Filtering Sources of Unwanted Traffic

Fabio Soldo, Karim El Defrawy, Athina Markopoulou
University of California, Irvine

Balachander Krishnamurthy, Jacobus van der Merwe
AT&T Labs-Research

***Abstract*— There is a large and increasing amount of unwanted traffic on the Internet today, including phishing, spam, and distributed denial-of-service attacks. One way to deal with this problem is to filter unwanted traffic at the routers based on source IP addresses. Because of the limited number of available filters in the routers today, aggregation is used in practice: a single filter describes and blocks an entire range of IP addresses. This results in blocking of all (unwanted and wanted) traffic generated from hosts with IP addresses in that range. In this paper, we develop a family of algorithms that, given a blacklist containing the source IP addresses of unwanted traffic and a constraint on the number of filters, construct a set of filtering rules that optimize the tradeoff between the unwanted and legitimate traffic that is blocked. We show that our algorithms are optimal and also computationally efficient. Furthermore, we demonstrate that they are particularly beneficial when applied to realistic distributions of sources of unwanted traffic, which are known to exhibit spatial and temporal clustering.**

I. Introduction

It is well known that there is a large and increasing amount of unwanted traffic on the Internet today, including flooding and other denial-of-service attacks, scanning, phishing, etc. The sources of unwanted traffic are typically compromised hosts infected with malicious code. One mechanism that is used today to prevent unwanted traffic from reaching the victims, is to use access control lists (ACLs) at the routers to block packets that are considered unwanted. ACLs are rules that can classify packets according to a combination of fields in the IP header. In this paper, we are interested in filtering based on the source IP addresses of unwanted traffic [1]–[3]. We assume that these sources are known and given to us as a (black)list. Several such blacklists are constructed today from firewall and intrusion detection system logs, made publicly available [4] or offered as a service [5], [6]. ISPs [3] and organizations also maintain their own private blacklists based on historical data relevant to their own networks. Such systems that distinguish between the unwanted and legitimate traffic are necessary to construct blacklists but are considered out of the scope of this paper. The input to the problem we study here is a blacklist, i.e. a list of sources of unwanted traffic, that is considered given and accurate.

Once a blacklist is given, filtering can be used as a first, coarse but cheap, step of defense [1], [3]. Filtering based on a blacklist involves constructing a set of ACL rules to block unwanted traffic so as to meet certain criteria. A router would ideally block each source in a blacklist with a single ACL rule. Unfortunately, the number of ACLs that can be instantiated in today's routers is orders of magnitude smaller than the number of attackers. A practical approach used instead is to block ranges (typically prefixes) of addresses. This aggregation has the advantage that it reduces the number of filters and the disadvantage that it also blocks traffic originating from legitimate IP addresses in the blocked range. For a given number of filters, the main tradeoff involved in filtering is between the number of sources of unwanted traffic successfully blocked and the legitimate traffic accidentally blocked (also called "collateral damage"). Clearly, there are several parameters that affect the effectiveness of filtering, including the number of filters available compared to the number of attackers, and the distribution of attack sources in the IP address space.

The spatial and temporal behavior of sources of unwanted traffic have been studied recently by several measurement papers [7]–[9]. These papers found that the sources of unwanted traffic (or "Internet background radiation" as it was called in [7]) exhibit clustering in the IP address space, meaning that most sources of unwanted traffic are concentrated in a few ranges. This has also been consistent with our experience from studying various blacklists of unwanted traffic [4], [5].

Motivated by this observation, we construct compact rules that block entire ranges of source IP addresses. In particular, we develop two algorithms that take as input a blacklist of IP addresses and select ranges to block. Algorithm FILTER-ALL-STATIC blocks all blacklisted sources so as to minimize the collateral damage. Algorithm FILTER-SOME-STATIC blocks some of the sources, trading-off a decrease in the number of blacklisted IP addresses filtered for a decrease in collateral damage. Our algorithms are optimal with respect to the above criteria and also computationally efficient. We also extend these algorithms, to FILTER-ALL-DYNAMIC and FILTER-SOME-DYNAMIC respectively, so as to deal with time-varying blacklists, i.e., instances of a blacklist at different times. The algorithms bring significant benefit when applied to blacklists with inherent clustering of addresses, as it is the case in practice [7]–[9]; we demonstrate that through simulation using realistic models for the spatial distribution of addresses in a blacklist [10].

This paper builds on our previous work [11], where we looked at a related problem: how to filter attack sources at the granularity of individual hosts or gateway tiers so as to minimize the amount of legitimate traffic dropped,

subject to constraints in the number of filters and the victim's access bandwidth. The problem was a variation of the two-dimensional knapsack and its solution had high complexity. In this paper, we made the following key observation: by thinking source addresses as numbers in the range $[0, 2^{32}-1]$, we can exploit the ordering in the one-dimensional IP address space and group ranges of consecutive addresses. This structure allows us to develop greedy optimal algorithms, which is not possible if arbitrary grouping of attackers is allowed. Furthermore, filtering based on the IP source address is more in line with the practice of filtering using ACLs and prefix-based filtering.

The rest of the paper is structured as follows. Section II gives the rationale behind the filtering problems studied here, an overview of the algorithms, and introduces the notation. Section III presents the four algorithms and proves their optimality. Section IV applies the algorithms to realistic models of input blacklists that exhibit different degrees of clustering and density, and discusses the tradeoffs involved. Section V concludes the paper.

II. Problem Formulation

A. Definitions and Notation

Let us first define the notation, used throughout the paper, also summarized in Table I.

Source IP addresses. Every IP address A is a 32-bit sequence and therefore can be thought as an integer number in the range $[0, 2^{32}-1]$. Let us consider a set of consecutive IP addresses in that range, $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$, in increasing order $A_i < A_{i+1}$. $[A_i, A_j] \subseteq \mathcal{A}$ denotes the set of consecutive addresses from A_i to A_j (included).

Blacklists. The input to our algorithms is a blacklist. A blacklist (BL) is simply a list of N unique source addresses that are known to be sources of unwanted traffic. An address is considered "bad" if it appears in a blacklist or "good" otherwise. The process that distinguishes good from bad traffic is necessary to construct a reliable blacklist but is out of the scope of this paper. In this paper, blacklists are considered known and accurate. We indicate with $\mathcal{B} = \{b_1, \ldots, b_N\} \subset \{1, \ldots, m\}$ the set of *indices* used to denote bad addresses, i.e., A_{b_i}, $\forall i \in \{1, 2, \ldots, N\}$ denote a single address reported in the blacklist. Similarly, $\mathcal{G} = \{g_1, \ldots, g_{m-N}\} \subset \{1, \cdots, m\}$ is the set of indices used to denote good addresses. If we have several instances of the blacklist at different times, we use the notations BL_{t_i} and N_{t_i},to indicate the blacklist, and the number of bad addresses at time t_i respectively.

Filters. A filter, in this paper, is a rule that specifies that all addresses in the range $[A_i, A_j]$ should be blocked. We use the terms address range/cluster/filter interchangeably. F_{max} denotes the maximum number of filters that can be deployed. $f \leq F_{max}$ denotes the number of filters actually used.

