# Trawling Traffic under Attack

Overcoming DDoS Attacks by Target-Controlled Traffic Filtering

by

Shlomi Dolev, Yuval Elovici, Alex Kesselman, Polina Zilberman

Technical Report #2009-07

August 2009

**Abstract**

As more and more services are provided by servers via the Internet, Denial-of-Service (DoS) attacks pose an increasing threat to the Internet community. A DoS attack overloads the target server with a large volume of adverse requests, thereby rendering the server unavailable to "well-behaved" users. Recently, the novel paradigm of traffic ownership that enables the clients of Internet service providers (ISP) to configure their own traffic processing policies has gained popularity. In this paper, we propose two algorithms belonging to this paradigm that allow attack targets to dynamically filter their incoming traffic based on a distributed policy. The proposed algorithms defend the target against DoS and distributed DoS (DDoS) attacks and simultaneously ensure that it continues to receive valuable users' traffic.

In a nutshell, a target can define a filtering policy which consists of a set of traffic classification rules and the corresponding amounts of traffic, measured in bandwidth units, which match each rule. The filtering algorithm is enforced by the ISP's or the Network Service Provider's (NSP) routers when a target is being overloaded with traffic. The goal is to maximize the amount of filtered traffic forwarded to the target, according to the filtering policy, from the ISP's or the NSP's network.

The first algorithm that we propose relies on complete collaboration among the ISP/NSP routers. It computes the filtering policy in polynomial time and delivers to the target the best possible traffic

mix. The second algorithm is a distributed algorithm which assumes no collaboration among the ISP/NSP routers, each router uses only local information about its incoming traffic. We show how the traffic mix that reaches the target depends on the network's topology and we prove a lower bound on the worst-case performance of the second algorithm.

# 1 Introduction

A Denial-of-Service (DoS) attack aims to prevent legitimate users from accessing shared services or resources (hereafter we will refer to the owner of the shared services or resources as the target). The attacks can be classified into two types [11]. Attacks of the first type aim to crash the system by sending few carefully crafted packets that exploit software vulnerability in the target system. The second type of attacks is employing massive volumes of useless traffic to occupy all the resources that could service legitimate users. This study focuses on the latter type of attacks. When the useless traffic is originated at multiple sources, scattered all over the Internet, the attack is called a Distributed DoS (DDoS) attack [17].

DDoS attacks represent an increasingly frequent disturbance for the global Internet and target services hosted on high-profile web servers such as banks, credit card payment gateways, and even root nameservers [15,17]. Significant research efforts have been devoted to designing security mechanisms for coping with DDoS attacks. Unfortunately, there is no "silver bullet" solution for protecting a target from DDoS attacks, since DDoS traffic may be indistinguishable from the legitimate user traffic [17].

Most Internet routers have some rate-limiting and traffic shaping capabilities [3, 13, 14]. By controlling the bandwidth consumption of different traffic classes, routers can limit the amount of resources consumed by an attack and reduce its impact. However, routers do not have enough knowledge about the priorities of each target and may discard valuable user traffic together with the attack traffic. Basically, there is no universal traffic shaping policy that would be suitable for all targets. The challenge is to design a general scheme which can successfully counter a variety of DDoS attacks and take into account the targets' priorities.

A traffic shaping policy may grant the target control over the incoming traffic by taking into account its priorities. In many cases the target wishes to have control over the traffic reaching it [18, 22]. For example, if an e-commerce website can provide service to a limited number of users, the web site operator may maximize its benefit by providing service to more profitable users. One way to achieve this goal is to control the incoming traffic at the target machine(s). However, such an approach consumes substantial computational resources. On the other hand, controlling traffic closer to its source reduces both network traffic and target workload [2, 6, 10]. We believe that large companies will be willing to pay for a

guarantee that their *benefit* from the arriving communication traffic is maximized even under attacks. Therefore, even though ISPs were reluctant to grant their users a higher degree of control over traffic shaping in their backbone network [2, 10, 17] so far, this would also be practical and profitable for ISPs. Moreover, ISPs already support Quality of Service (QoS) requests of clients, and our approach extends the existing services of ISPs to allow clients to withstand DDoS attacks.

In this paper, we address the DDoS problem by shaping the target's incoming traffic. We provide the target with means for shaping its incoming traffic in order to maximize its resulting benefits from the legitimate user traffic. A target can compute in real time a filtering policy that is based on the actual utility of the arriving traffic by using, for instance, machine learning algorithms. It is feasible for the target to do so, because the target knows the current actual benefit gained from the arriving traffic. Such a policy allocates more bandwidth to high-priority traffic and limits the bandwidth consumption of the potential DDoS traffic. Thus, the proposed algorithms defend the target against DDoS attacks by ensuring that it continues to receive valuable user traffic. Once the policy has been pushed to the ISP's or the NSP's network routers, each router shapes its outgoing traffic destined to the target according to this policy. Since this service, as any service in networking, may be compromised and even subject to a DDoS attack, we assume a standard secure distribution procedure from the target to the border router of the ISP or NSP. Thus, any change to the policy requires a proper user (target) authentication. The target may receive the feedback in the form of a summary of the new incoming traffic, including the blocked traffic characteristics, and can further refine the filtering policy. Our scheme provides a general (not attack specific) mechanism to achieve graceful service degradation in the face of a DDoS attack.

We consider the problem of enforcing a filtering policy in the ISP's or NSP's backbone network. When the capacity of incoming traffic reaches a point where it overwhelms the target, the target activates (through the ISP/NSP) a distributed traffic filtering policy. This policy includes a set of traffic classification rules, each of which has the associated amounts of bandwidth that the target wishes to receive for traffic that matches this rule. The goal is to maximize the amount of filtered traffic forwarded to the target, according to the filtering policy, from the network. We assume that the target leases lines from the ISP or the NSP, so that the overall capacity of the leased lines, which connect the target to its users, is limited to the target's processing capacity. Note that DDoS is possible even in this case, since legitimate users are not being served by the target when the target is overwhelmed by un-useful (or even harmful) attack traffic, and does not receive requests of legitimate users. In addition, we assume that the ISP/NSP supports a notification procedure in which the target gets a summary on the traffic that has been discarded.

