Conflict Detection in Internet Router Tables
†Technical Report – Albert-Ludwigs-Universität Freiburg

Christine Kupich and Khaireel A. Mohamed

Institut für Informatik†. Lehrstuhl Algorithmen & DatenStrukturen.
{kupich, khaireel}@informatik.uni-freiburg.de

August 4, 2006

Preamble. Packet filters are rules in IP router tables for classifying packets based on the information in their header fields. For forwarding purposes, there has to be a unique best matching filter which applies to an incoming packet p. In order to avoid ambiguities in the classification, the set of filters must be conflict-free under the tie-breaking rule which is applied. In this report we establish some properties of filters to provide a basis for our future work in this field. Furthermore, we give a review of the accomplishments in this area and outline open problems.

Keywords: filter conflict, detection, reporting, classification

1 Introduction

An internet packet p flowing into an internet router is assigned an action based on its header fields. The actions in the router table are rules that act as *filters* to direct p to its next destination. When the header fields of an incoming packet trigger several filters from the table, such that the router *cannot* determine one unique filter to classify p, then the set of filters in the router table is said to be conflicting. A variety of aspects exists as to how they collectively define the term 'filter conflict' [1, 2, 3], and there is still a considerable amount of discussions on how to optimally detect such conflicts in a given set of filters R, how to break the ties when confronted with a conflict situation, and how to 'resolve' unbreakable conflicting pairs of filters [4, 5, 6, 7]. Classifications of IP are particularly important to achieve optimal performance in routers handling both IPv4 and IPv6 protocols and have attracted a considerable amount of attention [6, 8, 9, 10, 11, 5, 12, 4, 13].

In this report, we do not consider the actions to be performed by the packets, but rather, draw on the geometric properties of the routing table and look upon it as containing a set of ranges R in d dimensions, for $d = 1, 2$. We shall use the generic terms 'range' and 'filter' interchangeably to refer to both the 1D and 2D packet filters. More precisely, we term a 1D-range as an 'interval' and a 2D-range as a 'region'. A filter $r \in R$ covers a specific range in the discrete universe U^d, where $U = \{0, \ldots, 2^v - 1\}$, for $v = 32$ in IPv4 and $v = 128$ in IPv6, such that $r|_{d=1} = [s, f]$, and $r|_{d=2} = ([s_x, f_x], [s_y, f_y])$, where $s, f, s_x, f_x, s_y, f_y \in U$. R is a *prefix set* and r is a *prefix range*, if all $r \in R$ are specified by address prefixes. That is, the ranges in each dimension have boundaries specified by prefix-bits in the form of $[b_1 \ldots b_k 0 \ldots 0, b_1 \ldots b_k 1 \ldots 1]$ for $b_1, \ldots, b_k \in \{0, 1\}$.

† Fakultät für Angerwandte Wissenschaften. Georges-Köhler-Allee 51. D-79110, Freiburg, Deutschland.

A point $p \in U^d$ is a d-dimensional packet to be classified, so that any filter $r \in R$ *applies* to p if $p \in r$. We also say that p *stabs* r, or p *matches* r. If there is ambiguity in classification, i.e. several filters match an incoming packet p, tie-breaking rules are used to determine which filter p is assigned to. Two popular tie breakers are the *highest-priority-filter* (HPF) rule, and the *most-specific tie-breaking* rule (MSTB) (e.g. see [4]). The former assigns priorities to all $r \in R$ and selects the range with highest priority matching a given point. Clearly, this rule is only feasible if the set R of ranges is conflict-free in the following sense: for every point p there is exactly one $r \in R$ with the highest priority which applies to p. The latter considers all filters to have no priority, and selects the **most specific** $r \in R$ that applies to p (see Section 2).

2 Detecting and Reporting Conflicts

Let R be a set of ranges under MSTB. The MSTB rule is only feasible if the set is conflict-free in the following sense.

Definition 1. *A set of ranges R is* conflict-free *iff for each point p there is a unique range $r \in R$ such that $p \in r$ and for all other ranges $s \in R$ stabbed by p, $r \subset s$.*

Definition 2. *The range r is more specific than the range s iff $r \subset s$ [14].*

If the set R of ranges is conflict-free then for each point p there is a well defined most specific filter in R which applies to p.