Address Weight. In the basic version of the problem, an address is either bad or good, depending on whether it appears or not in a blacklist respectively. In other versions of the problem, we may also want to assign a weight w_i to address A_i. This weight can capture the amount of good/bad traffic originating from the IP address space that is filtered out; or it can express probability/certainty about the "goodness"/"badness" of the address, e.g. based on historical data; or it can express policy for treating various addresses, e.g. to ensure that an important customer is never blocked. More details on the assignment of w_i will be provided later in the discussion of the FILTER-SOME-* algorithms.

BL	Blacklist: a list of "bad" addresses
N	Number of unique addresses in BL
BL_{t_i}	Instance of the blacklist at time t_i
N_{t_i}	Number of addresses in BL_{t_i}
$\mathcal{B} = \{b_1, \ldots, b_{N_{t_i}}\}$	set of indices used for bad addresses
$\mathcal{G} = \{g_1, \ldots, g_{m-N_{t_i}}\}$	set of indices used for good addresses
$\tilde{C}_{l,r}$ (or simply $C_{l,r}$)	Collateral Damage from filter $[A_l, A_r]$ (when A_l, A_r are both bad)
f	Number of filters used
F_{max}	Maximum number of available filters
w_i	weight assigned to address A_i
$R_{l,r} \in \{0, 1\}$	variable indicating if filter $[A_l, A_r]$ is used

TABLE I
NOTATION

Collateral Damage. We use $\tilde{C}_{l,r}$ to denote the collateral damages caused by the filter $[A_l, A_r]$. In this paper, we use this term as a measure of the number of legitimate hosts/source IP addresses that are blocked by a filter.[1] If the filter $[A_l, A_r]$ is not used, or it does not encompass any good address, $\tilde{C}_{l,r} = 0$; otherwise, $\tilde{C}_{l,r} > 0$. In general, each of the addresses A_l and A_r can be good or bad. When both A_l and A_r are bad addresses, (as it will turn out to be in the optimal solution), we use $C_{l,r}$, to indicate the collateral damage caused by the filter $[A_l, A_r]$.

The obvious choice for $\tilde{C}_{l,r}$ is simply the total number of good addresses included between A_l and A_r,

$$\tilde{C}_{l,r} = \sum_{l \leq i \leq r} \mathbb{I}_{\mathcal{G}}(i) \tag{1}$$

where, $\mathbb{I}_{\mathcal{G}}$ is the indicator function of $\mathcal{G}$. However, when weights are assigned to addresses, the collateral damage is:

$$\tilde{C}_{l,r} = \sum_{l \leq i \leq r} w_i \mathbb{I}_{\mathcal{G}}(i) \tag{2}$$

If $w_i = 1$ for every "good" address and $w_i = 0$ for every "bad" address, we have that $\tilde{C}_{l,r}$ represents exactly the number of good addresses in the interval $[A_l, A_r]$.

B. Rationale and Overview of Filtering Problems

Protecting a network from unwanted traffic is a complex problem with several different aspects [1], [3]. The first step is to distinguish good from bad sources, based e.g. on intrusion detection systems and historical data; this part is out of the scope of this paper and is considered as a pre-processing step that constructs a blacklist. The second

[1] In contrast, in previous papers [1], [11], the term "collateral damage" referred to the volume of legitimate traffic filtered out.

step, which is the focus of this work, is the construction of a compact set of filtering rules, so as to achieve certain goals. Goals may include a combination of the following objectives: filter all (or most) bad addresses; cause low or no collateral damage to legitimate traffic; stay within the budget in the number of filters; apply a policy to favor/punish specific addresses.

In this work, we formulated a family of filtering problems, each targeted to a different goal, and we develop optimal greedy (thus computational efficient) algorithms that achieve these goals. We expect these algorithms to be used as building blocks in a larger filtering-based defense system.

Below we summarize the rationale behind each considered filtering problem and their relation to each other.

- P_0 *FILTER-ALL-STATIC*: Given a blacklist and a number of filters F, filter out *all* bad addresses, so as to minimize the collateral damage.
  Rationale: Filter out *all* bad addresses in a blacklist is the natural first step. The blacklist is constructed by a pre-processing step, that has identified and confirmed a consistent malicious behavior of the addresses that must be filtered out. This problem is interesting only if $F < N$, otherwise we could filter out each individual address with a single filter. We develop a greedy optimal algorithm that solves this problem.
- P_1 *FILTER-SOME-STATIC*: Given a blacklist and a number of filters F, filter out *some* bad addresses, so as to optimize the achievable tradeoff between collateral damage (false positives) and unfiltered bad addresses (false negatives).
  Rationale: The requirement of P_0 to filter out all the source IPs is too strict and may lead to large collateral damage if bad addresses are too spread apart in the address space. P_1 differs from P_0 in that it tolerates leaving some bad sources unfiltered in exchange for a reduction in collateral damage. Instead, it tries to find and block only those subsets of bad addresses that have the highest negative impact on the network performance. We develop a greedy optimal algorithm thats solves this problem. In the formulation, we provide a knob (namely, the weight w_i assigned to an address i) that allows the administrator to express how much she values each address and thus control the tradeoff achieved by the optimal algorithm.
- P_2 *FILTER-ALL-DYNAMIC*: Given a set of blacklists $BL = \{BL_{t_0}, BL_{t_1}, \cdots\}$, and number of filters, F, find a set of filter rules $\{S_{t_0}, S_{t_1}, \cdots\}$, such that S_{t_i} solves problem P_0 when the input list is BL_{t_i}.
  Rationale: In practice, ISPs and organizations continuously collect data and update their blacklists over time. P_2 is a filtering problem that updates the filtering rules, according to a time-varying blacklist. Similarly to P_0, the goal is to filter out all bad addresses at all times, at minimum collateral damage. The trivial solution to this problem is to iteratively solve P_0 for every instance of the blacklist. However, the temporal correlation in successive blacklists [8] can be exploited to design efficient greedy algorithms, similar to $P0$.
- P_3 *FILTER-ALL-DYNAMIC*: Given a set of blacklists $BL = \{BL_{t_0}, BL_{t_1}, \cdots\}$, and number of filters, F, find a set of filter rules $\{S_{t_0}, S_{t_1}, \cdots\}$, such that S_{t_i} solves problem P_1 when the input list is BL_{t_i}.
  Rationale: P_3 is the version of P_1 that deals with a time-varying blacklist: it adapts the filtering rules, according to a time-varying blacklist so as to filter some, but not all, bad addresses.

For each of the above problems, we develop a greedy optimal algorithm.

III. Filtering Problems and Algorithms

In this section, we give the detailed formulation of each problem and the optimal algorithm that solves it.