In this study we propose two algorithms. The first algorithm assumes complete information, where each router has a full view of all the traffic in the network, in the spirit of an offline algorithm, while the second algorithm assumes that each router has information only about its incoming traffic, in the spirit of an online algorithm. The offline filtering algorithm determines in polynomial time a filtering scheme for the routers to follow, so that the traffic reaching a target complies in the best possible way with the target's policy. We study the worst-case performance of the online filtering algorithm. We prove that in the worst-case scenario the online algorithm delivers a significant fraction of the best available traffic depending on the number of filtering routers in the network's periphery.

The rest of the paper is structured as follows. In Section 2, we discuss the related work to this study and our contribution. Section 3 presents the system definitions. In Section 4, we describe the offline algorithm for optimal shaping of the network traffic. Following the offline algorithm, we illustrate in Section 5 the online filtering algorithm, and prove the online filtering algorithm's worst-case performance. Section 6 concludes the paper. In the Appendix we present two dynamic online filtering schemes; one assumes complete collaboration among the filtering routers, and the other scheme assumes partial collaboration.

## 2 Related Works and Our Contribution

**Packet Scheduling.** Packet scheduling algorithms give individual clients of ISP/NSPs the possibility to choose which traffic class will get a higher percentage of its bandwidth in real time. There exist plenty of algorithms that try to solve the problem of shaping one client's traffic in different ways [1,4,16,19,21]. However, none of the existing packet scheduling algorithms was developed to cope with DoS (and DDoS) attacks.

**DoS and DDoS Attack Mitigation.** There exist mechanisms to perform traffic shaping as a means to deal with DoS and DDoS [8, 12, 20]. In particular, [8, 12] treat DDoS attacks as congestion-control or Quality of Service (QoS) problems, which should be handled by the routers. These mechanisms classify traffic based on general network statistics, such as expected resource consumption [8] and expected traffic rate [12, 20], but do not take into account the traffic priorities of the attacked target. In most of the previous studies, traffic shaping performed by network routers does not operate according to the policy of a specific target, but according to some universal traffic shaping policy that is not optimal for all the targets. The well-known Pushback mechanism [12] is an example for such studies. The Pushback mechanism is an aggregate-based congestion control (ACC) mechanism that preferentially drops attack traffic to relieve the congestion. In Pushback aggregates' identification is based on bandwidth while in our scheme aggregates are identified by the target and comply with its priorities.

Furthermore, in Pushback, a filtering router may request adjacent upstream routers to rate-limit specified aggregates of traffic, whereas in our scheme the tuning is directly between the target and the filtering routers. This requires less involvement among the system's entities, and creates less traffic load.

The work presented in this article is based on the concept of traffic ownership, which has been introduced in [5, 6]. Traffic ownership enables ISPs/NSPs to delegate safely partial network control to their clients. This term has been presented by Bossardt et. al. [2] in the context of developing an infrastructure that facilitates general traffic control services within ISP networks. They have illustrated how this infrastructure can be applied with a DDoS mitigation scenario. However, they presented neither a comprehensive DDoS mitigation application nor an algorithm to shape the network traffic based on the target's policy.

**Our Contribution.** The scheme presented in this paper combines the objectives of packet scheduling and denial-of-service attack mitigation algorithms. On the one hand, the proposed scheme gives the targets the ability to choose online the preferred traffic classes to receive bandwidth resources (or other resources). In other words, the target is guaranteed the compliance of the arriving traffic with its priorities for bandwidth allocation between the classes. On the other hand, this scheme minimizes the damage caused by an attack to the target by two means: (a) the amount of attacking traffic reaching the target is reduced substantially, and (b) the target is allowed to continue providing the most profitable services, according to its current priorities. Note that (a) and (b) are possible even when attack traffic has similar patterns to legitimate traffic. In our scheme, the target distinguishes attack and regular traffic by an online traffic evaluation process, using the *actual utility* (the target gets from the traffic) for classification. Furthermore, the choice of traffic classes ensures that legitimate traffic is prioritized over attack traffic. Hence, attack traffic will be throttled wherever it competes with legitimate traffic over the outbound link bandwidth.

Thus, we propose an online distributed procedure for denial-of-service attack mitigation that is executed by the network's border routers. This procedure not only reduces substantially the amount of attack traffic reaching the target and allows the target to continue providing services, but also guarantees that the traffic reaching the target complies with its priorities for bandwidth allocation (for different traffic classes).

More sophisticated targets may design their online distributed traffic tuning algorithm. Other targets may benefit from receiving efficient algorithms from the network operator. We present such an algorithm and use competitive analysis to prove its worst-case performance.

# 3 System Description and Preliminaries

Our system consists of *inputs*, *routers*, and *targets*. *Input* traffic arrives to the *targets* via a network of *routers*. Traffic is sent in the network according to the Shortest-Path routing algorithm. Assuming that there is one shortest path between source and target utilized by the routing scheme, the routing scheme defines a rooted shortest-path tree from the inputs to each target.

Our basic premise is that the target leases lines from the ISP or the NSP, and the capacity of these lines is bounded by the maximal processing capacity of the target. Specifically, we suppose that the outbound capacities of the border routers is uniform, and that these capacities sum to the upper bound of what the target can process. We assume that the measurement units are bandwidth units, and the bandwidth values are considered to be integers (such as bps). Another important assumption is that the target can compute in real time a filtering policy that is based on the actual utility of the arriving traffic by using, for instance, machine learning algorithms. The target knows the utility of a certain incoming traffic class by the business activity, for instance.

In our study, the target may define two parameters: $T_{max}$ – the upper bound on the bandwidth that the target can process; and $\langle p_1,..,p_d\rangle$ – the desired vector of traffic classes' proportions, which we also call the *desired traffic proportions*. A vector component $p_i$ stands for the proportion of desired incoming traffic, corresponding to some criterion $i$, of the overall incoming traffic mix. Thus, $\sum_{i=1}^{d} p_i = 1$ holds.

For example, suppose that *Amazon* is a target providing a service and the criteria for traffic are the control bits of a TCP packet; URG, ACK, PSH, RST, SYN, FIN, NONE. The desired proportions may be $\langle 10\%, 30\%, 5\%, 5\%, 10\%, 10\%, 30\%\rangle$. These ratios represent the expected proportions of ongoing sessions, initiations of sessions, terminations of sessions, etc. of the overall incoming traffic. Note that the proportions may be expressed in much more detailed fashion, specifying even the connection's identifiers.