Definition 3. *Two filters r and s* intersect *if $r \cap s \neq \emptyset$ but neither $r \subseteq s$ nor $s \subseteq r$. A set R of filters is* nested *if for any pair $r, s \in R$, either $r \subseteq s$ or $s \subseteq r$, or $r \cap s = \emptyset$.*

Lemma 1. *A set of 1D prefix filters is always nested.*

Proof. Refer to Sahni and Kim [12].

The reverse of the above lemma is not true:

Lemma 2. *Not every nested set of intervals is a set of prefix filters.*

Proof. By counterexample. The ranges $[3, 9]$ and $[3, 6]$ are nested and do not correspond to any prefix.

It is obvious that the MSTB rule applies for every nested set of 1D or 2D filters. There may be, however, conflict-free sets of ranges which are not nested. Consider, e.g. a set R of three ranges $\{r, s, t\}$, where r and s intersect and $t = r \cap s$.

Definition 4. *Two ranges $r, s \in R$ are in conflict with respect to R if r and s intersect and there is a point p such that p applies to r and s ($p \in r \cap s$) but there is no range $t \in R$ such that p applies to t and $t \subset r$ and $t \subset s$.*

Definition 5. *Let $R = \{r_1, \ldots, r_n\}$ be a set of n ranges. Then $\prod(R) = \bigcup_{i=1}^{n} r_i$ (see [14]).*

For $A = \{2, 5], [3, 6], [8, 9]\}$, $\prod(A) = \{2, 3, 4, 5, 6, 8, 9\}$, and for $B = \{[4, 8], [7, 9]\}$, $\prod(B) = \{4, 5, 6, 7, 8, 9\}$. $\prod(A)$ is not a range, $\prod(B)$ is the range $[4, 9]$.

Lemma 3. *If two ranges $r, s \in R$ are in conflict then there is no subset $S \subseteq R$ such that $\prod(S) = r \cap s$.*

Proof. Let r, s be two conflicting ranges. Consider the overlapping range $r \cap s$ and assume that there is a subset $S \subseteq R$ such that $\prod(S) = r \cap s$. Take $t \in S$ and $p \in t$ arbitrarily. Then r, s and t apply to p and $t \subset r$ and $t \subset s$ contradicting the assumption that r and s are in conflict.

The reverse of the above lemma is also true. We show:

Lemma 4. *If $r, s \in R$ are conflict-free then either $r \cap s = \emptyset$ or $r \subseteq s$ or $s \subseteq r$ or there is a subset $S \subseteq R$ such that $\prod(S) = r \cap s$.*

Proof. It is sufficient to assume that $r \cap s \neq \emptyset$ and neither $r \subseteq s$ nor $s \subseteq r$. Because r and s are conflict-free, there exists for each $p \in r \cap s$ a range $t_p \in R$ such that $p \in t_p$ and $t_p \subset r$ and $t_p \subset s$. Choose $S = \bigcup_{p \in r \cap s} t_p$. Then $\prod(S) = r \cap s$.

Hari *et al.* 's [1] definition of a resolve filter for 2D prefix filters naturally translates to 1D.

Definition 6. *Let $r, s \in R$ be two conflicting ranges. Then we call the overlapping range $r \cap s$ the* resolve filter *for r and s with respect to R. We denote by* resolve (R) *the set obtained from R by adding a resolve filter for every pair of conflicting filters in R.*

Definition 7. *A filter $r = [r.lo, r.hi]$* left-*overlaps filter $s = [s.lo, s.hi]$ iff $r.lo < s.lo < r.hi < s.hi$.*

Definition 8. *A filter r* right-*overlaps s iff $s.lo < r.lo < s.hi < r.hi$.*

Lemma 5. *Let R be a set of 1D ranges. Then $R \cup resolve(R)$ is a conflict-free set of filters.*

Proof. We prove this by induction.

