Source Address Filtering For Large Scale Network: A Cooperative Software Mechanism Design

Shu Yang[1], Mingwei Xu[1], Dan Wang[2] and Jianping Wu[3]
[1]Dept. of Comp Sci. & Tech., Tsinghua Univ., Tsinghua National Laboratory for Information Science and Technology
[2]Dept. of Computing, The Hong Kong Polytechnic Univ. [3]Network Research Center, Tsinghua Univ.

***Abstract*—Source address filtering is used as an important mechanism to prevent malicious traffic. Currently, most networks store filters in hardware such as TCAM, which has limited capacity, high power consumption and high cost. Although software can accommodate large number of filters, it needs multiple accesses to memory on the border router, which bears much more additional burden than other routers.**

In this paper, we propose a software-based mechanism for source address filtering. In our mechanism, we only need to check a few bits in source addresses on each router, rather than checking all bits on the ingress router. Through cooperation among routers, our mechanism ensures that malicious traffic will be filtered in the network.

We formulate this problem as finding a cooperative scheme such that the loads on all routers are optimally balanced. We show that the problem can be optimally solved by dynamic programming. We evaluate our algorithms using comprehensive simulations with BRITE generated topologies and real world topologies. We conduct a case study on China Education and Research Network 2 (CERNET2) configurations, a large IPv6 network. Compared to checking 128-bit IP addresses on ingress routers, our algorithm checks at most 40 bits on each router.

I. Introduction

Packet filtering is a prevalent mechanism for preventing malicious traffic, such as DDoS attack and scanning. Because of the important semantics of source address, source address filtering is widely used in networks. Usually, ingress router maintains a blacklist, i.e., the source addresses that should be filtered. With security problem becoming more serious, the blacklist has increased explosively in the past few years, especially for large scale networks. In 2003, it was reported that there were more than 20K sources during an attack against an online betting site [3]. In 2007, the size of the Storm botnet reached 50M [12]. In 2008, there were more than 800K unique malicious IP sources reported every day, according to data from Dshield.org [2]. The devastating security crisis forces ISPs to increases their blacklist size, to defend against attacks from possible malicious sources.

TCAM is currently used as the de facto industry standard to process IP prefixes. Though it has wire speed performance, TCAM has very limited capacity due to high cost. Cisco 12000, a high end core router, can only accommodate 20000 entries per line-card. With tens of thousands of filters per-path in the network, the limited TCAM resources are not enough to block currently witnessed attacks, not to mention larger attacks from millions of sources expected in the near future [18]. Such limitation make some TCAM-based solutions even block part of the legitimate traffic for better aggregation [16]. We believe that the number of blacklisted source address to be filtered will keep increasing in a much faster pace in the foreseeable future than the increase of TCAM capacity. Therefore, software based solutions are promising to accommodate the large space required by the filters. The key reason that software based solutions are not widely used in practice nowadays is that, although software-based SRAM also has fast speed[1], these schemes require multiple accesses to perform a single lookup; this introduces large latency and congestion.

Conventionally, the filters are placed at border routers, where the transit traffic definitely passes by. The processing burden on border routers is high. In this paper, we design a novel cooperative mechanism, where not only border routers, but also some downstream routers in the network can work cooperatively to handle the source-IP filtering. Such design scales well facing the increase of the filtering requirement.

Unlike previous schemes, that assign tasks to routers by filters, i.e., address blocks that should be filtered. Our scheme assigns tasks to routers by bits. That is, multiple routers will each check partial rather than all bits in source addresses. Thus, fast lookup speed can be achieved on each router. We guarantee correctness, i.e., filtering all malicious traffic in the network; in other words, the routers will cooperatively check all bits in source addresses. In our scheme, we can get the best of both worlds: larger storage space in SRAM/DRAM, and faster lookup speed.

Simple Example: We first use a simple example to explain our idea. In Fig. 1. Traditionally, the filters are placed at the border router, e.g., router a. Thus a requires 3 accesses to memory to filter malicious traffic in the worst case. In our new mechanism, a has only to check the 0_{th} bit, b checks the 1_{st}, and c checks the 2_{nd}. Assume that a packet with source address 010 arrives at router a, and the path towards

Email:{yangshu,xmw}@csnet1.cs.tsinghua.edu.cn,csdwang@comp.polyu.edu.hk,jianping@cernet.edu.cn

The research is supported by the National Basic Research Program of China (973 Program) under Grant 2009CB320502, the National Natural Science Foundation of China (61073166 and 61161140454),the National High-Tech Research and Development Program of China (863 Program) under Grants 2011AA01A101, the National Science & Technology Pillar Program of China under Grant 2011BAH19B01

[1]The maximum clock rate of SRAM is 400MHz, while TCAM is 266MHz, the price of SRAM is 10-100 times lower than TCAM [10][11]

destination is $\{a, b, c, d\}$. First, router a checks the 0_{th} bit in source address, and moves the pointer from the root to the 1_{st} level, then it passes the packet to router b. b checks the 1_{st} bit and passes the packet to c, which checks the 2_{nd} bit and concludes that the packet should be filtered. With the mechanism, each router requires fewer accesses to memory. When the network is large, the amortized burden on each router can be quite low.