A. FILTER-ALL-STATIC

Problem P_0: Given a blacklist, $BL = \{A_{b_1}, A_{b_2}, ...A_{b_N}\}$, and a number of filters $F_{max} < N$, find a set of filters $[A_l, A_r]$, $l, r = 1, \ldots, N$ s.t. each bad address is covered by some filter $[A_l, A_r]$ and the total collateral damage from all filters is minimized, i.e.,

$$\min \sum_{l \le r} \tilde{C}_{l,r}$$

1) Optimal Filtering - Dynamic Programming (DP) Formulation: Let $OPT(f, n)$ be the optimal solution to the problem, i.e., blocking n bad addresses $\{A_{b_1}, A_{b_2}, ...A_{b_n}\}$, using up to f filters, with minimum collateral damage. We can compute $OPT(f, n)$ from the optimal solution that blocks the subset of addresses $\{A_{b_1}, A_{b_2}, ...A_{b_{n-1}}\}$: either we extend the f^{th} filter from $A_{b_{n-1}}$ to also cover A_{b_n}, thus filtering out the addresses between $A_{b_{n-1}}$ and A_{b_n} and adding cost $C_{A_{b_{n-1}}, A_{b_n}}$; or we use the optimal assignment of $f-1$ filters to addresses $\{A_{b_1}, ...A_{b_{n-1}}\}$ and assign one filter to address A_{b_n}[2]. Below is the corresponding dynamic programming equation:

Algorithm 1 FILTER-ALL-STATIC, DP Formulation

```
for f = 1, ...F do
    for n = 1, ..N do
        OPT(f, n) = min{OPT(f, n − 1) + C_{A_{b_{n−1}} A_{b_n}}, OPT(f − 1, n − 1)}
    end for
end for
```

Both the running time complexity and the memory space required by DP are $O(NF_{max})$, corresponding to the steps required to fill up the table and compute $OPT(F_{max}, N)$.

[2]These are the only two possible cases, as it can be proved by contradiction, using the same argument as in the proof of the Greedy algorithm. Due to lack of space, we don't repeat it here.

2) Optimal Filtering - Linear Programming (LP) Formulation: Below we also give an alternative formulation as a binary optimization problem. The reason is that this formulation can be extended to a similar formulation of P_1. Let $R_{l,r} \in \{0,1\}$ be decision variable equal to 1 iff the filter $[A_l, A_r]$ is used; 0 otherwise. Then the following program solves P_0:

$$\min \sum_{l \le r} \tilde{C}_{l,r} R_{l,r} \tag{3}$$

$$\sum_{l \le r} R_{l,r} \le F_{max} \tag{4}$$

$$\sum_{l \le i \le r} R_{l,r} \ge 1 \ \forall i \in \{b_1, b_2, \ldots, b_N\} \tag{5}$$

$$R_{l,r} \in \{0,1\} \ \forall l, r \in \{1, 2, \ldots, m\} \tag{6}$$

Eq.(4) imposes that at most F_{max} filters should be used. Eq.(5) states that for every bad address $i \in \mathcal{B}$ there should be at least one filter, $R_{l,r}$, for some l and r, blocking it. Eq.(3) represents our objective function: the sum of collateral damages caused by all the deployed filters. Note that, the summation in Eq.(3) is over indices $l \le r$ for symmetry. Furthermore, since we are looking for the minimum and $\tilde{C}_{l,r} \ge 0$, the optimal solution obtained by solving the LP program will not contain intervals that overlap on good addresses.

3) Optimal Filtering - Greedy Algorithm: The idea is to compute the "distances" (i.e., collateral damages) between consecutive bad addresses, and sort the resulting vector. Alg.2 picks the two closest addresses and covers them with a single filter. Then, it iteratively repeats this step until all bad addresses are filtered. An example is shown in Fig.1.

Algorithm 2 FILTER-ALL-STATIC, Greedy Algorithm

1: Compute $C_{b_i, b_{i+1}}, i = 1, ..., N-1$
2: Sort the resulting vector C in increasing order.
3: Pick up the first $N - F$ components.
4: Add/extend a filter for each selected interval.

The rationale behind this algorithm is that we want to cover all addresses while filtering out the smallest possible number of addresses. The ordering of all addresses on a single dimension, allows for this efficient solution.

Theorem 3.1: The greedy algorithm (Alg.2) computes an optimal solution to $P0$ Problem.

Proof: Assume that we have enough filters to block all bad addresses: $F = N$ and we assign a filter to each bad address. This is clearly an optimal solution with zero collateral damage. However, it is also an infeasible solution because, in reality, we have $F_{max} < N$ filters. The algorithm then proceeds by decreasing the number of filters by one in each step, while maintaining the optimality of the solution, until it reaches a feasible solution $F = F_{max}$.

In particular, assume that we want to decrease the number of filters from $F = N$ to $F = N - 1$. We insert the filter that causes the smallest amount of collateral damage; this is a filter between the two addresses with the smallest distance, e.g. R_{34} in Fig.1. It is easy to see that the obtained solution is still optimal in the case of $N - 1$ filters. Now, imagine that we "shrink" the newly added filter in a single point; e.g. in Fig.1 A_{b_3} and A_{b_4} are merged into $A_{b_{34}}$. Then there is no conceptual difference between the first solution (with N filters) and the new solution (with $N-1$ filters). Thus, we can repeat the same procedure: to reduce again the number of filters by 1, we simply pick the 2nd smaller element in the sorted array $\tilde{C}$ (e.g. filter $R_{N-1,N}$ in Fig.1), and so on. By induction, the above argument proves that Alg.2 is optimal. ■

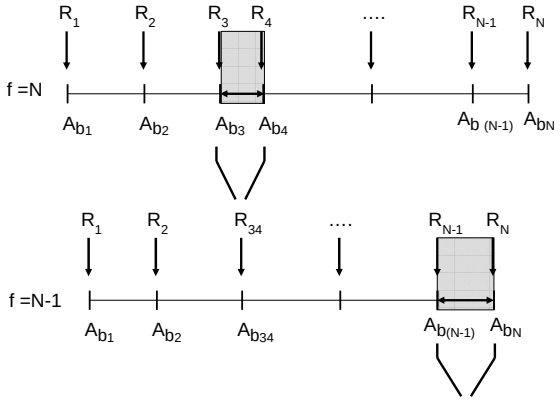


Fig. 1. Greedy algorithm for FILTER-ALL-STATIC. For $f = N$, one filter is assigned to each attack source. For $f = N - 1$, the two closest addresses (here A_{b_3} and A_{b_4}) are merged into a single range (R_{34}) and assigned a single filter. The same process continues iteratively for fewer filters.

Complexity. In the above construction, at each step, the number of filters will decrease *exactly* by one. Therefore, if we start with N filters, we need to perform exactly $N - F_{max}$ steps. The computational cost is due to sorting the vector of distances between addresses, C, which is $O(N \log(N))$. In practice, the values in C are not all distinct. Thus, sorting the distances can be done in even faster ways leveraging the redundancy of equal/similar values.

Since the greedy algorithm has low complexity and computes the optimal, this is the algorithm to use for problem P_0, for all practical purposes.

B. FILTER-SOME-STATIC

Problem P_1: Given a blacklist of N bad IP addresses, a number of filters $F_{max} < N$, and a weight w_i associated with every source address, assign filters so as to optimize the tradeoff between: (i) the total collateral damage and (ii) the total benefit associated with blocking bad addresses.

