The forwarding phase of one test-set can be overlapped with the dropping phase of the previous set, so that the testing phase is relatively short.

Due to a random choice of children set to be tested, the attacker has low probability of guessing when his malicious classifier is chosen for testing, and cannot fake the congestion response. The active testing has an obvious limitation that it only properly confirms legitimacy of children that mark TCP traffic with a HIGH-priority mark. While this is undesirable, we note that the issue of trust in distributed systems is a known hard problem, and that many distributed DDoS defenses do nothing to discover and eliminate malicious insiders [6, 18, 5, 10]. DefCOM with active testing thus has a better security model than other distributed DDoS defenses. Our future research will investigate how a victim's feedback could be integrated with active testing to improve the accuracy of malicious child identification.

3.1. Robustness to message loss

DefCOM implements application-level reliability mechanisms to counter sporadic message loss due to congestion created by the attack. Each message has to be acknowledged by the recipient. Unacknowledged messages are repeated after T_{rto} seconds (currently $T_{rto} = 2$). We expect that attack will not interfere with ALRM messages, since they travel in the opposite direction on full duplex links.

4. Scalability

DefCOM nodes communicate only with peers in the overlay, so the communication scalability depends on the number of peers. While the connectivity of the overlay depends on the underlying physical topology, the pattern of defense nodes' deployment and traffic patterns, deployment strategies that place rate limiters in the core lower the degree of all nodes and improve scalability.

A node stores only a small amount of state information per peer — some traffic statistics data, peer stamps and a public key — so the memory requirement is modest even for a node with several thousands of peers. The other factor that affects scalability is the number of attack reports, as a separate traffic tree is built for each report. We plan to investigate strategies to combine traffic trees in cases when multiple attack reports coincide, but we expect that this should not be a frequent situation.

5. Implementation

We implemented DefCOM in a Linux router (RedHat 9.0), with the packet marking, WFSA and rate limiting being implemented as a loadable kernel module, and the messaging between peers implemented at the application layer.

We couple an alert generator with a simple mechanism that detects an attack if one of the following rules become

true: (Rule 1) The ratio of the incoming to outgoing TCP packets is higher than 3. This rule tests in a crude manner if the incoming TCP traffic is congestion responsive; (Rule 2) The total incoming traffic rate is larger than the bottleneck link bandwidth during last 3 seconds. This attack detection is simple but sufficient to detect attacks in our experiments. In a real deployment, an alert generator should be coupled with more sophisticated detection mechanisms, and our future work will integrate [12] with DefCOM.

We couple D-WARD [7, 8] with a classifier. D-WARD prevents outgoing DoS attacks by keeping statistics on incoming and outgoing packet counts for each TCP connection established with the victim, and using these statistics to differentiate legitimate from attack traffic. D-WARD classifies TCP connections with low sent-to-received packet ratio as legitimate, and uses application-level models for detection of legitimate UDP connections. It also distinguishes legitimate TCP SYN packets by performing sequence number prediction for known source hosts. D-WARD can detect and respond to attacks autonomously, but we disabled these functionalities to force D-WARD to act as pure traffic classification engine. Other traffic classification approaches could be interfaced with DefCOM, such as [18].

We deploy the WFSA algorithm in the rate limiter, using ideas from core-stateless fair queuing [15]. WFSA has two traffic classes: HIGH and LOW priority. The algorithm estimates the resource consumption rate in class i after receipt of a packet of size l_i, as:

$$r_i^{new} = (1 - e^{-dT_i/S})\frac{l_i}{dT_i} + r_i^{old}e^{-dT_i/S}, \;\; i = 1...n \quad (1)$$

where r_i is the consumption estimation, dT_i is the time elapsed from the last packet's arrival, S is the average time interval for the estimation and n is the total number of the classes. After each packet's arrival, its forwarding probability p_{FW} is computed as [15]:

$$p_{FW} = \min(1, \frac{w_i \cdot \alpha}{r_i}) \;\; i = 1...n \quad (2)$$

where w_i is the weight for the class i, and α denotes the fair share of RLM. The fair share is updated every K seconds (currently K= 0.333) by first calculating the true resource consumption rate R_i for each class i. If $\sum_{i=0}^{n} R_i \leq RLM$, then the link is not congested and we set $\alpha = \max_{i=1...n}(R_i)$. We do not allow α to decrease more than 20% during two consecutive updates, to avoid overreaction to traffic bursts. If $\sum_{i=0}^{n} R_i > RLM$, then α is the unique solution of the equation