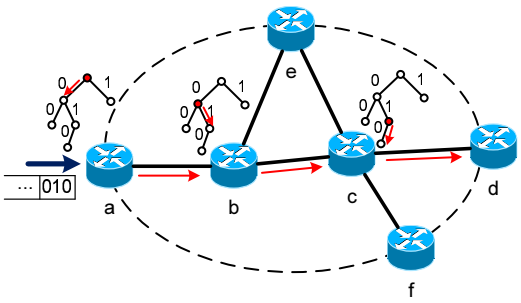

Fig. 1: Router a is the ingress and d, e, f are egress routers. Assume the source address has 3 bits, and there are 3 source filters: 1*, 00*, 010. The source filters are organized as uni-bit trie.

The above example is not special. In this paper, we generalize the above example by formulating a problem where we need to balance the load across different routers, given 1) the total bit set that has to be checked; 2) the spare capacity on each router that can be used for source filter; in other words, there is a limitation on the extra burden for each router. We develop an optimal solution based on dynamic programming.

We conduct comprehensive simulations using both real and BRITE generated topologies. The results show that our algorithm can largely reduce the number of bits each router has to check, and load is much more balanced across routers.We also conduct a case study using the topology of China Education and Research Network 2 (CERNET2), the world's largest IPv6 backbone network (including 59 Giga-PoPs). The results show that each router has only to check at most 40 bits out of 128-bit IPv6 address and the maximum utilization is much lower. We also evaluate the control overheads of our scheme, using real data traces, that recorded the topology changes of CERNET2 during four months. The results show that the control overheads caused by our scheme is quite low.

II. Related Work

Many solutions have been proposed to battle DoS problem and spoof problem with filters. For example, ingress access list [4], static ACL (Access Control List) is used to keep private source addresses out of the Internet [6]. However, most solutions store the blacklist in TCAM, which is a scarce resource in network. While the blacklist increases tremendously, TCAM-based filtering can not accommodate so many entries.

Due to lack of hardware memory space, many solutions tried to reduce the number of filters, [14] aggregated source prefixes to shorter prefixes. [22] proposed to use bloom filter to exclude suspected malicious traffic. However, reduced filters will cause collateral damage, i.e., legitimate traffic may also be blocked. To make efficient use of scarce TCAM resources, [18] studied the problem of filter selection as a resource allocation problem, [8] employed bayesian decision theory to optimize the set of filters based on history of attacking information.

In [18], distributed filtering was studied to reduce collateral damage. It places different filters on different routers, such that each router blocks different IP addresses along a path. However, the scheme has two drawbacks, 1) it is based on ACLs that resides in TCAM, but in most cases only border routers have ACLs for preventing malicious traffic into the network; 2) once the filter set changes, e.g., adding a new blocked IP address, or network topology changes, routers have to re-compute across different routers, which increases additional control overheads. Compared with [18], our scheme only re-computes when topology changes.

TABLE I: Comparison of different filtering schemes

Schemes \ Metrics	Lookup speed	Storage space	Overheads**
Centralized&Hardware	Fast	Low	
Centralized&Software	Slow	Large	
Distributed by filters &Hardware	Fast	Larger*	Higher
Distributed by filters &Software	Slow	Large	Higher
Distributed by bits &Software	Fast	Large	Lower

* The storage space is larger if routers along the path have ACLs, however, only border routers have ACLs practically.
** Control overheads caused by distributed schemes.

We compare all possible schemes (distributed by bits&hardware Scheme does not exist) in Table I. We can see that our scheme is the only one, that can achieve fast lookup speed, large storage space and low control overheads.

III. Design Overview

A. Assumptions

To restrict the scope of our study, we first make a few assumptions: 1) We assume the existence of blacklist, this can be constructed based on either history data [12] or attacking information from other hosts [17]. Constructing the blacklist is orthogonal to our paper; 2) We assume that we can insert additional information between IP and MAC header, (e.g., the same as MPLS), or in the option positions so as to carry some information between adjacent routers; 3) We assume the routers are less likely to be attacked and intra-domain communications are secure, despite our efforts to take fail-safe into account; 4) We admit that using our scheme, we may not prevent malicious traffic at the border routers. Some ISPs do tolerate the existence of malicious traffic inside their networks [15]. We will study this in more details in our future work so that we can focus on router cooperation in this paper.

B. Encoding the Trie and Header Format

Ingress routers can get a blacklist, i.e., a set of filters, through existing methods. Different ingress routers may have different blacklists, as they may be connected to attackers from different sources, they may also share the same blacklist.

Ingress routers will first distribute the filters to other routers in network. When any router obtains the filters, it should install

them in software, i.e., construct a trie. In this paper, we will focus on leaf-pushed uni-bit trie [20], where each trie node either points to a prefix index or a child node, and it is easy to extend this work to other tries, e.g., multi-bit trie.