In section II-A, we introduced the weight w_i assigned to an address A_i as a measure of its "badness"/"goodness". W.l.o.g., let us assign a weight to every address according to the following convention: $w_i \ge 0$ for every address not in the blacklist ("good" address), and $w_i \le 0$ otherwise ("bad" address). In the latter case, we can interpret $w_i < 0$ as the *benefit* associated with blocking a single bad address A_i. Examples of w_i in the

case of bad addresses include the number of reports from this IP, the amount of traffic sent, etc. $w_i = 0$ indicates that we do not account for the specific address, whether it is good or bad.

Problem P_1 can be formulated as follows:

$$\min \sum_{l \le r} \sum_{l \le i \le r} w_i R_{l,r} \tag{7}$$

$$\sum_{l \le r} R_{l,r} \le F_{max} \tag{8}$$

$$\sum_{(i,j)\,:\, i \le r \text{ and } j \ge l} R_{i,j} \le 1 \quad \forall l, r \in \{1, 2, \ldots, N\} \tag{9}$$

$$R_{l,r} \in \{0, 1\} \quad \forall l, r \in \{1, 2, \ldots, N\} \tag{10}$$

This is similar to the LP formulation of P_0, but with some major differences. Eq.(5) has been removed, since we no longer require all IPs to be blocked. In addition, since the coefficient associated with every range, $\sum_{l \le i \le r} w_i$ can be either positive or negative, the argument used in P_0 to prove the non-existence of overlapping ranges in the optimal solution, does not hold anymore. For this reason, we need to explicitly introduce Eq.(9) which prevents overlapping ranges. Without Eq.(9) the above is exactly a knapsack 0/1 problem, with items the filters/ranges. This is, in general, a well-known NP-hard problem. In addition, Eq.(9) introduces correlation between the knapsack items, since the choice of a bigger interval excludes, for instance, to jointly select any of the intervals included in it.

However, by leveraging the peculiarities of the problem (such as the fact that all address can be represented as points in a 1-dimensional space), we can design a greedy, thus computationally efficient, algorithm to solve P_1. The new algorithm is described in Alg.3 and it is designed along the lines of Alg.2.

Assume that we have as many filters as the number of bad addresses, $F = N$. The optimal solution (of negative value) will use a filter for every single bad address. If we want to use $N - 1$ filters, the optimal solution will connect two bad addresses, if and only if there is gain in the objective function by extending an existing filter; otherwise, a bad address (the one with the highest value) will be left unblocked. We can state this condition by observing that there is gain in connecting two consecutive bad addresses, if and only if:

$$\exists\, i \in \{1, ..., N\} \text{ s.t. } \sum_{b_i \le j \le b_{i+1}} w_j \mathbb{I}_{\mathcal{G}}(j) \le |\max\{w_{b_i}, w_{b_{i+1}}\}|$$

If such an index i does not exist, P_1 will simply block $(N-1)$ IP addresses, and leave the address with the higher weight unblocked.

More formally, let us denote with $Ranges$ the set of filter rules constructed, and with D the set of distances (collateral damages) between *consecutive* filters. Starting from an optimal solution with $F_{max} = N$, Alg.3 iteratively decreases the number of used filters, while preserving the optimality of the solution, until a feasible solution is reached (i.e., $f \le F_{max}$).

Algorithm 3 FILTER-SOME-STATIC, Greedy Algorithm

```
1:  Ranges = {R_{j,j} : j ∈ B}
2:  D = {C_{j,j+1} : j ∈ B}
3:  F = N
4:  Z = Σ_{j∈B} w_j
5:  while f > F_max do
6:      w = max Ranges
7:      I = min D
8:      if |w| < I then
9:          Let (ī, j̄) be such that: w = R_{ī,j̄}
10:         Ranges = Ranges \ {R_{ī,j̄}}
11:         if C_{l,ī}, C_{j̄,r} ∈ D then
12:             D = D \ {C_{l,ī}, C_{j̄,r}}, where l, r ∈ B
13:             D = D ∪ {C_{l,r}},  C_{l,r} = C_{l,ī} + C_{j̄,r} + w
14:         end if
15:         Z = Z − w
16:         f = f − 1
17:     else
18:         Let (ī, j̄) be such that: I = C_{ī,j̄}
19:         D = D \ {C_{ī,j̄}}
20:         if R_{l,ī}, R_{j̄,r} ∈ Ranges then
21:             Ranges = Ranges \ {R_{l,ī}, R_{j̄,r}}, where l, r ∈ B
22:             Ranges = Ranges ∪ {R_{l,r}},  R_{l,r} = R_{l,ī} + R_{j̄,r} + I
23:             Z = Z + I
24:             F = F − 1
25:         end if
26:     end if
27: end while
```

Lines 6-7 check for the maximum and the minimum of the lists $Ranges$, and D, respectively. At every step of the while loop (Line 5), a greedy decision is taken: a single filter is removed, or two different filters are merged together (whenever they exist), depending on which results in the lowest increase in the objective function. Specifically: if $|w| < I$, the filter $R_{\bar{i},\bar{j}}$ corresponding to the value w, i.e., $R_{\bar{i},\bar{j}} = w$, is removed from the list of used filters (Line 10), and the list of distances between filters, D, is correspondingly updated: if two intervals, one on the left and one on the right of $R_{\bar{i},\bar{j}}$, are found, they are merged together, resulting in the larger interval $C_{l,r}$ (Lines 12-13). Otherwise, the smallest distance, I, is selected, and the two adjacent filters (if they exist) are merged in a single filter which encompasses I (Lines 19-22). Unless the selected interval, I, reaches one the extrema of the address space, and thus it is not actually possible to merge two filters, the above algorithm, at every step, reduces the number of used filters, f, by exactly one.

Finally, we observe that, the objective function, defined in Eq.(7), can be split into two contributions

$$\min \sum_{l \le r} \left(\sum_{l \le i \le r,} w_i \mathbb{I}_{\mathcal{G}}(i) + \sum_{l \le i \le r} w_i \mathbb{I}_{\mathcal{B}}(i) \right) R_{l,r} \tag{11}$$

thus allowing for an intuitive cost/benefit analysis. For every address range, the first term represents the cost of introducing that filter, which in our case is the collateral damage, i.e., the (weighted) sum over all good addresses that are accidentally blocked. The second term represents the benefit from imposing filter $R_{l,r}$; this term depends on the number of bad addresses blocked by the filter and on the weight assigned to each of them. The tuning of

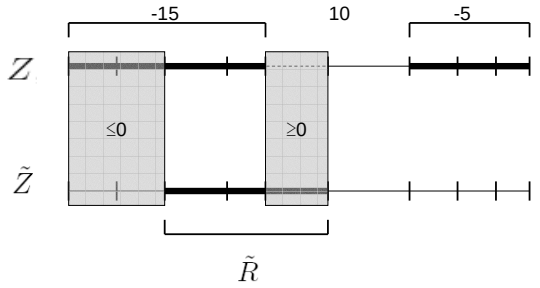


Fig. 2. [Part of the proof of optimality of P1.] Z is the optimal solution using f filters. Let $\tilde{Z}$ be an optimal solution for $f-1$ filters, different than the solution constructed inductively from Z (i.e., merging two filters or releasing a single filter). This leads to a contradiction.

w_i allows the network provider to control the outcome of the algorithm, as it will be discussed later.