Basis : The basis of the induction is a set of two ranges and the resolve filter induced by them. There are three cases, **(i)** the two ranges do not overlap, **(ii)** one of them encloses the other, and **(iii)** they overlap. There is no need to introduce any resolve filter in the first two cases and hence the theorem holds. It is easy to see that the resolve filter introduced in the last case does not introduce any conflict.

Figure 1. Intervals and their relative position.

Induction : Suppose we have a set of $i - 1$ ranges and their resolve filters already introduced and this set is conflict-free. We have to prove that the resulting set is still conflict-free when we introduce a new range in this set as well as the resolve filters for this new range. We refer to Fig. 1 for the proof.

All the ranges existing before the introduction of the new range can be classified into five categories : **(i)** the ranges that left overlap the new range, **(ii)** the ranges that right overlap the new range, **(iii)** the ranges that enclose the new range, **(iv)** the ranges that are enclosed by the new range, and **(v)** the ranges that do not overlap the new range.

Note that the ranges in category **(v)** or their resolve filters cannot cause any conflict after the introduction of the new range, because none of these resolve filters overlap the newly introduced range. Similarly, the ranges or their resolve filters in category **(iv)** do not introduce any conflict as these ranges and their resolve filters are nested within the newly introduced range.

Hence, we have to concentrate on the ranges in the first three categories. However, the proof is almost similar for the ranges in categories **(i)** and **(ii)** and we will only consider the proof for ranges in category **(i)**, i.e., the ranges that left overlap the new range and category **(iii)**, i.e., the ranges that enclose the new range. According to our definition of conflict, left- or right-overlap does not necessarily cause a conflict. We only insert resolve filters when there is a conflict, otherwise, we only insert the new range. For each of these categories, we have to prove, **(a)** the old resolve filters do not conflict with the resolve filters introduced for the new range, and **(b)** the old resolve filters do not conflict with the new range. If the new range overlaps with existing ranges but there is no conflict, then **(a)** is straightforward to prove, since in this case there are no newly inserted resolve filters. Hence in the following, we will only consider ranges that overlap **and** conflict. We denote all the ranges in category **(i)** by L. Suppose we are adding the i-th range r_i and its low and high end points are l_i and h_i respectively. Consider a range $r_k \in L$ that left overlaps and conflicts with r_i. The low and high end points of r_k are l_k and h_k. We can sort all the ranges that left overlap r_i according to their right end points. After the introduction of r_i, we introduce a resolve filter for a range $r_k \in L$ (as r_k conflicts with r_i) and this resolve filter is a new range $[l_i, h_k]$. Clearly, all these new resolve filters (we call this set RF_i) due to r_i are not in conflict as they are nested.

Consider now the resolve filters (we call this set RF_{i-1}) that existed due to the ranges in L before the introduction of r_i. We have to check whether the resolve filters in RF_{i-1} are in conflict with the resolve filters in RF_i. We prove by contradiction that there is no such conflict. Assume that there is a resolve filter $rf_{jk} \in RF_{i-1}$ such that rf_{jk} has a conflict with some resolve filter in RF_i. rf_{jk} was introduced for resolving the conflict of two ranges r_j and r_k before the introduction of r_i. Suppose rf_{jk} conflicts with $rf_{im} \in RF_i$. Clearly, the left end point of rf_{jk} is to the left of l_i as both r_j and r_k left overlap r_i. The left end point of rf_{im} is l_i as this resolve filter was introduced due to r_i. Consider now rf_{im} and the part of rf_{jk} to the right of l_i (the part of rf_{jk} to the left of l_i remains conflict-free after the introduction of r_i). See Fig. 2.

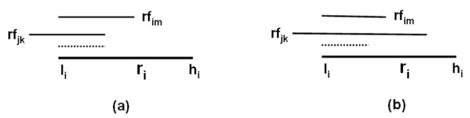

Figure 2. Any two ranges in RF_i and RF_{i-1} are conflict-free.