After constructing the trie, routers should encode each node with a node index. The encoding results should be consistent across different routers, i.e., we should assign node index 1, 2, 3, ...to the trie nodes following some rule (please refer to [21]). After encoding the trie, the router should store the nodes linearly in memory, therefore, the router can get the trie node in constant time after obtaining the node index. Besides, the node index can be *empty*, i.e., previous routers have checked all bits and concluded that a packet should not be filtered. The empty state is encoded to be 0.

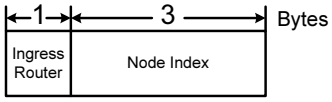

Fig. 2: The additional header format

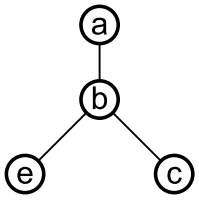

Fig. 3: Covering tree rooted at router a in Fig. 1, with maximum depth be 2.

An additional header has to be inserted into all data packets that travel through the network. Fig. 2 shows the format of the additional header, which has 32 bits, equivalent to MPLS header. The new header has two fields, the first field denotes the ingress router, and the second denotes the node index.

C. Data and Control Plane

1) Data Transmission: Upon receiving a packet, router should first obtain the fields of ingress router and node index from the additional header. If the node index is empty, then the router just delivers the packet to the next router. If the node index is non-empty, router finds the exact trie node according to the node index. The router also obtains the *delegated bits*, which is the bit set the router should check, then the router continues to look up the delegated bits in the trie. We will later discuss how the router obtains the delegated bits.

If the process finally arrives at some leaf node, indicating that the source address matches a filter in the blacklist, then router just discards the packet. If we can not search more deeply into the trie, indicating that the source address does not match any filter, then the node index is set to be empty. Else the node index is set to be the node index of the trie node where the lookup process stops. After setting the node index, the router delivers the packet to the next hop.

Ingress router looks up from the root node in the trie, and insert the additional header. Egress router removes the header.

2) Control Information: Combined with the trie and network topology information, we can compute the delegated bits on each router for traffic from different ingress routers (described in the next section), such that load is balanced among all routers. The computed results across different routers should satisfy that, all bits that appear in the trie should be checked orderly (from root to leaf nodes) by routers along any paths from ingress to egress routers.

There are three kinds of control information exchanged between routers. First, ingress router computes and distributes the delegated bits for each router. Second, ingress router should distribute the filters to other routers. thus each router can independently construct and encode the trie. When updating (inserting or deleting) on filters happens, the ingress router should notify other routers about the updates. Each update message should be tagged with a sequence number, so all routers can set up the trie consistently. Finally, the topology information can be distributed through protocols such as OSPF, and the spare capacity of each router can be known through extension of OSPF or a new protocol.

All control information is exchanged between adjacent routers, by setting TTL of control packets to be 255 [2], we can prevent spoofed message from attackers.

IV. Optimal Covering Scheme

To share the load among routers, we formulate and solve the problem of computing the delegated bits on each router for traffic from different ingress routers.

A. Problem Formulation

Let $G = (V, E)$ be a network, where V is the set of routers, and E is the set of links. Let $\mathcal{R}$ denote the set of ingress routers in the network. Let $\mathcal{L}$ be a path (i.e., an ordered set of routers), $\mathcal{P}^r$ ($r \in \mathcal{R}$) be the set of paths that a packet may traverse from r to any other egress routers. For $u, v \in \mathcal{L}$, define $u \preceq_{\mathcal{L}} v$ as node u be the predecessor of v on $\mathcal{L}$.

Let $\mathcal{T}^r = \{b_0, b_1, \ldots, \}$, ($r \in \mathcal{R}$, $0 \le b_i \le 31$ for IPv4, and $0 \le b_i \le 127$ for IPv6) be the ordered (i.e., $b_i < b_j$, if $i < j$) set of bits that should be checked for traffic from ingress router $r \in \mathcal{R}$. Each router only checks a few bits in source addresses, let $\mathcal{B}_v^r \subseteq \mathcal{T}^r$ be the delegated bits that v checks for traffic from r, and $\overrightarrow{B}^r = (\mathcal{B}_{v_1}^r, \mathcal{B}_{v_2}^r, \ldots), v_i \in V$ be a vector representing a *covering scheme* for traffic from r. Along all paths from ingress to egress, all routers cooperatively check $\mathcal{T}^r$ from higher[3] to lower bits. Each router in the network has limited capacity due to CPU limitations. We model this as the maximum number of additional bits in source addresses that the router can process. Let C_v be the capacity of v. To balance the load across the network, let $f^r(v)$ be the utilization function on node v for ingress router r, i.e., $f^r(v) = \frac{|B_v^r|}{C_v}$.