$$\sum_{i=1}^{n} \min_i(R_i, w_i * \alpha) = RLM \quad (3)$$

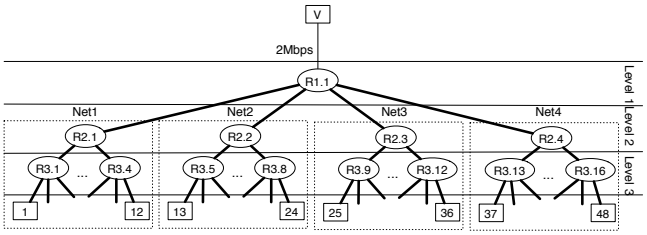


Figure 2. Large-scale topology

6. Evaluation

We evaluate DefCOM in live-traffic experiments in the Emulab testbed [17]. In all experiments we create legitimate traffic by establishing multiple telnet-like sessions over TCP between good clients and the victim. The attack is created by sending high-volume TCP data packets to the victim, using the *raw socket* functionality. Although a lot of bandwidth attacks use UDP or ICMP traffic, we deliberately chose TCP to make the attack traffic similar to the legitimate traffic. Modern DoS tools use the same variety of attacks for bandwidth exhaustion, their sophistication lies only in hiding control traffic between attack machines.

We initially conducted many experiments to test DefCOM's performance using the simple topology of Fig 1, and varying legitimate traffic (FTP-like vs telnet-like) and attacks (UDP, ICMP, TCP floods). Due to space constraints we summarize the results from these small-scale experiments and we next present in detail experiments in a large-scale topology.

In classifier tests we concluded that: (1) Legitimate traffic receives priority treatment by DefCOM and is well isolated from the attack; (2) Traffic classification and isolation are not sensitive to variations in legitimate and attack traffic rates, but higher rates introduce more variance in legitimate traffic's service.

In rate limiter tests we concluded that: (1) Legitimate traffic from a network with a classifier receives priority treatment by DefCOM and is well isolated from the attack; (2) Traffic classification and isolation are not sensitive to variations in legitimate and attack traffic rates; (3) Legitimate traffic from legacy networks competes with the attack for bandwidth; (4) We denote as *malicious* the traffic that comes from attack hosts, but is marked with HIGH mark by a malicious classifier. Malicious traffic competes with legitimate traffic for bandwidth. Attackers must generate limited-rate malicious traffic to pass the non-aggressive test.

In the next set of experiments we test DefCOM with the topology shown in Fig. 2. There are three levels of routers, and 48 traffic sources. The bottleneck link leads from a level-1 router to the victim and its size is RLM=2 Mbps. In all tests R1.1 deploys a rate limiter and an alert generator. Attack hosts are deployed in Net1 on 2 out of each 3

hosts (hosts 2, 3, 5, 6, 8, 9, 11, 12) and in Net3 and Net4 throughout (hosts 25-48). The legitimate hosts are in Net1 (hosts 1, 4, 7 and 10) and in Net2 (13-24). Total legitimate traffic reaching the bottleneck link is 0.7*RLM. Since each legitimate host sends at the same rate, the baseline value for legitimate traffic from Net1 is 0.525*RLM and for Net2 it is 0.175*RLM. Without DefCOM, the attack occupies all of the bottleneck's bandwidth and completely denies service to legitimate traffic. This topology and settings are similar to Pushback dense experiments from [5], where traffic from Net1 was called "poor" because it shares the path with the attack before it reaches defense nodes. Pushback, as well as many other defenses, cannot help such users and inflict collateral damage to poor traffic. We expect DefCOM to protect this traffic when classifiers are deployed in Net1. In experiments we use two attack rates: the low rate generates 2.88 Mbps flow on the bottleneck link, while the high rate generates 4.8 Mbps.

We acknowledge that the topology we use, containing 70 PCs, does not match the scale of real DDoS attacks that involve hundreds of thousands of nodes. However, our experiments involve live traffic generated from real PCs. We chose live traffic experimentation over simulation because current network simulators cannot faithfully capture network behavior under extreme traffic load. Due to the disruptive nature of our experiments they must be fully contained in an isolated testbed. Two large testbeds accessible to researchers, Emulab [17] and Deter [2] have on the order of 200 nodes each, and these nodes are shared among multiple researchers. Under these circumstances, we designed our experiments with a largest topology we could obtain for our exclusive use. While these experiments are smaller-scale than real DDoS events, they are largest non-simulated DDoS experiments in the research literature.