There are two cases as shown in Fig. 2, depending on the relative positions of the end points of r_j, r_k and r_m. First, we consider **(a)**. In this case, the right end point of rf_{im} is to the right of the right end point of rf_{jk}. Note that the right end point of rf_{jk} is either due to the right end point of r_j or due to the right end point of r_k (as rf_{jk} is a resolve filter for r_j and r_k). Suppose, without loss of generality, the right end point of rf_{jk} is due to the right end point of r_j. Then we have already introduced a resolve filter that starts at l_i and ends at the high end point of r_j, since we have added a resolve filter to resolve the conflict between r_j and r_i after we added r_i. This resolve filter for r_i and r_j is shown by the dashed line in Fig. 2(a). This resolve filter for r_i and r_j resolves the conflict between rf_{im} and rf_{jk}. Next, consider case **(b)**, i.e. where the right end point of rf_{im} is to the left of the right end point of rf_{jk}. Clearly, there is no conflict, as rf_{im} is nested in rf_{jk}. We can prove in a similar way that the resolve filters in RF_{i-1} do not conflict with the new range r_i.

Finally, we have to consider the ranges in category **(iii)**, i.e., the ranges that enclose r_i. Suppose, r_p is such a range. Clearly, r_i is not in conflict with r_p. Hence, **(a)** is straightforward to prove. However, we have to check whether any resolve filter due to r_p is in conflict with r_i. Suppose there is a range r_l which has a conflict with r_p and we have introduced a resolve filter earlier to resolve this conflict. Clearly, if $r_l \cap r_i = \emptyset$ then their resolve filter does not conflict with r_i. There are three possibilities left as shown in Fig. 3.

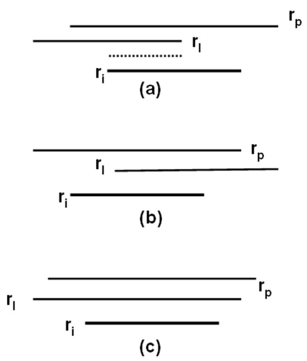

Figure 3. A range that encloses R_i cannot have any conflict due to the introduction of R_i.

In the first case, the resolve filter due to r_p and r_l has a conflict with r_i (Fig. 3(a)). However, in this case r_l left overlaps r_i and hence we have already introduced a resolve filter (shown by the dashed line in Fig. 3(a)) to resolve the conflict between r_l and r_i which also resolves the conflict between r_i and the resolve due to r_p and r_l. In the second case, r_l right overlaps r_i and the proof is similar. In the last case r_l encloses r_i and hence the resolve filter for r_p and r_l also encloses r_i.

This concludes the proof.

Fig. 4 shows an example for a set R of 1D filters: r and s are not conflicting, because for each point $p \in r \cap s$ there is a range $t \in R$ such that $p \in t$ and $t \subset r$ and $t \subset s$. However, the pairs (a,b), (b,c), (c,d) are all conflicting pairs of filters. Hence, R is not conflict-free. Adding $a \cap b, b \cap c, c \cap d$ to R results in a conflict-free set of filters, i.e. $R \cup resolve(R) = R \cup \{a \cap b, b \cap c, c \cap d\}$ is conflict-free.

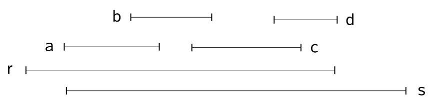

Figure 4. Filters r and s are conflict-free. However, the set as a whole is not conflict-free under MSTB.

Note that the above definition of conflicting ranges is fairly weak. However, with the MSTB rule in mind, we know for each point $p \in r \cap s$, r, s two nonconflicting but intersecting ranges, that neither r nor s is the filter determining the routing of p. Intuitively speaking, we consider only those pairs of filters as conflicting in R if there are points in the intersection for which classification can not be transfered to more specific ranges.

This definition of conflict implies that there may be sets of n 1D ranges having $\Omega(n^2)$ pairs of conflicting ranges. Fig. 5 shows an example of a set R of n ranges. Here, any pair $r_i, r_j \in R, i \neq j$ is conflicting with respect to R. $Resolve(R) = \{r_i \cap r_j | 1 \leq i, j \leq n, i \neq j\}$ contains $\Omega(n^2)$ elements. Though $R \cup resolve(R)$ is conflict-free according to Lemma 5 it is not necessary to add $\Omega(n^2)$ resolve filters to R in order to make it conflict-free. We can show:

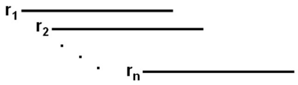

Figure 5. Any pair $r_i, r_j \in R, i \neq j$ is conflicting with respect to R. However, only a linear number of resolve filters is necessary to make the set R conflict-free.