Considering some ISPs do not want malicious traffic penetrates deeply into their networks, we define the *maximum depth*, i.e., distance that malicious traffic could travel before being filtered. Let $d(u, v)$ be the distance between node u and v, k be the maximum depth, e.g., $k = 0$ if the ISP forces all traffic be filtered at border routers. ISP administrator can select k based on a tradeoff between security and load balancing. Then we can formulate the problem as follows,

$$\min \max_{v \in V, r \in \mathcal{R}} f^r(v) \tag{1}$$

$$\text{s.t. } |\mathcal{B}_v^r| \le C_v, \forall v \in V, r \in \mathcal{R} \tag{2}$$

$$\bigcup_{v \in \mathcal{L}} \mathcal{B}_v^r \supseteq \mathcal{T}^r, \forall \mathcal{L} \in \mathcal{P}^r, r \in \mathcal{R} \tag{3}$$

[2]TTL is set to be 255 if communication entities are directly connected

[3]Here, let the most significant bit be the highest order bit

$$\bigcup_{u \preceq_{\mathcal{L}} v} \mathcal{B}_u^r = \mathcal{T}^r \text{ or } \bigcup_{u \preceq_{\mathcal{L}} v} \mathcal{B}_u^r = \{p \in \mathcal{T}^r | p > q, \forall q \in \mathcal{B}_v^r\}, \quad \forall r \in \mathcal{R}, \mathcal{L} \in \mathcal{P}^r, v \in \mathcal{L} \tag{4}$$

$$\mathcal{B}_v^r = \emptyset, \forall r \in \mathcal{R}, \forall v, d(v, r) > k \tag{5}$$

Eq. (1) specifies the objective to minimize the maximum utilization on all routers. Eq. (2) indicates the constraint on capacity. Eq. (3) states that all bits must be covered along any path that a packet can traverse. Eq. (4) states that on any path from ingress to egress, if not all bits have been checked, then the successor node should check lower bits. Eq. (5) expresses the maximum depth the administrator set.

The solution to the problem is called the optimal covering scheme.

TABLE II: Notation List

Notation	Definition
V	set of nodes
$\mathcal{R}$	set of ingress routers
$\mathcal{L}$	a path
r	an ingress router
$\mathcal{P}^r$	set of paths that a packet can traverse from r to egress routers
$\mathcal{T}^r$	set of ordered bits that should be checked for packets from r
$\mathcal{B}_v^r$	set of bits that node v checks for packets from r
$\mathcal{B}^r$	a covering scheme for r
C_v	capacity of node v
$f^r(v)$	utilization function on node v for ingress router r

B. Finding the Optimal Covering Scheme

In this section, we will present the algorithm *Opt-Cover()* to find the optimal covering scheme by dynamic programming.

Note that in Eq. (1)-(5), variables for different ingress routers do not interact, So it can be decomposed to a number of subproblems, one for each ingress router.

We can obtain a spanning tree rooted at an ingress router and towards all egress routers, and the height of the tree is less than the maximum depth k. For example, suppose $k = 2$ in Fig. 1, then we can get a tree rooted at ingress router a as shown in Fig. 3. We call the spanning tree *covering tree*, which has to cover all bits that has to be checked, more specifically, all paths from root to any leaf nodes should cover all bits.

Intrinsically, if a tree has to cover i bits and the root checks the highest j bits, then each sub-tree rooted at children of the root nodes has to cover $i-j$ bits. Suppose we know the optimal cover for each sub-tree, then we can compute the optimal cover for the entire tree by trying different number of delegated bits on the root node. For example, if the tree in Fig. 3 has to cover 3 bits, and capacity of each node is 2. If node a checks 0 bit, sub-tree rooted at node b should cover 3 bits, the maximum utilization of the optimal cover for the sub-tree (b checks 2 bits and c, e each checks 1 bit, or b checks 1 bit and c, e each checks 2 bits) is 100%, and the maximum utilization of the optimal cover for the entire tree is 100%. If node a checks 1 bit, sub-tree rooted at b should cover 2 bits, the maximum utilization of the optimal cover for the sub-tree (b checks 1 bits and c, e each checks 1 bit) is 50%, and the maximum utilization of the optimal cover for the entire tree is 50%. If a checks more than 1 bit, the maximum utilization of the optimal cover will be more than 50%. Thus we can conclude that the optimal cover is: a checks the 0_{th} bit, b checks the 1_{st} bit, and c, e each checks the 2_{nd} bit.

Let $\mathcal{O}(v, n)$ be the Min-max utilization if covering tree rooted at node v has to cover n bits, $\mathcal{O}_j(v, n)$ be the Min-max utilization if covering tree rooted at node v has to cover n bits, and v itself has to check j bits. Let $\mathcal{N}(v, n)$ be the number of delegated bits on v if Min-max utilization is achieved on the covering tree that is rooted at v and has to cover n bits. Let $Parent(v)$ be the parent node of v. Algorithm Opt-Cover() computes the delegated bits of each node as follows.