6.1. Full Deployment

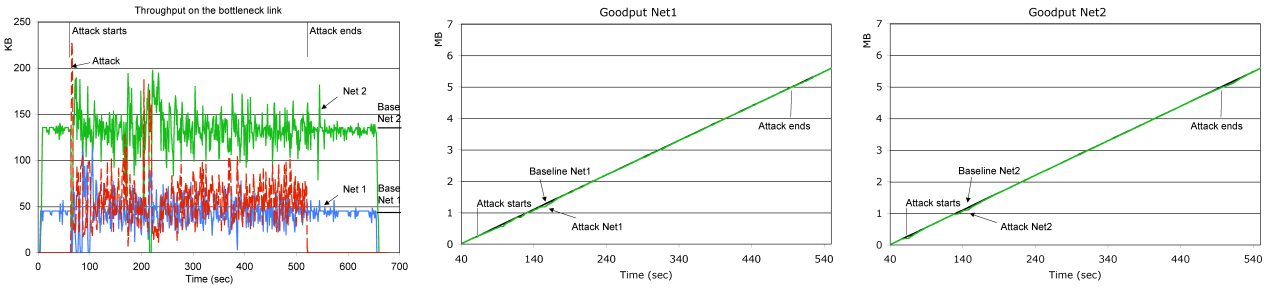
In the experiments FL1 and FL2 we test DefCOM under full deployment, with high-rate and low-rate attack traffic. This illustrates DefCOM's performance in an ideal setting, if it were widely deployed, and also represents a baseline to compare with partial deployment results. We deploy classifiers at all level-3 routers and we deploy rate limiters at all level-2 routers. Fig. 3 shows the usage of the bottleneck link by legitimate and attack traffic, and the goodput of clients from Net1 and Net2. Since the graphs for low and high attack rates are almost identical, we show here only the graph for the high-rate experiment (FL2). In both experiments, traffic from Net1 and from Net2 is well-protected by DefCOM and reaches its baseline values, shown on the Figure. The rest of the bandwidth is used by the attack traffic. The protection is especially well illustrated on the goodput graphs that compare the goodput during an attack with the baseline goodput without the attack — these two lines overlap indicating that legitimate users experience no denial-of-service effect.

6.2. Partial Deployment

In the experiments PT1—PT3 we test how DefCOM performs in partial, non-contiguous deployment. In all tests we generate a high-rate attack from the attack hosts. In the experiment PT1 we deploy classifiers on all level-3 routers. Level-2 routers are not participating in DefCOM. We see from Fig. 4 that DefCOM's performance in this case is identical with a fully-deployed DefCOM. In the experiment PT2 we explore a more realistic case when the classifiers are deployed only in front of Net2 nodes — at routers R3.5—R3.8. Results in Fig. 4 show that DefCOM successfully protects legitimate traffic from Net2. This traffic is delivered to the victim at the same level as the baseline. The sum of unstamped legitimate traffic from Net1, and the attack from Net3 and Net4 exceeds RLM at the rate limiter R1.1 and fails the non-aggressive test. Almost all this traffic is dropped by DefCOM, which explains why less attack traffic reaches the bottleneck link than in full-deployment experiments. In the experiment PT3 we test if we can improve the protection of the legacy traffic from Net1 by deploying a rate limiter at node R2.1. The protection of Net1 traffic is now comparable to the full deployment case, since this traffic is marked LOW and treated differently from high-rate unstamped attack traffic.