Lemma 6. *For every set R of n 1D ranges, there is a set S of $O(n)$ 1D ranges s.t. $R \cup S$ is conflict-free.*

Proof. Let $R = \{r_1, \ldots, r_n\}$. These ranges partition the universe U into at most $2n$ consecutive slabs defined by the endpoints of the ranges. Let $ep_0, ep_1, \ldots, ep_k$ be the boundaries of these slabs. Let $\sigma = \{[ep_i, ep_{i+1}], 0 \leq i < k\}$. Then $R \cup \sigma$ is obviously conflict-free.

3 Previous Work

The conflict detection problem occurs in two variants; the *offline* and *online* modes. Algorithms for the offline version are applied directly to a given set R containing n ranges, and are expected to report only as many resolve filters as are necessary, which means, never more than $O(n)$, should the set R not be conflict-free. The online version, on the other hand, is a gradual build-up of a conflict-free set R; such that for every insertion and deletion of a range r, a check is made on the current status of R so that if conflicts occur, the algorithms will offer solutions to resolve them and maintain the conflict-free property of R.

3.1 Offline conflict detection and resolution

Let us consider solving the offline variant of the problem for a given set R of n 1D ranges under MSTB. The ranges divide the discrete universe into $O(n)$ *slabs* defined by their endpoints. R is conflict-free iff in each slab there exists a well-defined *most specific* interval. Naïvely, we can sort the intervals in each slab in decreasing order of lengths, find the shortest one, and test whether or not it is contained in all other intervals covering the slab. This trivial brute force method is not very efficient. The worst case gives $O(n)$ intervals in each slab, making it necessary to perform $O(n)$ inclusion tests for each slab. This results in the algorithm taking at least $\Omega(n^2)$ time. We consider an output-sensitive algorithm to be only dependent on the input and on the number of conflicts or the number of resolve filters that are necessary to make the set conflict-free. The 2D problem under MSTB is simplified (as to output-sensitivity) when we only consider the set F of prefix-filters [2]. Two filters $f, g \in F$ conflict iff an edge of f perfectly crosses an edge of g; that is, two edges *perfectly cross* iff they cross, and their crossing point is not an endpoint. In other words, a proper-edge intersection between two ranges in F is a direct cause for conflict. This implies that all conflicts in F can be detected (and possibly resolved) by computing all proper intersecting pairs of filters in F (and determining their regions of overlap). Therefore, we get an output-sensitive algorithm if all pairs of properly intersecting ranges in F are detected and reported. This can be done by a slight modification of the classical sweep-line algorithm for reporting all intersections in a set of n iso-oriented line-segments, which gives a runtime of $O(n \log n + k)$ to report all k conflicts in F. Lu and Sahni [2] further discuss the problem of resolving conflicts by adding a set of resolve filters to F. Such that for each conflicting prefix-pair $f, g \in F$, we add a new resolve filter $h = f \cap g$ to F. The resulting set of $F \cup resolve(F)$ is conflict-free under MSTB (Fig. 6(a)).

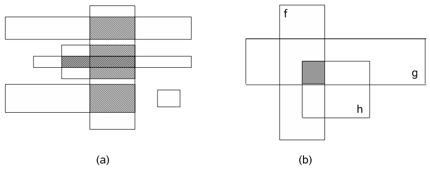

Figure 6. (a) Conflict-free set of prefix filters. (b) Conflicting set of arbitrary 2D range filters.

However, if F is not a prefix set, then the above discussions on resolving conflicts in F becomes insufficient. Fig. 6(b) gives an example by Lu and Sahni [2], where if $F = \{f, g, h\}$ and $resolve(F) = \{f \cap g, f \cap h, g \cap h\}$, then $F \cup resolve(F)$ is *still not* conflict-free. In fact, the set $resolve(F)$ contributes to the cause of conflict, and that for any point p in the shaded region, there is still no well-defined most specific filter to match it!