Algorithm 1: Opt-Cover()

```
   Input      : T^r
   Output     : B_v^r, ∀v ∈ V
   Initialzation : B_v^r = ∅, ∀v ∈ V
1  begin
2      PreOrder traverse the Covering tree and push nodes into Stack
3      while Stack ≠ null do
4          v = Pop(Stack) for i = 0, 2, ..., |T^r| do
5              if v is a leaf node then
6                  O(v, i) = i / C_v,  N(v, i) = i;
7              else
8                  for j = 0, 2, ..., i do
9                      O_j(v, i) = j / C_v
10                     foreach child node u of v do
11                         O_j(v, i) = max{O_j(v, i), O_j(u, i − j)}
12                 O(v, i) = min_{j=0,2,...,i} O_j(v, i)
13                 N(v, i) = j, where O_j(v, i) = O(v, i)
14     if O(r, |T|) ≤ 100% then
15         PostOrder traverse the Covering tree and push node into
           Stack
16         while Stack ≠ null do
17             v = Pop(Stack)
18             len(v) =
               len(Parent(v)) + N(v, |T| − len(Parent(v)))
19             B_v^r = {T_i^r | |T^r| − 1 − len(v) ≤ i <
               |T^r| − 1 − len(Parent(v))}
```

The input of Algorithm Opt-Cover() is the set of bits that have to be checked for traffic from an ingress router, and the output is the delegated bits on each router. Basically, Algorithm Opt-Cover() first traverses from leaf nodes to root, following a dynamic programming structure. After computing the optimal cover, if the Min-max utilization is larger than 100%, then there does not exist a feasible solution. else the algorithm traverses from the root to leaf nodes and computes the delegated bits on each node.

Theorem 1: The complexity of Algorithm Opt-Cover() is $O(|V| \times (|\mathcal{T}^r|^2))$.

Proof: $Stack$ has less than $|V|$ nodes initially. The number of loops in line 4 and line 8 is less than $|\mathcal{T}^r|$. The number of loops in line 10 is bounded by a constant. ■

The complexity of Opt-Cover() is low, and linearly increases with V. Thus the algorithm adapts to large scale network.

C. Distributing the Optimal Covering Scheme

Configuring the delegated bits on all routers is almost impossible for a large scale network, we need a distribution

mechanism. Computed results should be distributed from ingress router towards egress routers. During setup time, the ingress router should first check all bits, and compute the optimal covering scheme. Then it will send the scheme to its children along the covering tree rooted at itself. The child receives the scheme, checks all bits except the delegated bits of its ancestors, then sends an acknowledgement (ACK for short) to its parent. After the parent receives ACKs from all children, the parent will check the delegated bits of the optimal covering scheme. The children will recursively repeat this process until egress routers receive the scheme.

Continue the example in Fig. 3, router a first checks all three bits, and passes the optimal covering scheme to b. b sets its delegated bits be 1_{st} and 2_{nd} bits and send ACK to a. a receives the ACK, and sets its delegated bits be 0_{th} bit. At the same time, b will send the optimal covering scheme to router c and e. c and e set their delegated bits be 2_{nd} bit and send ACKs to b. After receiving ACKs from both c and e, b will set its delegated bits be 1_{st} bit.

When topology changes, e.g., links/nodes fail, the upstream routers of the changed links/nodes should check all bits except the delegated bits of their ancestors. Thus malicious traffic will be filtered before arriving at the changed links/nodes. When ingress router finds the changes, it first checks all bits, computes the new optimal covering scheme, and repeats the distribution process.

Theorem 2: At any time, all bits will be covered by the covering tree.

Proof: If the covering tree only has one level, i.e., only one router, it will check all bits at any time. Suppose a covering tree that has k levels can cover all bits at any time. Then if a covering tree has $k+1$ levels, before the root router receives all ACKs from its children, root router will check all bits. During the time, all sub-trees, that root at its children and have less than $k+1$ levels, will cover all bits other than delegated bits of the root router. Thus when root router receives all ACKs and only checks the delegated bits, the covering tree can still cover all bits. ■

We can distribute the delegated bits across routers based on the extension of OSPF [23]. The protocol works in a similar way with OSPF, that has to re-distribute the link state database when topology changes. All control messages can be piggybacked in the extension of OSPF control messages. Thus our scheme will not increase the number of exchanged messages between routers.

D. Discussions on some practical issues

Configuration Cost: Our scheme may raise concerns regarding additional configuration cost. We have designed a light-weight protocol to distribute the covering scheme (Section IV-C); so we need to maintain a blacklist on the border router, and the configuration complexity is the same with traditional mechanism.

Topology and Filter Changes: Network paths are normally stable (topology changes daily [13]). In the case where the border router has to re-compute and distribute a new covering scheme, it may introduce congestion for a short period of time. In practice, we believe this is tolerable for current routers. Filters, or blacklist, changes more frequently. To prevent oscillation when the filter set changes, we set the total bit set (which should be checked) to be all bits in source address.

V. Performance Evaluation

A. Simulation Setup

Opt-Cover() is the key component of the mechanism that we designed. We evaluate the performance of the algorithm using both BRITE [1] generated topologies and real topologies. A case study on CERNET2 will be discussed in the next section.