Theorem 3.2: Alg.3 computes an optimal solution to P_1.

Proof: In $O(N-F)$ iterations of the while cycle, the number of filters is reduced from N to $F \leq F_{max}$, thus obtaining a feasible solution for P1.

We now prove, by reverse induction, that for every $F \geq 0$, the above algorithm produces an optimal solution to P_1. When $F = N$, $Z = \sum_{j \in \mathcal{B}} w_j$ represents an optimal solution for P_1, which blocks all the bad IPs, without collateral damage; clearly, there is no better solution.

To prove the inductive step, we first observe that given two optimal solutions of P_1, S_n^{OPT}, and, S_m^{OPT}, with n and m, filters respectively, if $m < n$, then we also have $S_n^{OPT} \leq S_m^{OPT}$. By contradiction, assume $S_n^{OPT} < S_m^{OPT}$; having at most $n > m$ filters we can use m of them to reproduce the same solution obtained with m filters, S_m^{OPT}. Moreover, since $n < N$, the remaining $n-m$ filters can be used to cover $n-m$ bad IPs, each of them with weight, $w_j \leq 0, \forall j \in \mathcal{B}$. Thus, we have constructed a solution with n filters S_n such that: $S_n \leq S_m^{OPT}$. Recalling that, $S_n^{OPT} \leq S_n$, we conclude that $S_n^{OPT} \leq S_m^{OPT}$. This implies that when searching for the optimal solution with $F \leq F_{max}$ filters, we can assume that we are using exactly $F = F_{max}$ filters.

Now, assume that the above algorithm produces an optimal solution, Z, with $F = n$, where $n = 1, 2, ..., N$. This implies that an optimal solution, Z^*, can be constructed at the next step $F = n-1$. In order to reduce the number of filters from n to $n-1$ we are given exactly two options: either (i) release a filter or (ii) merge two neighboring filters. Among these two options, Alg.3 chooses the one that leads to the smallest increment in the objective function, thus guaranteeing the optimality of the solution.

A different solution, that is not constructed using any of the two aforementioned options (i.e. without neither merging two different filters, nor completely releasing an existing filter) does not decrement F. We can prove this by contradiction. Assume that there exists an optimal solution with $n-1$ filters, that is constructed with a completely new reallocation of filters with respect to Z, e.g. by partially extending/releasiing a filter as shown in Fig2. Let us denote this solution with $\tilde{Z}$. We show that $\tilde{Z}$ cannot be an improved solution, i.e., $\tilde{Z} \geq Z$. By the definition of $\tilde{Z}$, there is at least one filter, $R_{\tilde{Z}}$, which does not coincide neither with a single range in Z, nor with the possible result of a merging operation of two ranges in Z, Fig.2. Then, if $R_{\tilde{Z}}$ covers a portion of the address space that is not covered in Z, it is possible to construct a new solution $\tilde{Z}'$ by shortening $R_{\tilde{Z}}$. In fact all ranges between bad IPs which are not covered in Z, and that do not cause the merge of two different ranges, give a non-negative contribution to Z. This is easily seen by contradiction: if it was not true (i.e., they give a negative contribution to Z), then we could add it to Z and thus obtain a new solution Z' which uses the same number of filters as Z, but such that $Z' < Z$, which violate our assumption on the optimality of Z. With the same argument we can prove that if $R_{\tilde{Z}}$ covers only a portion of the address space covered by in Z, then we can extend $R_{\tilde{Z}}$ and thus obtain a new solution $\tilde{Z}'$, such that, $\tilde{Z}' \leq \tilde{Z}$. Iterating these steps, we obtain a new solution $\tilde{Z}'$ which is or strictly lower than $\tilde{Z}$, or is exactly the same solution we can obtain from Z releasing a filter, or merging two of them. ■

1) Complexity: Given a pre-processing step in which $Ranges$, and D are sorted, Lines 6-7, can be performed in $O(1)$ time. Under this assumption, Lines 12-22 perform insertion and removal operations from sorted lists and, each of them can be performed in $O(\log N)$. With a simple modification to Alg.3 it can be easily proved that $N - F_{max}$ iterations are required to converge to the optimal solution. Thus the running time complexity of Lines 5-27 is $O((N - F_{max}) \log N)$, and the overall complexity of Alg.3 results bounded by the maximum between this cost and the sorting cost of the input vectors, (e.g. $O(N \log N)$ using standard techniques).

2) How to select the w_i's?: The framework we introduced provides the network provider with the flexibility to control the output of the optimal solution, and therefore to allocate filtering resources according to her needs and preferences. The main knob to configure is the weight w_i assigned to each address A_i that can be used to express preference per individual address. For example, the weights w_i can be tuned to assigned different access "privileges" to different users (e.g. higher positive weights to "trusted" or "preferred" customers); and conversely they can also be used to make sure to block IPs that are responsible for the highest number of malicious activities (by assigning smaller negative weights to those IPs).

Below are some options and guidelines for possible assignments of w_i based on Eq.(11):

- In general, $w_i \geq 0$ should be assigned to good addresses, $w_i \leq 0$ should be assigned to bad addresses. For extreme flexibility/granularity, one could assign w_i to each individual address or blocks of addresses. In fact, what matters is the ratio of weights between good and bad addresses and not the absolute values. E.g. one could assign $w_i = 1$ to all good addresses and $w_i = W < 0$ to all bad addresses to express their relative importance.

- One simple case is: $w_i = 1$ for all good addresses and $w_i = -\infty$ for all bad addresses. The objective is then to minimize the collateral damage $\sum w_i$, while filtering all bad address, and problem P_1 degenerates to P_0.
- One could assign $w_i = 0$ to show indifference to the treatment of those addresses.
- Other possible assignments of the weights can be driven by: historical data/reputation systems (e.g. assign lower negative weight to IPs with worst reputation, or that showed up repetitively in the past instances of the blacklist); by a policy of the network operator (e.g. protect important customers from being accidentally blocked: the higher positive weight to those addresses the higher the probability that they will not be blocked); by the volume of good/bad traffic exchanged and/or by the severity of attacks launched.