The traffic arriving at the target consists of two portions. One portion is the maximal traffic mix which complies with the desired proportions $\langle p_1,..,p_d\rangle$. Its bandwidth size is denoted as $T_{comp}$, where $0 \leq T_{comp} \leq T_{max}$ holds. The second portion is the extra traffic which the target is able to process. Based on these definitions we define the optimization problem as a problem in designing a distributed algorithm for filtering the inputs at the network's periphery, such that the target's incoming $T_{comp}$ is maximized ($T_{comp}$ as close as possible to $T_{max}$).

Essentially, we are dealing with traffic scheduling/shaping problem. It is possible to reformulate the filtering problem presented in this paper to a packet scheduling problem. The traffic is accumulated in the router queues if the arrival rate exceeds transmission rate. Each packet is placed in the queue corresponding to its traffic class. The scheduler repeatedly executes

the greedy filtering algorithm, presented in Section 5, and fetches packets out of these queues as computed by the greedy filtering (the inputs for the algorithm are the packets in the queues).

Assume that router $R$ receives an input vector, denoted by $\langle in_1,..,in_d\rangle$, which corresponds to the distribution of traffic classes in $R$'s incoming traffic. Similarly, $R$'s output vector, denoted by $\langle o_1,..,o_d\rangle$, corresponds to the distribution of traffic classes in $R$'s outgoing traffic. The traffic vector $\langle b_1,..,b_d\rangle$ is defined as $T_{max} \times \langle p_1,..,p_d\rangle$, and specifies the maximal proportional traffic mix desired by the target, which we also call the *desired traffic mix*. The vector components $in_i$, $o_i$ and $b_i$ stand for the amounts of traffic corresponding to some criterion $i$ in the corresponding vectors. For each input vector we define:

• *Bandwidth shortage* for component $i \in \{1,..,d\}$ is defined to be $1 - \frac{in_i}{b_i}$ when $in_i < b_i$, and 0 otherwise.

• *Minimal component* $j \in \{1,..,d\}$ is the component for which the *bandwidth shortage* is maximal (see illustration in Fig. 1).

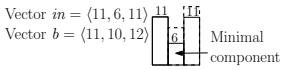



Figure 1: Component Shortage

**Definition 1.** *Consider the vectors $\langle x_1,..,x_d\rangle, \langle y_1,..,y_d\rangle$ and the desired traffic vector $\langle b_1,..,b_d\rangle$. The binary operator $\oplus_b$, denoted by $\vec{x} \oplus_b \vec{y}$, defines a vector $\vec{c} = \langle c_1,..,c_d\rangle$ where $c_i = min\{b_i, x_i + y_i\}$.*

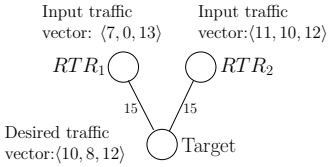
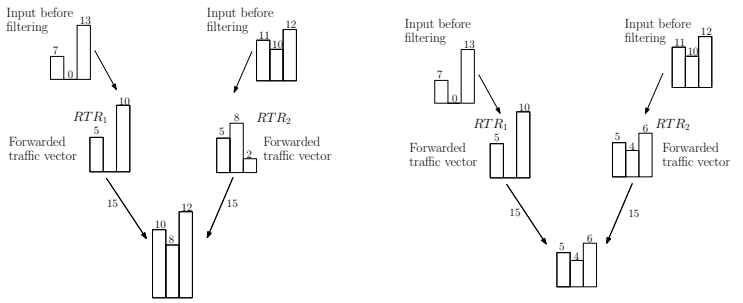



Figure 2: Two Routers Nework

Fig. 3 demonstrates the influence of the forwarding decisions of the rightmost router $RTR_2$ in the routing tree depicted in Fig. 2, over $T_{comp}$ which the target receives. In both cases, the target defines: $T_{max} = 30$ and $\langle p_1,..,p_d\rangle = \langle 33\%, 27\%, 40\%\rangle$. Hence, the target's desired incoming traffic is $\langle b_1,..,b_d\rangle = \langle 10, 8, 12\rangle$. Assume the inputs for $RTR_2$ are $\langle 11, 10, 12\rangle$, while its output bandwidth leased by the target is limited to 15. Assume the inputs for the router $RTR_1$ are $\langle 7, 0, 12\rangle$, while its output bandwidth leased by the target is also limited to 15. Assume that $RTR_1$ forwards in both cases $\langle 5, 0, 10\rangle$. Fig. 3(a) demonstrates that in case $RTR_2$ decides to forward

$\langle 5,8,2\rangle$, the final proportional vector, obtained by assembling $\langle 5,0,10\rangle$ and $\langle 5,8,2\rangle$ equals $\langle 10,8,12\rangle$ and $T_{comp} = T_{max} = 30$. In contrast, in Fig. 3(b) it is demonstrated that if $RTR_2$ decides to forward $\langle 5,4,6\rangle$, then, by the fact that $RTR_1$ does not have any input for the second component of the desired vector, the maximal final vector which complies to $\langle 33\%,27\%,40\%\rangle$ will be $\langle 5,4,6\rangle$ and, thus, $T_{comp} = 1/2 \times T_{max} = 15$.




(a) Optimal Traffic Vector

(b) Non-Optimal Traffic Vector

Figure 3: Optimal versus Non-Optimal Filtering Decisions

In the sections 4 and 5 we will present two algorithms for filtering the routers input traffic in order to maximize $T_{comp}$.

# 4 Offline Filtering Algorithm

Let us assume a network's routing tree in which every router knows all the input vectors of all the filtering routers. We present an offline filtering algorithm which determines in polynomial time the maximal $T_{comp}$ that can reach the target and the corresponding output vector for each router in the routing tree.

As an example, consider the routing tree depicted in Fig. 2. $RTR_1$ and $RTR_2$ know their own and each other's input vectors. As a first step we show linear reduction to the Maximal Flow problem.

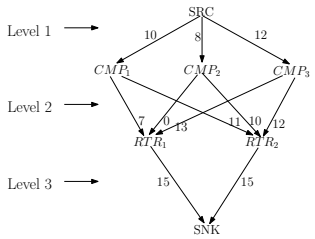



Figure 4: Example of Reduction to Maximal Flow Problem

Given the routing tree in Fig. 2, we will construct a graph $G$ (see Fig. 4) on which we will compute the maximal flow. The reduction algorithm,

according to which graph $G$ is constructed, is described in Algorithm 1. The reduction is linear in the sum of number of routers and edges in the original routing tree and number of traffic vector components.