1) BRITE Topology: We generate topologies with routers from 100 to 500. The average number of links per new router is set to be 2 to 10. We allow the most specific IP source address filtering, thus we set the number of bits that will be checked to be 32 (for IPv4). On each router, we set the capacity constraint to be 4 to 64. According to [9][5], we set the number of border routers to be 2 to 20. Among the border routers, we randomly select one as ingress router, and others as egress routers. We set the maximum depth k to be infinity, i.e., malicious traffic should be filtered inside the network. The default value and other parameters are in Table III.

TABLE III: Parameter table of brite-generated topologies

No. Routers	α / β	Placement	links/new router
300	0.15/0.2	Random	3
Mode	Model	No. Border Routers	k
Router Only	Waxman	5	$+\infty$

TABLE IV: Parameter table of real topologies

	No. Routers	No. Edges	Avg. links/new router
CERNET	110	238	2.16
AS 1221	104	151	1.45
AS 1239	315	972	3.09
AS 3257	161	328	2.04
AS 6461	128	372	2.90

2) Real Topology: We obtain the real topology of CERNET, which is a medium-scale IPv4 network with 110 routers and 238 links. We also obtain four real topologies (AS 1221, AS 1239, AS 3257, AS 6461) from Rocketfuel [19]. The details of real topologies are in Table IV.

We set ingress-base filtering as a benchmark for comparison. Our evaluation metric is the Min-max utilization (utilization for short) on all routers. The results are averaged by 100 independent and random experiments.

B. Simulation Results

Fig. 4 shows the relation between utilization and network size and compares Opt-Cover() with ingress-based filtering. We set the number of routers to be 100 to 500. In Fig. 4, we see that the utilization of Opt-Cover() decreases with network size. For example, when the network has 100 nodes, the utilization is 39.65%, when the network has 500 nodes, the utilization decreases to be 29.74%. This is because when the network size increases, the path from ingress to egress router becomes longer, thus more routers will share the load. Compared to ingress-based filtering, the utilization of Opt-Cover() is much lower. The utilization of Opt-Cover() stays below 40%, on

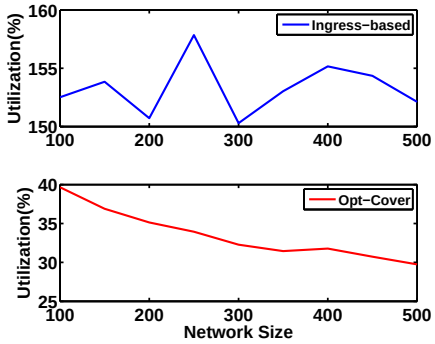

Fig. 4: Utilization as a function of network size.

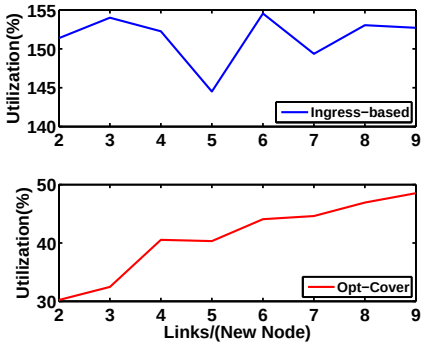

Fig. 5: Utilization as a function of network degree.

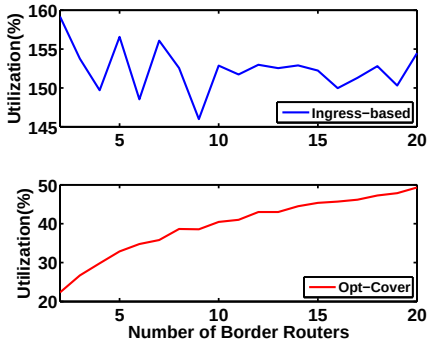

Fig. 6: Utilization as a function of number of border routers.

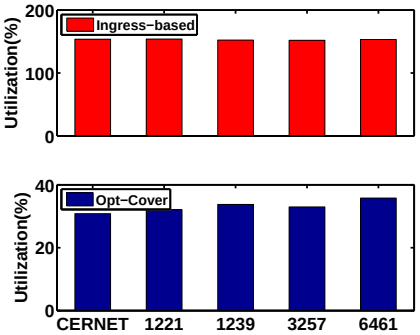

Fig. 7: Utilization on real topologies.

the contrary, the utilization of ingress-based filtering stays around 155% (here, we do not put limit on router capacity, i.e., utilization can exceed 100%). Actually, the ingress-based filtering is insensitive to network topology, including number of border routers, this is obvious because the ingress router takes all responsibility for filtering malicious traffic.

Fig. 5 shows the relation between utilization and network degree (links/(new router)). We see that the utilization of Opt-Cover() increases with network degree, this is because the path from ingress to egress becomes shorter when degree increases. However, considering the average degree of Internet is less than 3 (links/(new router)) [19][7], we believe that essential benefits can be obtained from our mechanism.