3) Comparison between P_1 and a P_0-based heuristic: Given a blacklist, P_0 aims at filtering out *all* bad addresses in it at the minimum possible collateral damage. In contrast, P_1 blocks only some addresses trying to find a tradeoff between the bad traffic left unfiltered and the collateral damage inevitably when $F < N$. An alternative to P_1 for trading-off good vs. bad addresses could be to first select a specific subset of IPs, $S \subseteq BL$, (e.g. the subset of heavy hitters) and then run P_0 on this specific subset. To allow a fair comparison in terms of the solutions provided by this method, and the solutions of P1, in the following of this section, we assume that the objective function of P_0 includes a constant term given by the sum of weights assigned to all bad addresses included S [3]

Given any subset of the blacklist, $S \subset BL$, the solution of P_1 given the whole blacklist, $Z_{P1}(BL)$, is less or equal to the solution of P_0 given S as an input list, $Z_{P0}(S)$, $\forall S$. We now prove that it exists a subset of BL such that also the reverse relation holds true, that is, $\exists S^* \subset BL \, s.t. \, Z_{P0}(S^*) \leq Z_{P1}(BL)$. Obviously this relation does not hold $\forall S \subset BL$; however, we can construct S^* by first running P_1 on BL. Let us denote the set of filters constructed by P_1 as $\bar{S}$. By Theorem 3.1, running P_0 on $\bar{S}$ gives a solution that minimizes collateral damages while filtering out all addresses in $\bar{S}$. Given this construction, the two solutions, $Z_{P0}(\bar{S})$ and $Z_{P1}(BL)$ are guaranteed to have the same amount of bad traffic filtered out; moreover, since $Z_{P0}(\bar{S})$ has the smallest amount of collateral damages, we have: $Z_{P0}(\bar{S}) \leq Z_{P1}(BL)$.

In conclusion, the technique discussed here represents a heuristic method for P_1. Only when an optimal subset S^* is somehow selected, this approach is as effective as Alg.3. However, since the complexity of this heuristic is lower-bounded by the complexity of Alg.2, which is asymptotically equal to the complexity of Alg.3, we conclude that, Alg.3 is a powerful method even compared to this heuristic.

[3]In fact, all these addresses will be filtered out in any feasible solutions of P0.

C. FILTER-ALL-DYNAMIC

Let us consider a time-varying blacklist, meaning that a new instance of the blacklist is available in each time slot. By appropriately choosing the timeslot, we can assume w.l.o.g. that in each time slot, either one new bad address arrives, or an old bad address departs. The goal is similar to P_0: to filter out all bad addresses at minimum collateral damage in every time slot.

In the first time slot, we run the greedy algorithm FILTER-ALL-STATIC and we create a sorted list of collateral damage for filters of consecutive bad addresses. In subsequent time slots, we update the sorted list and our filtering choice by exploiting the greedy property. This is very efficient: it is basically sufficient to add/remove values from the sorted list, C, of collateral damages caused by filters of the type: $[A_{b_i}, A_{b_{i+1}}]$ (consecutive bad addresses). The new address A_i is located between bad address A_{l_i} and A_{r_i}, such that $A_{l_i} < A_i < A_{r_i}$. A symmetric argument holds for an old bad address, A_i, that is removed from the blacklist, $N \leftarrow N-1$. The only difference in this case, is that we basically remove from C the two old values $C_{l_i,i}$ and C_{i,r_i}, and add a single new value C_{l_i,r_i} in the correct position. The notation $C[k]$ indicates the k-th component of the vector C before the updating step was performed (i.e., before A_i was added/removed). OPT indicates the value of the optimal solution before the updating.

Algorithm 4 basically keeps the list sorted, and evaluates the value of the new optimal solution. The optimality of the new solution is based on the properties of P_0, described in previous section.

Algorithm 4 FILTER-ALL-DYNAMIC, Greedy Algorithm

A new bad address, A_i, arrives: $N \leftarrow N+1$
Let I_{l_i,r_i} be the position of C_{l_i,r_i} in the vector C, i.e., $C_{l_i,r_i} = C[I_{l_i,r_i}]$
Remove C_{l_i,r_i} from the sorted vector C.
if $I_{l_i,r_i} \leq N - F$ **then**
    OPT = OPT - C_{l_i,r_i}
end if
Insert the two new values $C_{l_i,i}$ and C_{i,r_i} in the correct position. Let, $I_{l_i,i}$, and I_{i,r_i} be their positions in C respectively.
if $\max\{I_{l_i,i}, I_{r_i,i}\} \leq N - F$ **then**
    if $I_{l_i,r_i} < N - F$ **then**
        OPT = OPT + $C_{l_i,i} + C_{r_i,i} - C[N-F]$
    else
        if $I_{l_i,r_i} = N - F$ **then**
            OPT = OPT + $C_{l_i,i} + C_{r_i,i} - C[N-F-1]$
        else
            OPT = OPT + $C_{l_i,i} + C_{r_i,i} - C[N-F] - C[N-F-1]$
        end if
    end if
else
    if $\min\{I_{l_i,i}, I_{r_i,i}\} \leq N - F$ **then**
        if $I_{l_i,r_i} \leq N - F$ **then**
            OPT = OPT +$\min\{C_{l_i,i}, C_{r_i,i}\}$
        else
            OPT = OPT +$\min\{C_{l_i,i}, C_{r_i,i}\} - C[N-F]$
        end if
    end if
end if

Handling batch arrivals/departures. If in one timeslot, several bad addresses arrive and/or depart together, then

we can apply the above algorithm for each individual arrival and departure. As long as the number of entering and departing addresses remains smaller than N, the computational cost of updating the list remains smaller than the cost of re-running P_0 from scratch for every instance. However, it is possible, that a more efficient variation may exist for batch arrivals/departures.

Clearly, the efficiency of Alg.4 depends on the characteristics of the time-varying blacklist, i.e., on how much blacklists change from one time slot to the next. One extreme case is that the blacklist never changes: then we run the greedy algorithm for P_0 once. The other extreme is that the blacklist is always completely different from one time slot to the next; then we run the greedy algorithm for P_0 for every instance. In practice, it is has been observed that there is temporal correlation in blacklists [8], which the Alg.4 should be able to exploit to make greedy decisions.

D. FILTER-SOME-DYNAMIC

Similarly, we can design a time-varying version for P_1 to leverage the temporal correlation between successive time instances of the blacklist and thus be computationally more efficient than running Alg.3 form scratch on every instance of the blacklist. Below we only outline the main ideas.

Assume, that a new address is added to the blacklist. The key idea is that, given an optimal solution of P_1, any filter cannot be further extended without causing the objective function to increase or remain at the same value. When a new address, A_i, is added to the blacklist we can assume to temporarily place a filter on it, with associated coefficient equal to w_i. Then, we can extend it until there exist a possible extension (to the left, or to the right of the filter) which causes the coefficient associated with that filter (i.e., the sum of weights of good and bad addresses covered by the filter) to decrease, and thus possibly improve the objective function. When no further extensions are possible, we are in one of two possible situations: we have merged the newly created filter with an existing one, and in this case the algorithm stops; or we have created a new filter disjoint from all other filters. In the latter case we need to remove a filter. This is done by removing the filter with the smallest contribution to the objective function (it can be either the filter just created with this process or an existing one).

Similarly, when an address is removed from the blacklist we have two possible cases, depending on whether the address was filtered in the original solution or not. In the latter case, no modification is required. However, if the address removed from the blacklist was included is some filter, say R_{l_i,r_i}, we must ensure that the coefficient associated with R_{l_i,r_i} is still the smallest among all unused filters in $Ranges$. Otherwise, the one which gives the smallest contribution must be added instead of R_{l_i,r_i}.