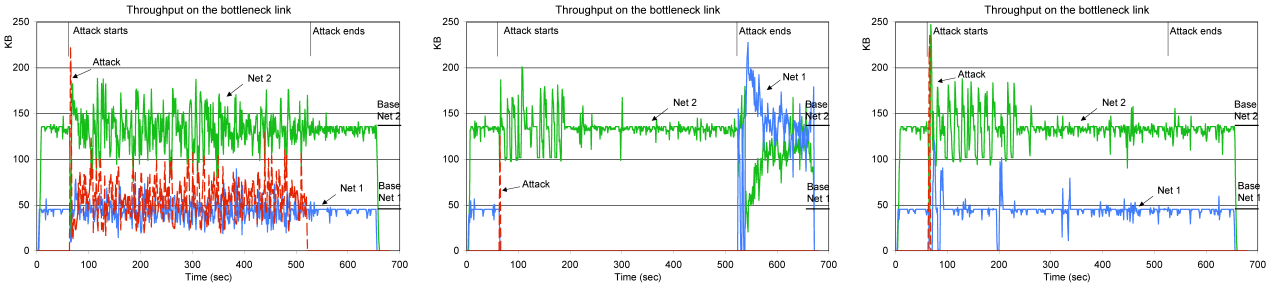
6.3. Malicious classifiers

Finally, we test how DefCOM performs when attackers deploy malicious classifiers in front of their machines and mark all traffic as legitimate. We only generate attack traffic from machines 25—48, i.e., we remove attackers from Net1. In the experiments ML1 and ML2, rate limiters are deployed on routers R2.3 and 2.4, trusted classifiers are in front of Net2 on routers R3.5—R3.8 and malicious classifiers are on routers R3.9—R3.16. The attack is configured to be low at the second level, so that the rate limiters R2.3 and 2.4 will not reclassify the traffic as unstamped according to the rule 1 in Section 2.3. We manually disable active testing in ML1, and we enable it in ML2. Figure 5 shows the throughput on the bottleneck link. Without active testing, legitimate traffic receives a very low share while aggressive distributed attack dominates the bottleneck link. With active testing, the malicious classifiers are identified after 50 seconds and its traffic is being dropped. Legitimate traffic reaches its baseline levels after this testing period and is no further impacted by the attack. We note that legacy traffic from Net1 is also well-protected and isolated from the attack, even though this network does not deploy a classifier.



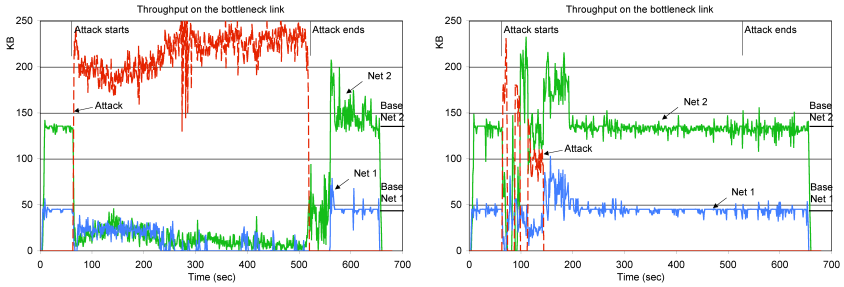
(a) Throughput in FL2 (b) Goodput of Net1 in FL2 (c) Goodput of Net2 in FL2

Figure 3. Full deployment experiments



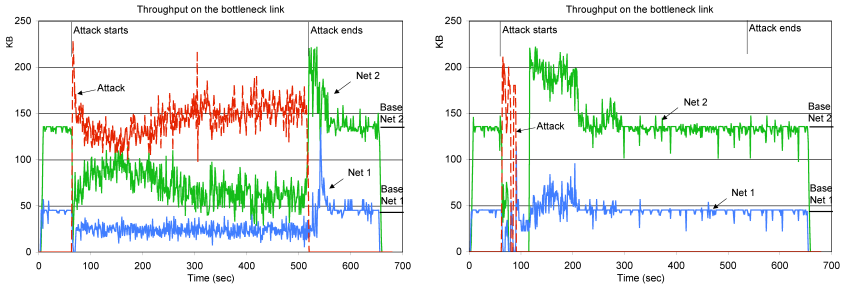
(a) Throughput in PT1 (b) Throughput in PT2 (c) Throughput in PT3

Figure 4. Partial deployment experiments



(a) Throughput in ML1 (b) Throughput in ML2

Figure 5. Experiment with malicious classifiers, and rate limiters on routers R2.3 and R2.4



(a) Throughput in ML3 (b) Throughput in ML4

Figure 6. Experiment with malicious classifiers, and rate limiters on all level-2 routers

If Net3 and Net4 had legitimate users, along with attack machines and a compromised classifier, then their traffic would also be penalized. This is unfortunate, but ultimately a compromised classifier is a problem that should be handled by local security administrators.