In Fig. 6, we study the impact of number of border routers on utilization. We see that the utilization of Opt-Cover() increases with the number of border routers. When there are 20 border routers, the utilization reaches almost 50%. This is because in Opt-Cover(), all paths from ingress to egress routers have to cover the bits that should be checked, thus more paths lead to more constraints on the optimal solution. However, according to [9], most networks have less than 10 border routers, thus our conclusion is the same as Fig. 5.

Fig. 7 shows the utilization of Opt-Cover() and ingress-based filtering under different topologies. The results are similar with results on BRITE generated topologies. Under all topologies, Opt-Cover() performs much better than ingress-based filtering. And the performances of Opt-Cover() on different topologies are related to the size and degree of topologies. For example, AS 6461 has only 128 nodes while the degree of it is almost 3.0, thus the utilization of Opt-Cover() on AS6461 is the highest (35.78%).

Note that we compute the average utilization based on multiple experiments. On average, our algorithm is much better than ingress-based filtering. Although our algorithm may not perform so well under some topologies, such as topologies that ingress router is adjacent with some egress routers. However, we believe that there exists a wide range of application for our mechanism.

VI. A Case Study

We conduct a case study with the real topology information of CERNET2. CERNET2 has two international exchange centers connecting to the foreign Internet, Beijing (CNGI-6IX) and Shanghai (CNGI-SHIX). We want to block malicious traffic between CNGI-6IX and CNGI-SHIX along a pre-defined path, i.e., {Beijing, Tianjin, Jinan, Hefei, Nanjing, Shanghai}. Each router has limited capacity, as shown in Fig. 9 (the top bar), estimated based on performance data of routers.

Traditional source filtering places the filters either in Beijing or Shanghai, and needs to check 128 bits (CERNET2 is an IPv6 network) in source addresses. The additional burden exceeds the left capacity. And the maximum utilization across all routers reaches 160% when filters are place in Beijing.

Fig. 9 (the bottom bar) shows the results computed by Opt-Cover(). Out of 128 bits, the routers check at most 40 bits (on the router in Shanghai) in source addresses. The maximum utilization is only 36.11%, on the router in Tianjin.

Considering the control overheads, in Figure 10, we plot the estimated number of re-computations, which will consume CPU resources on routers and cause re-distribution of new optimal covering scheme (see Section IV-C), using the real

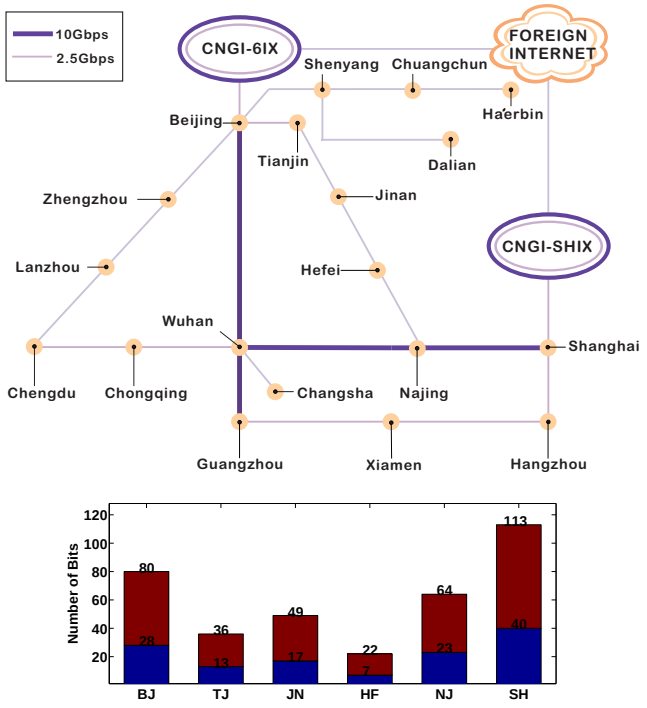

Fig. 9: Capacity of each router, and number of additional bits that each router has to check in source addresses.

data traces (topology changes during four months in 2010) from CERNET2. The maximum number of re-computations can reach several hundreds in one day. However, because the complexity of Algorithm 1 is low (it cost less than 1ms for one computation within CERNET2 topology, according to our experiments on our PCs with Intel Core i3 CPU 2.4GHz and 2.0G Memory), and re-distribution of new optimal covering scheme is piggybacked in OSPF messages. Considering the CPU utilization of CERNET2 routers is lower than 10%, we believe that the additional control overheads of our scheme are bearable within CERNET2.

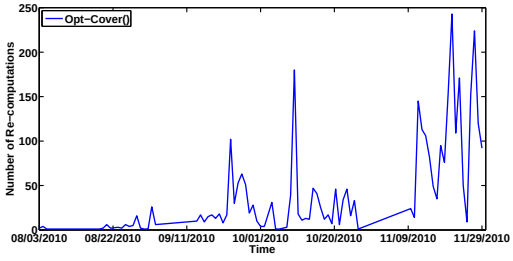

Fig. 10: Number of re-computations

VII. Conclusion and Future Work

In this paper we proposed a new cooperative filtering mechanism. Our scheme reduced the number of accesses to memory through cooperatively lookup by multiple routers along the path that packets traverse. We formulated the problem as finding a cooperative scheme such that the loads across routers are balanced, and guaranteeing the correctness by constraining that all bits should be checked when traversing the network. Through simulation, we showed that the Min-max utilization can be largely reduced through our mechanism.