**reduction algorithm sketch**:
*Input:* original routing tree
*Output:* graph $G$
*Steps:*
1: Create and add to $G$ a source node *SRC*
2: Create and add to $G$ one node for each traffic vector component $i$, denoted by $CMP_i$
3: Denote each border router $j$ in the original routing tree by $RTR_j$, and add it to $G$
4: Denote the target node in the original routing tree by a sink node *SNK*, and add it to $G$
5: Connect *SRC* node to each $CMP_i$ node with an edge of capacity $b_i$ (corresponds to the target's desired amount for traffic vector component $i$)
6: Connect each $CMP_i$ node to each $RTR_j$ node with an edge of capacity $in_{i,j}$ (corresponds to $RTR_j$'s input for traffic vector component $i$)
7: Add without modification the rest of the routers and edges of the original routing tree (those that were not added to the reduction graph yet)

**Algorithm 1:** Reduction to Network Flow

According to the reduction, if a flow of size $\sum_{1\leq i\leq d} b_i = T_{max}$ can pass from *SRC* to *SNK* in $G$, then it fulfills the following three conditions: **Condition (i)** The traffic flow from *SRC* to *SNK* is the desired traffic mix (ensured by the first level of edges). **Condition (ii)** The traffic flow respects the constraints imposed by the input vectors (ensured by the second level of edges). **Condition (iii)** The traffic flow respects the constraints imposed by the leased bandwidth capacities of the edges in the original routing tree (ensured by the third level of edges).

In our example (see Fig. 4), the maximal feasible flow in $G$, which maintains the desired proportions of the traffic mix is 30 ($T_{comp} = 30$).

**offline filtering** :
*Input:*
$G_{original}$ - Original routing tree
$T_{max}$ - The upper bound on target's desired traffic mix size
$\langle p_1,..,p_d\rangle$ - target's desired traffic proportions
$S_{inputs}$ - Set of the input vectors of all border routers in $G_{original}$
*Output:*
$T_{comp}$
Output traffic vector of each router in $G_{original}$
*Steps:*
1: $\langle b_1,..,b_d\rangle := T_{max} \times \langle p_1,..,p_d\rangle$
2: $G := reduction(G_{original}, \langle b_1,..,b_d\rangle, S_{inputs})$
3: int $maxFlow := Maximum\_flow\_alg(G)$
4: **If** $(maxFlow < T_{max})$ **do**
5: **Return** $compute\_T_{comp}(G_{original}, T_{max}, \langle p_1,..,p_d\rangle, S_{inputs})$
6: **Else**
7: **Return** ($G$, $T_{max}$)

**Algorithm 2:** Offline Optimal Filtering Algorithm

The *offline_filtering* algorithm is described in Algorithm 2. The inputs for this algorithm are: (a) the original routing tree denoted by $G_{original}$, (b) the upper bound on target's desired traffic mix size $T_{max}$, (c) the targets's desired traffic proportions $\langle p_1,..,p_d\rangle$, and (d) the set of the input vectors

of all border routers in $G_{original}$ denoted by $S_{inputs}$. The algorithm returns the maximal $T_{comp}$ that can reach the target in $G_{original}$, and the resulting graph $G$ (the graph constructed when applying the reduction on $G_{original}$). At the end of the *offline_filtering* algorithm, $G$ holds for each edge the traffic flowing through it. These flows form the maximal flow from *SRC* to *SNK*.

The algorithm starts in Line 1 by computing the vector of desired traffic mix $\langle b_1,..,b_d\rangle$. Given $\langle b_1,..,b_d\rangle$ and the set of input vectors, $S_{inputs}$, we apply in Line 2 the reduction algorithm on $G_{original}$, that produces the graph $G$. The next step executes one of the known polynomial algorithms for the Maximal Flow Problem, such as the algorithms for Maximal Flow computation described in [7, 9], to derive the maximal flow in $G$. We assume that during the Maximal Flow algorithm every edge in $G$ is assigned the traffic flowing through it. If the maximal feasible flow in $G$ is smaller than $T_{max}$, then we execute the *compute_*$T_{comp}$ algorithm (Algorithm 3) for finding the maximal flow that maintains the desired proportions of the traffic mix (Line 5). If the maximal feasible flow equals $T_{max}$, then the algorithm returns $T_{max}$ and the resulting graph $G$ and terminates.

**compute** $T_{comp}$:
*Input:*
$G_{original}$ - Original routing tree
$T_{max}$ - The upper bound on target's desired traffic mix size
$\langle p_1,..,p_d\rangle$ - target's desired traffic proportions
$S_{inputs}$ - Set of the input vectors of all border routers in $G_{original}$
*Output:*
$T_{comp}$
Output traffic vector of each router in $G_{original}$
*Steps:*

1: **double** $L := 0, R := T_{max}$
2: **Repeat**
3: **int** $M := round(L + \frac{(R-L)}{2})$
4: $\langle b_1,..,b_d\rangle := M \times \langle p_1,..,p_d\rangle$
5: $G^M := reduction(G_{original}, \langle b_1,..,b_d\rangle, S_{inputs})$
6: int $maxFlow := Maximum\_flow\_alg(G^M)$
7: **If** $(maxFlow == M)$ **do**
8: $L := M$
9: **Else**
10: $R := M$
11: **Until** $R - L \leq 1$
12: **Return** $(G^M, M)$

**Algorithm 3:** Binary Search Algorithm

*compute_*$T_{comp}$ algorithm appears in Algorithm 3. This algorithm is a binary search algorithm which receives: (a) $G_{original}$, (b) $T_{max}$, (c) $\langle p_1,..,p_d\rangle$ and (d) $S_{inputs}$. *compute_*$T_{comp}$ finds the maximal $T_{comp}$ and the corresponding graph $G$ by scaling the traffic mix size (possible $T_{comp}$) up and down. The initial range for search of the maximal $T_{comp}$ is $[L,\ R] = [0,\ T_{max}]$. Line 3 computes the midpoint $M$, which is used in Line 4 for computing the corresponding proportional vector of traffic components' amounts. In Line 5 we apply the reduction and derive new $G^M$ ($G$ that corresponds to a candidate $T_{comp}$ that equals $M$). If the maximum flow, computed in Line 6,

is equal to $M$, then the next search interval is $[M,\ R]$, otherwise, the next search interval is $[L,\ M]$. The algorithm converges when $R-L \leq 1$ (Line 11).