Finally, we show how the protection changes if we deploy rate limiters at level-2, at routers R2.3 and R2.4. In the experiment ML3 we disable active testing, and we enable it in the experiment ML4. The throughput is shown in the Figure 6. Because attack traffic passes through more DefCOM nodes it is better controlled and competes less with the legitimate traffic. Even without active testing, the legitimate traffic's service is significantly improved when compared with experiment ML1. In ML4, malicious children are quickly identified and legitimate traffic is protected.

6.4. Deployment and operation cost

We measure the attack detection time from the start of the attack to the issue of the first ALRM message, and the attack response time from the first ALRM message to the first attack packet drop. For the experiments described in this paper, attack detection time was 1.58—2.45 seconds and the attack response time was 1.78—2.93 seconds. Note again that these experiments used a very simple attack detection mechanism.

The number of messages exchanged by a DefCOM node depends on number of its peers, and the setting of DefCOM parameters $T_r, T_{end}, T_{rto}, T_{change}$ and p_{JOIN}. In our large-scale experiments we had around 5 messages per second per node and this number did not grow during attacks. We also tested robustness to message loss by deliberately dropping a certain percentage of DefCOM control messages. Our experiments indicate that DefCOM can tolerate $<20\%$ message loss, and its performance quickly degrades with higher loss rates. DefCOM's packet processing time on 850 MHZ Pentium III was around 0.5 μs without the attack and it grew to 1.3 μs in a rate limiter and a classifier during the attack. Additionally, the rate limiter's packet processing time should include WFSA cost. Our WFSA implementation took around 50 μs per packet, most of which was spent running custom-written code for exponentiation (since this function is not implemented in Linux kernel) needed for rate estimation. In real-world implementation these operations would be done through a much faster, hardware fair queueing module in commercial routers.

7. Deployment Motivation

End-network defenses would benefit from DefCOM by achieving wider observation and policing perimeter. Distributed defenses would also benefit, and we illustrate this claim by discussing an integration of the distributed TVA [18] defense with DefCOM. TVA uses server-issued capabilities to differentiate between legitimate and suspicious traffic. Routers help create capabilities, rate limit new capability requests and give highest priority to capability-carrying traffic, then to request traffic and finally to legacy traffic. TVA can handle all attacks as well as DefCOM and should outperform DefCOM with flash-crowd attacks. However, TVA is always active and its processing and memory cost are high. DefCOM can lower TVA's cost by: (1) having attack alerts serve as power-on signals for TVA so that processing cost is paid only during attacks and (2) TVA senders would mark capability-carrying traffic with a HIGH-priority mark; DefCOM would pass legitimate packets over to TVA for finer, more expensive checks only if total HIGH-marked traffic exceeds RLM.

DefCOM has a good economic model for all deploying networks. Alert generators provide victim-end networks with means to request help in handling DDoS attacks. Classifiers guarantee good service to legitimate traffic during the attack, which directly benefits the deploying network. Rate limiters can be deployed by ISPs as a service they can sell to their customers. As customers usually request their ISP's help when under a flooding DDoS attack, deploying a DefCOM rate limiter would streamline these requests and make response faster and more accurate. Our experiments indicate that DefCOM can be very effective in partial deployment. Still, the robustness against malicious classifiers and the ability to provide good service to legitimate traffic from legacy networks increases with wide deployment. We believe that a strong economic model and the ability to integrate diverse defense systems with DefCOM will naturally motivate wide deployment of this framework.

8. Related Work

We review here only those approaches that provide some form of cooperative defense between different nodes or share other strong similarities to DefCOM. We omit TVA [18] because we discussed it in the previous section.

SOS [6] uses access points (SOAPs) close to source networks to verify legitimate users and send their traffic on the overlay to secret servlets, that tunnel it to a distributed firewall protecting the victim. SOS offers good protection to the server but the traffic experiences a significant delay because it is routed on the overlay. Mayday [1] generalizes SOS approach with a variety of authentication and overlay routing mechanisms and suffers from similar drawbacks.

Pushback [5] enables routers to identify high-bandwidth aggregates that contribute to congestion rate limit them. If the congested router cannot control the aggregate itself, it requests its upstream neighbor's help in rate limiting. The performance of Pushback is good when attackers are collocated on a path separate from the legitimate traffic, other-

wise it inflicts collateral damage. Further, Pushback cannot work in non-contiguous deployment and cannot detect attacks that do not congest core routers.