IV. Performance Analysis

In Section III, we designed greedy algorithms and proved their optimality. However, the performance of each algorithm strongly depends on the number of available filters and also on the inherent characteristics of the input blacklist, namely the *density* and the degree of *clustering*. Clearly, the higher both of these metrics are, the better the tradeoff between collateral damage (false positives) and unfiltered bad addresses (false positives). E.g. a blacklist with N addresses spread at equal distances $2^{32}/N$ apart is the worst case input; consecutive addresses is the best case, as they can be blocked with a single filter. The interesting question is then how do actual blacklists look like and how do our algorithms perform in practice.

There is a number of measurement papers that have studied the structure of addresses in IP traffic, both wanted [10] and unwanted [7]. With regards to the sources of unwanted traffic, which are of interest here, they all observed clustering in the sense that a few prefixes contain most of sources [7]–[9]. In particular Barford et al. [7] showed that sources of unwanted traffic exhibit multi-fractal behavior and provided a probabilistic model for generation of such addresses. This work built on and extended the work by Kohler et al. [10], which modeled the spatial distribution of destination IP addresses in legitimate traffic, showed that they exhibit multifractal behavior and provided a deterministic way to generate such a distribution using Cantor sets.

In our evaluation, we use the multi-fractal model from [10], which is an extreme case of the model in [7], to generate blacklists with realistic spatial distribution of sources in the IP address space. In particular, the following parameters of the model in [10] are of interest:

- $0 \leq h \leq 1$ controls the fractal exponent and is set to $h = 1/3$, as in canonical Cantor dust.
- $0 \leq m_0 \leq 1$ controls the spatial variability/clustering, which is the crucial parameter in the filtering problem. We considered a range of $m_0 \in \{0.65, 0.7, ...0.95\}$, where higher m_0 means higher clustering.

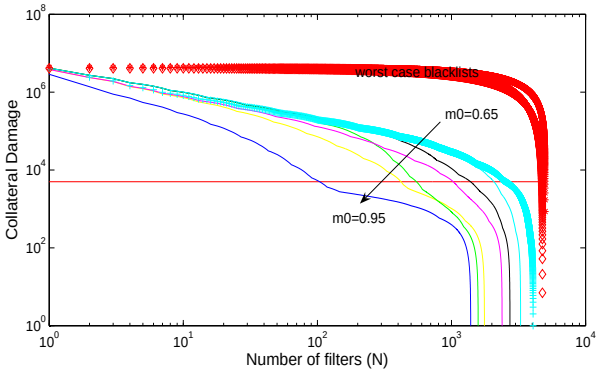
For computational reasons we considered 1/1000 of the entire address space ($2^{32}/1000$ addresses). We varied the degree of clustering via the parameter m_0. Instead of using h, we varied the density d, i.e., the % of the total address space considered being bad, by varying the number of bad addresses N. We considered a low density ($N = 5000$ bad addresses leading to density $d = 0.001$) and a high density ($d = 0.1$) scenario.

We generated several blacklists with different characteristics and simulated the algorithms for the static case, namely FILTER-ALL-STATIC, FILTER-SOME-STATIC. We are interested in several performance metrics: (i) the collateral damage or "CD" (or false positives) and (ii) the number of unfiltered IPs (false negatives) (iii) number of filters F used. Below are the results.

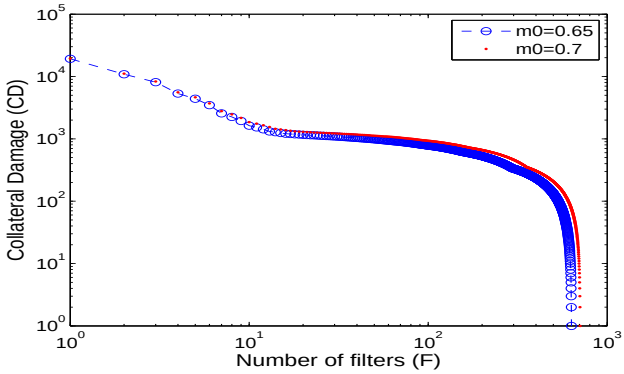
A. Performance of FILTER-ALL-STATIC

In Fig.3, we consider input blacklists with varying degrees of clustering, for both low and high density of bad addresses.

In Fig.3(a), we show the performance of FILTER-ALL-STATIC for a blacklist with low density $d = 0.001$.

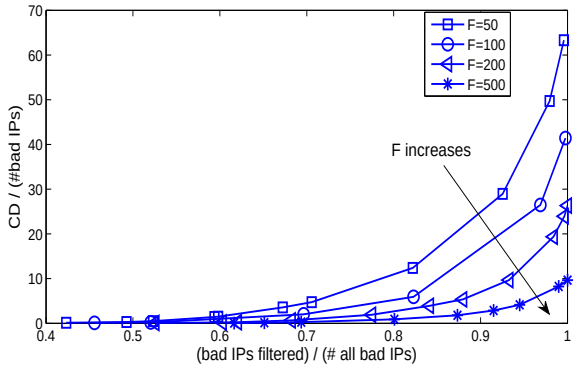


(a) Low density blacklist ($d = 0.001$) and $m = 0.65, 0.70, ...0.95$

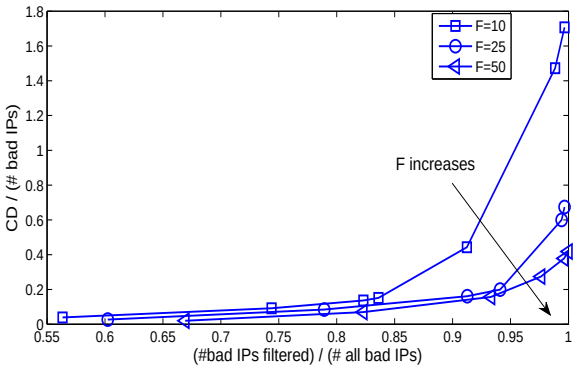


(b) High density blacklist ($d = 0.1$) and medium $m_0 = 0.65, 0.7$

Fig. 3. Performance of FILTER-ALL-STATIC (P_0) for blacklists with various densities and degrees of clustering



(a) Low density blacklist



(b) High density blacklist

Fig. 4. Performance of FILTER-SOME-STATIC (P1) for a blacklists with medium degree of clustering ($m_0 = 0.75$). Varying the weight W controls the tradeoff between (collateral damage) and (number of filtered bad addresses). Both these numbers are normalized with respect to the total number of bad addresses (N).

As expected for higher degree of clustering (i.e., higher m_0), the algorithm performs better. Moreover, we observe that higher gains are obtained when a small amount of collateral damage is allowed ($\frac{CD}{N} \simeq 1$). When the collateral damage is high, all solutions use a very low number of filters ($F \in [1, 10]$), and there is no substantial difference in terms of collateral damage. When collateral damage is low ($\frac{CD}{N} \ll 1$), m_0 does not have a significant effect either. This is due to the way bad IPs are distributed in the address space and also due to the requirement to cover *all* bad IPs. Indeed, in the case of a low density, regardless of the degree of clustering degree, there will always be some isolated IPs located "far away" from the main bulk of bad IPs; covering those addresses causes a large portion of the good address space to be filtered.