**Algorithm's Complexity.** The complexity of the *offline_filtering* algorithm depends on the complexity of the reduction algorithm and that of the Maximal Flow algorithm. We denote the complexity of the Maximal Flow Algorithm by $MF(n,m)$ for a graph with $n$ nodes $m$ edges. Assume that the number of routers in $G_{original}$ is $n$ and the number of edges in $G_{original}$ is $m$. We have that $n' = n+d+2$ and $m' = m+k\times d+d$ ($k$ is the number of border routers in $G_{original}$) are the numbers of nodes and edges in $G^M$, respectively. The number of iterations during binary search in *compute_*$T_{comp}$ is bounded by $O(\log T_{max})$ and at each iteration we perform a reduction and compute a Maximal Flow. The reduction algorithm is linear in $n'+m'$. Hence, the complexity of the *offline_filtering* algorithm is $O((n'+m'+MF(n',m'))\times \log T_{max})$. For example, if Edmond Karp maximal flow algorithm [7] is used, then $MF(n',m') = O(n'(m')^2)$ and the overall complexity of the *offline_filtering* algorithm is $O(n'(m')^2 \times \log T_{max})$.

# 5 Online Filtering Algorithm

In this section we present a greedy online filtering algorithm executed independently by each border router. The *greedy_online_filtering* algorithm does not consider any collaboration between the routers in the network. For the execution of the *greedy_online_filtering* algorithm, each filtering router needs to know its output bandwidth capacity, its input traffic and the traffic vector desired by the target. We limit the output bandwidth capacities of the border routers so that their output bandwidth sum to $T_{max}$. We apply this limit in order to avoid overflow at the target.

**Greedy Filtering Algorithm.** The *greedy_online_filtering* algorithm (Algorithm 4) is used by each border router independently to compute the maximum possible amount of traffic to be forwarded, while trying to maximize the portion of traffic that complies with the target's desired proportions $\langle p_1,..,p_d\rangle$. Namely, the goal of the *greedy_online_filtering* algorithm is to maximize $T_{comp}$.

The algorithm inputs are: (a) the router's input traffic vector $\langle in_1,..,in_d\rangle$, (b) the router's output traffic vector $\langle out_1,..,out_d\rangle$, which is initially set to $\langle 0,..,0\rangle$, (c) the target's desired traffic proportions $\langle p_1,..,p_d\rangle$, and (d) the router's unutilized output bandwidth - $bandwidth$, which is initially set to $T_{max}$/(number of border routers).

The *greedy_online_filtering* algorithm is a recursive algorithm. In each iteration of the recursion, the output traffic vector is increased. The increase is calculated based on $\langle p_1,..,p_d\rangle$, the remaining unutilized output bandwidth and the remaining input traffic. The idea is to compute in each iteration, the maximal traffic mix which complies with $\langle p_1,..,p_d\rangle$, and thus

try to maximize $T_{comp}$. In each iteration, all the inputs matching the minimal component of the remaining inputs, $\langle in_1,..,in_d\rangle$, are added to the output traffic vector $\langle out_1,..,out_d\rangle$.

The first step of the algorithm verifies that the filtering has not completed constructing the output vector. If it has completed, the execution terminates (line 1) and the output traffic vector $\langle out_1,..,out_d\rangle$ stores the traffic mix that the router should forward. The algorithm is divided into four major parts. The first part (lines 2 - 3) calculates the optimal output vector given $\langle p_1,..,p_d\rangle$ and the free bandwidth (*bandwidth*). The optimal output vector is the maximal traffic mix which matches the desired proportions, and its total size is equal to the free bandwidth. In the second part we find the minimal component (lines 4 - 11). The third part updates the input and output vectors (lines 12 - 23). Lines 14 - 17 calculate additions to the output vector in the current recursive call. The bandwidth, the input ($\langle in_1,..,in_d\rangle$) and the output ($\langle out_1,..,out_d\rangle$) vectors are updated accordingly (lines 18 - 23). The last part consists of recalculation of the $\langle p_1,..,p_d\rangle$ of the remaining mix. These new proportions are to be used in the next round for calculating the optimal output vector (in lines 2 - 3).

The method *is_done* verifies whether the filtering method has completed to construct the output vector. The filtering has completed if either we have no input left, or no unutilized bandwidth left or all $p_i = 0$.

**Algorithm's Complexity.** Complexity of the method *is_done* is linear in the length of $\langle in_1,..,in_d\rangle$ and $\langle p_1,..,p_d\rangle$, thus, its complexity is $O(d)$. The complexity of every **Foreach** loop in the algorithm is also $O(d)$. Therefore, to bound the algorithm's complexity it is needed to bound the number of recursion calls. The recursion continues until either we have no input left, or no unutilized bandwidth left or all $p_i = 0$. In each iteration, all the inputs matching one of the input's components are added to the output traffic vector, therefore, the number of the recursion calls is bounded by the number of traffic vector components $d$. Hence, the overall complexity of the algorithm is $O(d) \times d = O(d^2)$.

**Competitive ratio.** In this section we compare the optimal offline algorithm and the online algorithm, and we prove the worst case performance of the *greedy_online_filtering* algorithm. We compare the optimal $T_{comp}$ assuming all the routers in the network execute the offline filtering algorithm (which gives the strongest possible performance guarantee), with the $T_{comp}$ assuming every border router executes the *greedy_online_filtering* algorithm independently.