Active Security System (ASSYST) [3] supports distributed response with non-contiguous deployment, with nodes equivalent to classifiers being deployed only at edge networks. COSSACK [10] similarly forms a multicast group of defense nodes that are deployed at source and victim networks and cooperate in filtering the attack. Both [3] and [10] cannot handle attacks from legacy networks that do not deploy their defense mechanisms. Parameter Based Defense [14] constructs a multicast group at an ISP that rate limits an attack originated from one of its customer networks. It requires wide deployment and does not perform well in non-contiguous deployment. Yau et al. propose a router throttle mechanism [19] installed at the routers that are close to the victim. In contrast to DefCOM, this defense system incorporates only victim-end and core defense mechanisms, and thus inflicts collateral damage to legitimate traffic. Like DefCOM, the router based solution [13] consists of an overlay of routers with added functionality, which helps them trace and stop the attacks close to the source. Tracing is done using signatures assigned to each source network, and inflicts collateral damage on legitimate users that share a network with an attacker.

9. Conclusion

DefCOM is an innovative and scalable distributed defense framework that facilitates collaboration among diverse DDoS defense systems through secure messages exchanged via an overlay network. DefCOM enables each node to perform functions it can do best, while complementing its weaknesses with the strengths of others. Victim networks are protected against DDoS attacks, source networks are guaranteed that their legitimate traffic will reach the victim and the Internet backbones can crucially contribute to the war against DDoS attacks, by selling this extra service. DefCOM provides a good economic model for all involved parties, which should naturally lead to wide deployment. We have extensively tested DefCOM covering a wide variety of network scenarios; in all cases, it offered excellent protection to legitimate traffic during a DDoS attack.

References

[1] D. G. Andersen. Mayday: Distributed Filtering for Internet Services. In *4th Usenix Symposium on Internet Technologies and Systems*, Seattle, WA, March 2003.

[2] T. Benzel, B. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with DETER: A testbed for security research. In *Proceedings of TRIDENTCOM*, March 2006.

[3] R. Canonico, D. Cotroneo, L. Peluso, S. P. Romano, and G. Ventre. Programming routers to improve network security. In *OPENSIG 2001 Workshop Next Generation Network Programming*, September 2001.

[4] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. *RFC 2267*, 2000.

[5] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router Based Defense Against DDoS Attacks. In *ISOC Symposium on Network and Distributed System Security*, February 2002.

[6] A. Keromytis, V. Misra, and D. Rubenstein. SOS: An Architecture for Mitigating DDoS Attacks. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.

[7] J. Mirkovic. *D-WARD: source-end defense against distributed denial-of-service attacks*. PhD thesis, UCLA, 2003.

[8] J. Mirkovic, G. Prier, and P. L. Reiher. Attacking DDoS at the Source. In *Proceedings of the International Conference on Network Protocols*, pages 312–321, 2002.

[9] J. Mirkovic, M. Robinson, and P. Reiher. Alliance formation for DDoS defense. In *Proceedings of the New Security Paradigms Workshop*, pages 11–18, New York, NY, USA, 2003. ACM Press.

[10] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan. COSSACK: Coordinated Suppression of Simultaneous Attacks. In *Proceedings of DISCEX*, pages 2–13, 2003.

[11] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proceedings of the ACM SIGCOMM*, pages 295–306, 2000.

[12] S. Shin, K. Kim, and J. Jang. D-SAT: Detecting SYN Flooding Attack by Two-Stage Statistical Approach. In *SAINT Symposium*, pages 430–436, 2005.

[13] Z. Shu and P. Dasgupta. Denying Denial-of-Service Attacks: A Router Based Solution. In *International Conference on Internet Computing*, pages 301–307, 2003.

[14] Q. Song. Perimeter-Based Defense against High Bandwidth DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, 16(6):526–537, 2005.

[15] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: a scalable architecture to approximate fair bandwidth allocations in high-speed networks. *IEEE/ACM Transactions on Networking*, 11(1):33–46, 2003.

[16] H. Wang and K. G. Shin. Transport-aware ip routers: A built-in protection mechanism to counter ddos attacks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):873–884, 2003.

[17] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proceedings of the Operating System Design and Implementation*, pages 255–270, 2002.

[18] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proceedings of ACM SIGCOMM*, pages 241–252, 2005.

[19] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Transactions on Networking*, 13(1):29–42, 2005.