To make a set R conflict-free can often be met by adding to R only a subset of the filters in $resolve(R)$. Lu and Sahni [2] coined the notion of an **essential resolve filter**: A filter $f \in resolve(R)$ is an **essential resolve filter** iff $R \cup resolve(R) - \{f\}$ has no subset whose union equals f. Let $essential(R)$ be the set of essential resolve filters of R. To determine $essential(R)$ they construct a 2D trie for $R \cup resolve(R)$ and perform a trie traversal to identify $essential(R)$ in $O((n + s)W)$ time.

The only known result for arbitrary **prioritised** ranges in 2D is the $O(n^{3/2})$ result by Eppstein and Muthukrishnan [6]. They construct a 2-d-tree for the given set of ranges and use it in such a way that not all possible pairs of ranges have to be checked for solving the offline version of the conflict detection problem. But their algorithm yields only a *yes* or *no* answer to the offline version of the conflict detection problem and does neither report nor resolve the conflicts.

3.2 Online conflict detection and resolution

For a given nested set R of n 1D non-intersecting ranges under MSTB, the deletion of intervals maintains the conflict-free property of R. Hence, only the insertion of a new interval may become critical for the online variant of the problem. Lu and Sahni [14] show that storing the set R in two PST's is sufficient to detect all conflicts. However, the second PST is maintained exclusively for the detection of right-overlappings. Datta and Ottmann show that the same task can be performed using one min- and one max-augmented range tree [3]. These structures have the advantage that an underlying balancing scheme can be chosen freely and that operations are strictly local, which makes it possible to decouple the update and the rebalancing operations. However, in both cases, the actual information contained in the second structures is redundant. Lauer *et al.* [15] show that such an additional structure is not required. Lu and Sahni [14] further discuss this problem for sets of "conflict-free" 1D ranges under MSTB. In this case, both operations to insert and delete an interval may lead to conflicts in the resulting set. They consider two ranges r and s as conflict-free, iff r and s **(a)** are disjoint, **(b)** nested or **(c)** intersect and for each point p in $r \cap s$ there is a most specific range s that matches p. Collectively, the ranges s constitute what they call a *resolving subset* (see Fig. 7).

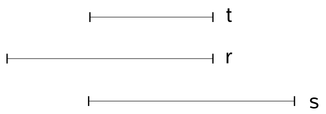

Figure 7. Ranges r and s have a resolving subset, i.e. they are conflict-free.

In the case of insertion, the new interval may *left-* or *right*-overlap with one or more intervals in R, such that the overlapping range has no resolving subset. Similarly, a conflict may arise if we remove $t = r \cap s$ from R, because $t \in R$ is an interval that resolves a conflict between $r, s \in R$. It is still open whether or not their rather complex solution can be simplified.

Hari *et al.* [1] consider the 2D online version of conflict detection and resolution for sets of prefix filters. They proposed the scheme of adding so-called "resolve filters" for conflict resolution. Their approach utilizes two recursive tries and adds resolve filters for each pair of conflicting, i.e. overlapping, filters, which guarantees a MSTB match. Reporting conflicts takes $O(W^2 + k)$ time, where W is the length of the longest prefix ($W \leq 128$ in IPv6), and k is the number of conflicts between a new filter f and a given set S of n filters. If f doesn't conflict it is added to S in $O(W)$ time.

Lu and Sahni [2] stated that the strategy by Hari *et al.* [1] to make a set F of 2D range-filters conflict-free does not work. This is not true because, while the strategy gives the wrong solution for the offline problem, it works correctly for the online variant of the problem considered in [1]. Consider Fig. 6(b). Let first f, then g and finally h be added to an initially empty set. Inserting f does trivially not create a conflict. However, when inserting g, a conflict is detected and $r = f \cap g$ is inserted as resolve filter. When inserting h, a conflict with f, g and $f \cap g$ is detected. Hence, three resolve filters will be added, particularly $h \cap r$ (shaded region).