**Two-Level Binary Routers Tree.** We will show the intuition behind the analysis using a simple case of two-level binary routers tree (see Fig. 5). We assume that the target has leased enough bandwidth capacity on the edges $\langle RTR_1,target\rangle$ and $\langle RTR_2,target\rangle$, hence the edges are able to forward the whole desired traffic mix to the target. So, the sum of bandwidth capacities of the edges $\langle RTR_1,target\rangle$ and $\langle RTR_2,target\rangle$ equals $T_{max}$. It

**greedy online filtering**:
*Input:*
$\langle in_1,..,in_d\rangle$ - router's input traffic vector
$\langle out_1,..,out_d\rangle$ - router's output traffic vector, initially equals $\langle 0,..,0\rangle$
$\langle p_1,..,p_d\rangle$ - target's desired traffic proportions
$bandwidth$ - unutilized output bandwidth, initially equals $T_{max}$/#border routers
*Ouput:*
$\langle out_1,..,out_d\rangle$ - router's computed output vector
*Steps:*

**If** is_done($\langle in_1,..,in_d\rangle$, $bandwidth$, $\langle p_1,..,p_d\rangle$) **do Return**
**Foreach** $p_i \in \langle p_1,..,p_d\rangle$ **do**
    $optimal_i := bandwidth \times p_i$
$minimalComponent := 0$
$minimalPercentOfOptimal := 1$
**Foreach** $in_i \in \langle in_1,..,in_d\rangle$ **do**
    **If** $(p_i \neq 0)$ **do**
        $percent := in_i / optimal_i$
        **If**$(percent < minimalPercentOfOptimal)$ **do**
            $minimalComponent := i$
            $minimalPercentOfOptimal := percent$
**Foreach** $in_i \in \langle in_1,..,in_d\rangle$ **do**
    **If**$(in_i \neq 0)$ **do**
        **If** $(i = minimalComponent)$ **do** /* Empty the minimal Component. */
            $outComponent := min(optimal_i, in_i)$
        **Else** /* Output the appropriate ratio of the optimal amount. */
            $outComponent := optimal_i \times minimalPercentOfOptimal$
        /* Update input vector. */
        $in_i := in_i - outComponent$
        /* Update output vector. */
        $out_i := out_i + outComponent$
        /* Update bandwidth. */
        $bandwidth := bandwidth - outComponent$
/* Update the proportions. */
$percentagesSum := 0$
**Foreach** $p_i \in \langle p_1,..,p_d\rangle$ **do**
    **If** $(in_i \neq 0)$ **do**
        $percentagesSum := percentagesSum + p_i$
**Foreach** $p_i \in \langle p_1,..,p_d\rangle$ **do**
    **If** $(in_i \neq 0)$ **do**
        $new\_p_i := p_i \times (1/percentagesSum)$
$greedy\_online\_filtering(\langle in_1,..,in_d\rangle, \langle out_1,..,out_d\rangle, \langle new\_p_1,..,new\_p_d\rangle, bandwidth)$

**Algorithm 4:** Greedy Filtering Method

is also assumed that the output bandwidth capacity of $RTR_1$ and $RTR_2$ is uniform, therefore, the capacity of each of the edges $\langle RTR_1, target\rangle$ and $\langle RTR_2, target\rangle$ equals $\frac{1}{2} \times T_{max}$.

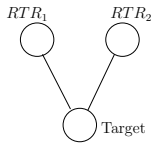



Figure 5: Two-Level Binary Routers Tree

Assume $RTR_1$ and $RTR_2$ execute the *greedy_online_filtering* algorithm. Then, the maximal proportional traffic mix that each of them can produce is $\frac{1}{2} \times T_{max}$. But what if $RTR_1$ (without loss of generality) has not received enough input for some component $i$ of the traffic vector? In such a case, the proportional traffic mix produced by $RTR_1$ is smaller than $\frac{1}{2} \times T_{max}$. Since there still remains unutilized bandwidth, $RTR_1$ fills it with other components of the traffic vector.
Finally, for every component of the traffic vector, the target receives from each of the routers the minimum between (a) all the existing input (which is the maximal possible output for this component) and (b) half of the desired amount. In the worst case, the traffic mixes produced by the routers do not compensate each other's traffic mix to construct a proportional mix. Hence, the proportional mix (out of the whole traffic) that the target finally receives is at least the sum of the routers' $T_{comp}$.
**Two-Level $l$-Routers Tree.** We now present a formal analysis of two-level $l$ routers tree. As in the previous case, the sum of bandwidth capacities leased by the target equals $T_{max}$ and the capacity of edges is uniform. Thus, the output bandwidth of each router is $\frac{1}{l} \times T_{max}$.

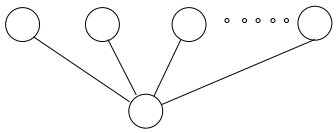


Figure 6: Two-Level $l$-Routers Tree

**Theorem 1.** $(a)$ *For each traffic vector component i,* $1 \leq i \leq d$*, every router* $R_k$*,* $1 \leq k \leq l$*, outputs* $o_i = min\{\frac{1}{l} \times b_i + \delta', in_i\}$*.* $(b)$ $T_{comp} \geq \frac{1}{l} \times T_{comp}^{opt}$ *produced by the offline algorithm.*

**Proof.** The output traffic vector of each of the routers is denoted by $\langle o_1, .., o_d\rangle$, and the proportional traffic vector which each of the routers produces is denoted by $\langle m_1, .., m_d\rangle$. Each router, for each component $i$, designates $\frac{1}{l} \times T_{max} \times p_i = \frac{1}{l} \times b_i$ output bandwidth. Note, that if the total input bandwidth is less than $R$'s output bandwidth capacity, then $R$ outputs all its inputs. For proving (a) we examine two cases:

**Case 1** For every traffic vector component $i$, $in_i \geq \frac{1}{l} \times b_i$. In this case, $R$ uses all the output bandwidth capacity for forwarding the proportional traffic mix $\langle m_1,..,m_d \rangle$. Hence, it holds that for each component $i$, $R$ outputs $o_i = m_i = \frac{1}{l} \times b_i$.
**Case 2** There exists a traffic vector component $i$ for which $in_i < \frac{1}{l} \times b_i$. According to the *greedy_online_filtering* algorithm, $R$ constructs the proportional vector $\langle m_1,..,m_d \rangle$ in the first level. $\langle m_1,..,m_d \rangle$ is constructed according to the minimal component $i$, denoted by $mc$. Since $in_{mc} < \frac{1}{l} \times b_{mc}$, the bandwidth of capacity $\delta_{mc} = \frac{1}{l} \times b_{mc} - in_{mc}$ is now free for being used by the other $d$ - 1 components.