There can be many future studies. We will conduct more comprehensive studies on the benefits that our mechanism will bring, including speeding up lookup process and reducing total delay. Beside, we will study the potential damages from letting malicious traffic penetrating into a network.

References

[1] Brite: Boston university representative internet topology generator. http://www.cs.bu.edu/brite.

[2] Dshield dataset. http://www.dshield.org.

[3] K. Argyraki and D. R. Cheriton. Scalable network-layer defense against internet bandwidth-flooding attacks. *IEEE/ACM Trans. Netw.*, 17:1284–1297, 2009.

[4] F. Baker and F. Savola. Ingress Filtering for Multihomed Networks. RFC 3704 (Best Current Practice), Mar. 2004.

[5] P. Barford, A. Bestavros, J. Byers, and M. Crovella. On the marginal utility of network topology measurements. In *Proc. ACM IMW'01*, San Francisco, California, USA, Nov 2001.

[6] R. Beverly, A. Berger, Y. Hyun, and k. claffy. Understanding the efficacy of deployed internet source address validation filtering. In *Proc. ACM IMC'09*, Chicago, Illinois, USA, Nov 2009.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proc. ACM SIGCOMM'99*, Cambridge, Massachusetts, United States, Aug 1999.

[8] M. Goldstein, C. Lampert, M. Reif, A. Stahl, and T. Breuel. Bayes optimal ddos mitigation by adaptive history-based ip filtering. In *Proc. IEEE ICN'08*, Cancun, Mexico, Apr 2008.

[9] B. Huffaker, A. Dhamdhere, M. Fomenkov, and k. claffy. Toward Topology Dualism: Improving the Accuracy of AS Annotations for Routers. In *Proc. PAM'10*, Zurich, Switzerland, Apr 2010.

[10] W. Jiang, Q. Wang, and V. Prasanna. Beyond tcams: An sram-based parallel multi-pipeline architecture for terabit ip lookup. In *Proc. IEEE INFOCOM'08*, Phoenix, AZ, Apr 2008.

[11] A. Liu, C. Meiners, and E. Torng. Tcam razor: A systematic approach towards minimizing packet classifiers in tcams. *Networking, IEEE/ACM Transactions on*, 18(2):490 –500, 2010.

[12] X. Liu, X. Yang, and Y. Lu. To filter or to authorize: network-layer dos defense against multimillion-node botnets. In *Proc. ACM SIGCOMM'08*, Seattle, WA, USA, Aug 2008.

[13] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot. Characterization of failures in an operational ip backbone network. *IEEE/ACM Trans. Netw.*, 16:749–762, 2008.

[14] G. Pack, J. Yoon, E. Collins, and C. Estan. On filtering of ddos attacks based on source address prefixes. In *Proc. IEEE Securecomm'06*, Baltimore, MD, Sep 2006.

[15] V. Sekar, R. Krishnaswamy, A. Gupta, and M. K. Reiter. Network-wide deployment of intrusion detection and prevention systems. In *Proc. ACM CoNext'10*, Philadelphia, Pennsylvania, Dec 2010.

[16] F. Soldo, K. El Defrawy, A. Markopoulou, B. Krishnamurthy, and J. van der Merwe. Filtering sources of unwanted traffic. In *Information Theory and Applications Workshop, 2008*, San Diego, California, Feb 2008.

[17] F. Soldo, A. Le, and A. Markopoulou. Predictive blacklisting as an implicit recommendation system. In *Proc. IEEE INFOCOM'10*, San Diego, CA, Mar 2010.

[18] F. Soldo, A. Markopoulou, and K. Argyraki. Optimal filtering of source address prefixes: Models and algorithms. In *Pro. IEEE INFOCOM'09*, Rio de Janeiro, Brazil, Apr 2009.

[19] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *Networking, IEEE/ACM Transactions on*, 12(1):2 – 16, 2004.

[20] V. Srinivasan and G. Varghese. Fast address lookups using controlled prefix expansion. *ACM Trans. Comput. Syst.*, 17:1–40, 1999.

[21] S. Yang, M. Xu, D. Wang, and J. Wu. Source address filtering for large scale network: A cooperative software mechanism design. Technical report, Tsinghua University, Aug 2011. http://www.wdklife.com/tech-110.pdf.

[22] F. Yi, S. Yu, W. Zhou, J. Hai, and A. Bonti. Source-based filtering scheme against ddos attacks. *International Journal of Database Theory and Application*, 1(1):2008–2011, 2008.

[23] A. Zinin, A. Roy, L. Nguyen, B. Friedman, and D. Yeung. OSPF Link-Local Signaling. RFC 1940 (Standards Track), Aug. 2009.