4 Open Problems and Related Work

The offline and online version of conflict detection for non-prefix 2D ranges under MSTB is still open. As mentioned in [1], online conflict detection in d dimensions is related to a problem studied in computational geometry, namely that of d-dimensional rectangle intersection: for a query rectangle q find all rectangles in a set of n filters which intersect q. Edelsbrunner [16] has proposed a data structure that solves this problem in worst case time $O((\log n)^{2d-1} + k)$, where k is the number of rectangles that intersect q. However, while nested filters do not conflict, the corresponding rectangles are considered as intersecting from the geometric point of view. Thus, the number of rectangle "intersections" can be much larger than the actual number of filter conflicts. As mentioned above, the 2D offline case under HPF is not completey solved, since Eppstein and Muthukrishnan [6] only report whether the set conflicts or not, without offering conflict resolution. The 2D online version under HPF appears to be completely open. The problem here, as well as in the offline case, is illustrated in Fig. 8: intersecting ranges all of same priority, and one large rectangle of higher priority covering the intersections.

References

1. A. Hari, S. Suri, and G. Parulkar, "Detecting and resolving packet filter conflicts," in *INFOCOM 2000: Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.* IEEE Press, 2000, pp. 1203–1212.

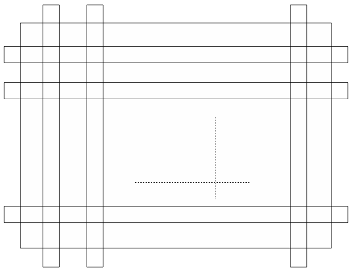

Figure 8. Conflict-free set of prioritized 2D range filters: large range f of high priority covers conflicting ranges of lower priority.

2. H. Lu and S. Sahni, "Conflict detection and resolution in two-dimensional prefix router tables," *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp. 1353–1363, 2005.
3. A. Datta and T. Ottmann, "A note on the IP table lookup problem," Dec. 2004, unpublished.
4. S. Sahni, K. S. Kim, and H. Lu, "Data structures for one-dimensional packet classification using most-specific-rule matching," in *ISPAN 02: Proceedings of the 2002 International Symposium on Parallel Architectures, Algorithms and Networks.* Washington, DC, USA: IEEE Computer Society, 2002, p. 3.
5. S. Sahni and K. S. Kim, "Efficient construction of multibit tries for IP lookup," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 650–662, 2003.
6. D. Eppstein and S. Muthukrishnan, "Internet packet filter management and rectangle geometry," in *SODA 01: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001, pp. 827–835.
7. F. Baboescu and G. Varghese, "Fast and scalable conflict detection for packet classifiers," *Computer Networks*, vol. 42, no. 6, pp. 717–735, 2003.
8. P. Gupta and N. McKeown, "Algorithms for packet classification," *Network*, vol. 15, no. 2, pp. 24–32, 2001.
9. H. Kaplan, E. Molad, and R. E. Tarjan, "Dynamic rectangular intersection with priorities," in *STOC 03: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing.* New York, NY, USA: ACM Press, 2003, pp. 639–648.
10. K. S. Kim, H. Lu, and S. Sahni, "Prefix and interval-partitioned dynamic IP router-tables," *IEEE Transactions on Computers*, vol. 54, no. 5, pp. 545–557, 2005.
11. H. Lu and S. Sahni, "Enhanced interval trees for dynamic IP router-tables," *IEEE Transactions on Computers*, vol. 53, no. 12, pp. 1615–1628, 2004.
12. S. Sahni and K. S. Kim, "An O(log n) dynamic router-table design," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 351–363, 2004.
13. P. Warkhede, S. Suri, and G. Varghese, "Multiway range trees: scalable IP lookup with fast updates," *Computer Networks*, vol. 44, no. 3, pp. 289–303, 2004.
14. H. Lu and S. Sahni, "O(log n) dynamic router-tables for prefixes and ranges," *IEEE Transanctions on Computers*, vol. 53, no. 10, pp. 1217–1230, 2004.
15. T. Lauer, T. Ottmann, and A. Datta, "Update-efficient data structures for dynamic IP router tables," *International Journal of Foundations of Computer Science*, submitted for publication.
16. H. Edelsbrunner, "A new approach to rectangle intersections, part ii," *International Journal of Computer Mathematics*, vol. 13, pp. 221–229, 1983.