In the next levels of the recursion, the free space $\delta_{mc}$ is divided among $d-1$ components. Each component $i \neq mc$ is being added to the output vector until either there is no input left, or there is no unutilized bandwidth left. Hence, the output amount of bandwidth for component $i$ is $o_i = min\{\frac{1}{l} \times b_i + \frac{\delta_{mc}}{X}, in_i\}$, where $\frac{1}{X}$ is a fraction of $\delta_{mc}$, corresponding to $p_i$.
From **Case 1** and **Case 2** we conclude that $(a)$ holds.

Let us denote the output vector of router $R_j$, $1 \leq j \leq l$ by $\vec{o_{R_j}}$, and the size of the proportional part of $\vec{o_{R_j}}$ by $T_{comp}^{R_j}$. In this case, the traffic vector that reaches the target is $\vec{o_{R_1}} \oplus_b \vec{o_{R_2}} \oplus_b .. \oplus_b \vec{o_{R_l}} = \langle o_{f_1},..,o_{f_d} \rangle$. According to $(a)$, for each component $i$, every router $R$ forwards $o_i = min\{\frac{1}{2} \times b_i + \delta', in_i\}$. Note, that $\frac{1}{l} \times b_i$ is $\frac{1}{l}$ of the optimal amount of bandwidth for component $i$ that the target desires, and $in_i$ (when $in_i < \frac{1}{l} \times b_i$) is the optimal amount that can reach the target (because there is no filtering scheme in which a router can send more than it receives). Therefore, the sizes of the output vectors $\vec{o_{R_j}}$s are at least $\frac{1}{l}$ of the size of the maximal proportional traffic mix that can reach the target.

In the worst case, the vectors $\vec{o_{R_k}}$, $1 \leq k \leq l$, do not compensate each other's traffic mix to construct a proportional mix. So, the proportional mix that the target finally receives is the output vector of only one of the routers. However, we have shown that the sizes of each of $\vec{o_{R_k}}$ is at least $\frac{1}{l}$ of optimal $T_{comp}$. Consequently, the size of the proportional traffic part of $\vec{o_f}$ is at least $\frac{1}{l} \times$ optimal $T_{comp}$ ($T_{comp} \geq \frac{1}{l} \times T_{comp}^{opt}$). Hence, $(b)$ holds as well. □
***k*-Level *l*-Routers Tree.** The last step is to generalize the analysis to trees with $k$ levels (and $l$ routers in the top level).

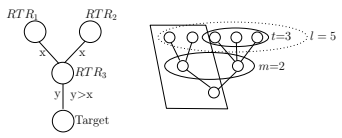



Figure 7: $k$-Level $l$-Routers Tree

**Theorem 2.** *The target receives $T_{comp} \geq \frac{1}{l} \times m \times T_{comp}^{opt}$, where $m$ is the number of successors to the border routers in the tree.*

**Proof sketch.** Initially, we examine the first level. Every structure of "brothers" (among the border routers) and "father" (their successor) operate as described previously (in the two-level $l$-routers tree). Thus, if there are $t$ ($t \leq l$) "brothers", then $T_{comp}$ reaching the "father" is at least $\frac{1}{t}$ ($\frac{1}{t} \geq \frac{1}{l}$) of the maximal possible $T_{comp}$. Therefore, if there are $m$ "fathers", then altogether they receive proportional traffic mix of size at least $\frac{1}{l} \times m \times T_{comp}^{opt}$.

Let us examine all "father" routers and downward. They receive proportional traffic mixes of sizes at least $\frac{1}{l} \times$ maximal possible $T_{comp}$ from their predecessors. Let us recall the assumptions that a target leases lines wide enough to forward $T_{max}$, and that the output bandwidth capacities of the border routers are equal. Therefore, as the tree width reduces towards the target, the target's leased capacity on the edges increases. So, in this case, every router $R$, starting from the "father" routers and downward, is able to forward the whole traffic which reaches it. The target, thus, receives aggregation of all traffic mixes from the top of the tree. This aggregation of traffic contains a proportional traffic mix of size at least $\frac{1}{l} \times m \times T_{comp}^{opt}$. □

# 6 Conclusions

In this paper, we presented two algorithms that allow attack targets to dynamically filter their incoming traffic based on a distributed policy. The proposed algorithms defend the target against DoS and distributed DoS (DDoS) attacks and simultaneously ensure that it continues to receive valuable users' traffic. The goal of the algorithms is to maximize the amount of filtered traffic forwarded to the target from the network according to the filtering policy of the target. The filtering algorithms are activated at the ISP's or at the NSP's [border] routers when the target detects a DDoS attack. We have proposed a distributed online filtering algorithm and demonstrated its worst-case performance. We also studied the offline version of the problem and presented an optimal polynomial-time algorithm.

# References


[1] Bennett J. C.R., Zhang H., "WF 2 Q: Worst-case Fair Weighted Fair Queueing", *INFOCOM '96*, Vol. 1, pp. 120-128, 1996.

[2] Bossardt M., Dübendorfer T., Plattner B., "Enhanced Internet security by a distributed traffic control service based on traffic ownership", *Journal of Network and Computer Applications*, Vol. 30, No. 3, pp. 841-857, 2007.

[3] Cisco, http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfpolsh.html

[4] Dekeris B., Narbutaite L., Adomkus T., "A New Adaptive Fair Queueing (AFQ) Scheduler for Support SLA", *Proc. of 29th International Conference on ITI*, 2007.

[5] Dolev S., "Novel Dynamic Firewall for NSP Networks Deutsche Telekom", *European Patent application*, 08003000.0-1244, 19.2.08. IL/19.02.07/ILA 18142707.

[6] Dübendorfer T., Bossardt M., Plattner B., "Adaptive distributed traffic control service for DDoS attack mitigation", *Proc. of 19th IEEE International Parallel and Distributed Processing Symposium*, pp. 8, 2005.

[7] Edmonds J., Karp R. M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", *Journal of the ACM*, Vol. 19, No. 2, pp. 248-264, 1972.

[8] Garg A., Narasimha Reddy A. L., "Mitigation of DoS attacks through QoS regulation", *10th IEEE International Workshop on Quality of Service*, pp. 45-53, 2002.

[9] Goldberg A. V., Tarjan R. E., "A new approach to the maximum-flow problem", *Journal of the ACM*, Vol. 35, No. 4, pp. 921-940, 1988.

[10] Huang Y., Geng X., Whinston A. B., "Defeating DDoS attacks by fixing the incentive chain", *ACM Transactions on Internet Technology*, Vol. 7, No. 1, Article No. 5, 2007.