The horizontal line shows the point where the ratio between collateral damage and filtered IPs is $\frac{CD}{N} = 1$. At the top right of Fig.3(a), we also show the worst case input blacklists: equally spaced source IPs is the worst case deterministic input, and uniformly distributed IPs is the worst probabilistic input. When the number of filters F takes extremes values ($F = 1$, $F = N$), the 2 curves basically coincide. In all other cases, as expected, the uniform distribution outperforms the deterministic case: the algorithm finds some bad addresses that are closer than other and merges them together thus limiting the collateral damage compared to the deterministic case.

In Fig.3(b), we consider a high density blacklist ($d = 0.1$) for various degrees of clustering. Comparing this figure to the previous, we observe the following. On one hand, the number of filters required to block all bad IPs without collateral damage is still quite high. On the other hand, contrary to the low density case, as the number of available filters decreases the collateral damage remains limited. In fact, it is always possible to construct a solution that covers all bad IPs keeping the ratio low $\frac{CD}{N} \simeq 1$.

B. Performance of FILTER-SOME-STATIC

Next, we analyze the performance of FILTER-SOME-STATIC as the number of filters varies. We assign the same weight, $W < 0$ to bad addresses and the same weight $W = 1$ to all good addresses; we then vary W. In this simple case, W represents the relative weight of bad addresses with respect to good addresses. Thus, by varying W we tune the tradeoff between the bad IPs filtered and the collateral damage: the higher W, the higher the number of bad IPs filtered at the expense of higher collateral damage; and vice versa.

Similarly to what we did before, we analyze the performance of FILTER-SOME-STATIC both in the case of low ($d = 0.001$) and high density ($d = 0.001$). In the case of low density, Fig.4(a), we see that we can cover most of

the IPs at low cost; but in order to cover the last IPs, we pay higher collateral damage. More precisely, depending on the number of available filters, we can cover from 70% up to 100% of the bad IPs while keeping the ratio $\frac{CD}{N}$ within one order of magnitude. However, if the number of filters is small, there is an exponential increase in the collateral damage to cover the remaining 20% of bad IPs. In practical scenarios, where the number of filters is not enough to cover all bad IPs, this further motivates the investigation of solutions that provide a good balance between the number of bad and good IPs filtered. In some cases, we may tolerate to block, for instance, up to 80% of unwanted traffic and to incur CD which is linear with N; in other cases, we may want to provide a higher protection from possible attacks at the expense of an exponential increase of the collateral damages. With the framework introduced here, all these different solutions can be simply obtained, by tuning W. Therefore, W is the control available to the network operator that offers flexibility in the quality of services they may want to provide.

We would also like to point out that, while in this section we presented the results using only two different constant weights, one for good and one for bad addresses, within the same framework it is also possible to assign different weights to a single IP or a group of IPs, to allow for finer control. For instance, by assigning weights equal to $-\infty$, (or $+\infty$) to a specific group of IPs, we can force the algorithm to always filter (or not filter) those addresses.

In Fig.4(b), where higher density of bad IPs is considered, results achieved by FILTER-SOME-STATIC are quite different. The ratio $\frac{CD}{N}$ is significantly lower (≤ 2) than in the low density case (≤ 70). There is an increase in the amount of collateral damage only to cover the last $5-10\%$ of IPs; even covering this last part of the BL, the collateral damage remains quite low in all cases.

C. FILTER-ALL-DYNAMIC, FILTER-SOME-DYNAMIC

The model introduced in [10], [7] is appropriate for the spatial distribution of source IPs of unwanted traffic. However, in a practical deployment the temporal behavior, i.e., how bad addresses appear, disappear, and re-appear in the blacklist, will also affect the performance of a filtering system. In same cases, a new IP appearing in the BL may be located in a region of the address space which is already blocked; in these cases, there is no need to re-evaluate a new set of filter rules, since the old one already provides an optimal solution. In other cases, it may be useful to recompute a new solution. FILTER-ALL-DYNAMIC and FILTER-SOME-DYNAMIC aim to tackle these later cases by rapidly providing an adaptation of the existing filtering rules to the new input BL at a much lower computational cost than running from scratch FILTER-ALL-STATIC and FILTER-SIME-STATIC for each instance from scratch.

Recent experimental evidence [8] and our own experience indicate that there is often temporal correlation in blacklists. One intuitive explanation may be that poorly administered networks are more likely to be infected for a period of time thus leading to temporal predictiveness of BLs and potentially of our filtering rules. As part of future work, we plan to evaluate the effectiveness of the dynamic algorithms using real time-varying blacklists [4], [5].

V. Summary and Future Work

In this paper, we designed optimal greedy algorithms that construct compact filtering rules to block IP address ranges given a blacklist. We are currently exploring several directions for future work. First, we are interested in extending our algorithms to be prefix-aware, in the sense that the ranges/filters will be aligned with prefixes/masks that can be directly specified by ACLs. Second, we are in the process of applying our algorithms to publicly available blacklists [4] and Dshield data [5]. Finally, we are interested in using the filtering algorithms as a building block of a bigger system that effectively protects a network from unwanted traffic.

References

[1] G. Pack, J. Yoon, E. Collins, C. Estan, "On Filtering of DDoS Attacks Based on Source Address Prefixes," *in Proc. of SecureComm*, Aug.2006.

[2] K. Argyraki and D.R.Cheriton, "Active Internet Traffic Filtering: Real-time Response to Denial-of-service Attacks", *in Proc. of USENIX Security 2005*.

[3] P. Verkaik, O. Spatscheck, J. van der Merwe and A. Snoeren, "PRIMED: A Community-of-Interest-Based DDoS Mitigation System," *in SIGCOMM Workshop on Large Scale Attack Defense (LSAD)*, Sept. 2006

[4] The Spamhaus Project, XBL, *http://http://www.spamhaus.org/xbl/*

[5] Dshield *http://www.dshield.org/*

[6] SRI International, "The highly predictive blacklist service," *http://www.cyber-ta.org/releases/HPB/*

[7] P.Barford, R.Nowak, R. Willett Rebecca, V. Yegneswaran, "Toward a Model for Sources of Internet Background Radiation," *in Proc. of the Passive and Active Measurement Conference (PAM '06)*, March, 2006.

[8] M. P. Collins, S. Faber, J. Janies, R. Weaver, M. De Shon, "Using Uncleanliness to Predict Future Botnet Addresses," *in Proc. of ACm IMC 2007*, San Diego, CA, Oct. 2007.

[9] Z. Chen and C. Ji, "Measuring Network-Aware Worm Spreading Ability," *in Proc. of IEEE INFOCOM 2007*.

[10] E. Kohler, J. Li, V. Paxson, and S. Shenker, "Observed structure of addresses in IP traffic," *in IEEE/ACM Transactions on Networking 14(6)*, Dec. 2006, pp. 1207-1218.

[11] K. El Defrawy, A. Markopoulou, and K. Argyraki, "Optimal Allocation of Filters against DDoS Attacks,", *in ITA Workshop*, San Diego, CA, Jan. 2007.