[11] Hussain A., Heidemann J., Papadopoulos C., "A Framework for Classifying Denial of Service Attacks", *Proc. of ACM SIGCOMM Conference*, pp. 99110, 2003.

[12] Ioannidis J., Bellovin S. M., "Implementing Pushback: Router-Based Defense Against DDoS Attacks", *Proc. of Network and Distributed System Security Symposium*, 2002.

[13] Intel, http://www.intel.com/it/pdf/Implementing-WAN-QOS.pdf

[14] Juniper, http://www.juniper.net/us/en/local/pdf/whitepapers/2000180-en.pdf

[15] Moore D., Voelker G. M., Savage S., "Inferring internet denial-of-service activity", *Proc. of 10th Usenix Security Symposium*, Vol. 10, pp. 2-2, 2001.

[16] Parekh A. K., Gallager R. G., "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 3, pp. 344 - 357, 1993.

[17] Peng T., Leckie C., Ramamohanarao K., "Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems", *ACM Computing Surveys*, Vol. 39, No. 1, Article No. 3, 2007.

[18] Poggi N., Moreno T., Berral J. L., Gavaldà R., Torres J., "Web Customer Modeling for Automated Session Prioritization on High Traffic Sites", *Lecture Notes in Computer Science*, Vol. 4511, Book User Modeling 2007, pp. 450-454, 2007.

[19] Rahbar A.G.P., Yang O., "LGRR: A new packet scheduling algorithm for differentiated services packet-switched networks", *Computer Communications*, Vol. 32, No. 2, pp. 357-367, 2009.

[20] Wongvivitkul C., Ngamsuriyaroj S., "The Effects of Filtering Malicious Traffic under DoS Attacks", *APAN 24th Meeting in Xi'An*, 2007.

[21] Yin H., Wang Z., Sun Y., "Enhanced WFQ Algorithm with (m,k)-Firm Guarantee", *Lecture Notes in Computer Science*, Vol. 3605, Book Embedded Software and Systems, pp. 339-346, 2005.

[22] Yue C., Wang H., "Profit-aware Admission Control for Overload Protection in E-commerce Web Sites", *15th IEEE International Workshop on Quality of Service*, pp. 188-193, 2007.

# 7 Appendix

## 7.1 Extensions

Many modern distributed security systems base on exchange of information between the system nodes. We propose two online filtering schemes that base on correlation between the filtering routers.

In the first scheme the execution consists of cycles. During each cycle the input traffic vectors are being forwarded down the routing tree, starting from the border routers to the target. When the target knows all the input traffic vectors in the routing tree, it instructs its predecessors what output traffic vectors to forward. These instructions percolate up the tree to the border routers.

The top level is indexed 0, the successing level is indexed 1, and so on. Every batch consists of the following actions:
**(I)** Router in level 0 sends a notification of its input traffic vector to its successor in level 1.
**(II)** Router in level $i$, after receiving notifications from all predecessors (in level $i-1$), sends to its successor (in level $i+1$), a notification of its inputs (sum of all notifications from the predecessors).
**(III)** The target (router in the last level), after receiving notifications from all predecessors, sends to each of its predecessors, the output traffic vector they ought to send him.

**(IV)** Router in level $i$, after receiving the command from its successor, calculates and sends to each of its predecessors, the output traffic vector they ought to forward him.
**(V)** Router in level 0, after receiving the command from its successor in level 1, sends an output traffic vector as instructed.
**(VI)** Router in level $i$, after receiving the traffic vectors from its predecessors in level $i-1$, constructs and sends an output traffic vector as instructed by its successor in level $i+1$.

**Observation 1.** *The target has a full view (offline view) of the traffic inputs to the routing tree, therefore, it can instruct the preceding routers what to forward in order to assemble the optimal traffic mix. Hence, optimality is ensured.*

The algorithm executed by each router in the tree is presented in Algorithm 5.

**feedback batch**:
1: status := first stage
2: **If** no predecessors **And** status = first stage **do**
3: send $\langle in_1,..,in_d\rangle$ to successor

/* Upon receiving $\langle in_1,..,in_d\rangle$ from predecessor */
4: **If** received $\langle in_1,..,in_d\rangle$ from all predecessors **do**
5: $\langle in_1,..,in_d\rangle = \sum_{in_i \text{ from predecessor}_i} \langle in_{i_1},..,in_{i_d}\rangle$
6: send $\langle in_1,..,in_d\rangle$ to successor
7: **If** status = second stage **do**
8: status := third stage
9: **Else**
10: status := second stage
11: **If** no successors **And** status = second stage **do**
12: **For** $v \in$ predecessors **do**
13: calculate v's output vector $\langle out_{v_1},..,out_{v_d}\rangle$
14: send $\langle out_{v_1},..,out_{v_d}\rangle$ to v

/* Upon receiving $\langle out_{v_1},..,out_{v_d}\rangle$ from successor */
15: **For** $v \in$ predecessors **do**
16: calculate v's output vector $\langle out_{v_1},..,out_{v_d}\rangle$
17: send $\langle out_{v_1},..,out_{v_d}\rangle$ to v
18: **If** no predecessors **do**
19: send $\langle out_{v_1},..,out_{v_d}\rangle$ to successor

**Algorithm 5:** Filtering Algorithm with Feedback Based on Batches

In the second scheme the collaboration is limited by depth parameter. We assume that depth in the routing tree is measured in levels starting at level 0 - the border level. We say that a routing tree has collaboration at depth $x$, if every router $R$, up till depth $x$ knows the input traffic vectors of all the routers in its subtree (the routers in $R$'s subtree also know all the input traffic vectors in its subtree).

For $x = 0$ there is no collaboration in the system (only the border routers know the input traffic vectors to their empty subtrees). For $x = n$, where $n$ is the maximal level in the tree, we get the offline filtering problem. For

$x = n - 1$, each predecessor, $P$, of the target (and all routers in $P$'s subtree) knows the input vectors in its subtree. Each of the subtrees execute the offline filtering algorithm, and the roots of these subtrees receive the maximal possible desired traffic mix (at their level). However, it cannot be guaranteed that the construction of these traffic mixtures will result in the maximal possible desired traffic mix at the target (it is still possible, as in the worst case of the online filtering algorithm in Section 5, that all except one of the predecessors' output traffic mixtures will be totaly useless for the target).