Chapter 2
Background

We now formally define the concepts of fields, packets, and packet classifiers. A field $F_i$ is a variable of finite length (i.e., of a finite number of bits). The domain of field $F_i$ of $w$ bits, denoted $D(F_i)$, is $[0, 2^w - 1]$. A packet over the $d$ fields $F_1, \ldots, F_d$ is a $d$-tuple $(p_1, \ldots, p_d)$ where each $p_i$ $(1 \leq i \leq d)$ is an element of $D(F_i)$. Packet classifiers usually check the following five fields: source IP address, destination IP address, source port number, destination port number, and protocol type. The lengths of these packet fields are 32, 32, 16, 16, and 8, respectively. We use $\Sigma$ to denote the set of all packets over fields $F_1, \ldots, F_d$. It follows that $\Sigma$ is a finite set and $|\Sigma| = |D(F_1)| \times \cdots \times |D(F_d)|$, where $|\Sigma|$ denotes the number of elements in set $\Sigma$ and $|D(F_i)|$ denotes the number of elements in set $D(F_i)$.

A rule has the form \langle predicate \rangle \rightarrow \langle decision \rangle. A \langle predicate \rangle defines a set of packets over the fields $F_1$ through $F_d$, and is specified as $F_1 \in S_1 \land \cdots \land F_d \in S_d$ where each $S_i$ is a subset of $D(F_i)$ and is specified as either a prefix or a nonnegative integer interval. A prefix $\{0, 1\}^k\{\ast\}^{w-k}$ with $k$ leading 0s or 1s for a packet field of length $w$ denotes the integer interval $[\{0, 1\}^k\{0\}^{w-k}, \{0, 1\}^k\{1\}^{w-k}]$. For example, prefix 01** denotes the interval $[0100, 0111]$. A rule $F_1 \in S_1 \land \cdots \land F_d \in S_d \rightarrow \langle decision \rangle$ is a prefix rule if and only if each $S_i$ is represented as a prefix.

A packet matches a rule if and only if the packet matches the predicate of the rule. A packet $(p_1, \ldots, p_d)$ matches a predicate $F_1 \in S_1 \land \cdots \land F_d \in S_d$ if and only if the condition $p_1 \in S_1 \land \cdots \land p_d \in S_d$ holds. We use $DS$ to denote the set of possible values that $\langle decision \rangle$ can be. Typical elements of $DS$ include accept, discard, accept with logging, and discard with logging.

A sequence of rules $\langle r_1, \ldots, r_n \rangle$ is complete if and only if for any packet $p$, there is at least one rule in the sequence that $p$ matches. To ensure that a sequence of rules is complete and thus a packet classifier, the predicate of the last rule is usually specified as $F_1 \in D(F_1) \land \cdots \land F_d \in D(F_d)$. A packet classifier $C$ is a sequence of rules that is complete. The size of $C$, denoted $|C|$, is the number of rules in $C$. A packet classifier $C$ is a prefix packet classifier if and only if every rule in $C$ is a prefix rule. A classifier with $d$ fields is called a $d$-dimensional packet classifier.

Two rules in a packet classifier may overlap; that is, a single packet may match both rules. Furthermore, two rules in a packet classifier may conflict; that is, the two
rules not only overlap but also have different decisions. Packet classifiers typically resolve such conflicts by employing a first-match resolution strategy where the decision for a packet \( p \) is the decision of the first (i.e., highest priority) rule that \( p \) matches in \( \mathbb{C} \). The decision that packet classifier \( \mathbb{C} \) makes for packet \( p \) is denoted \( \mathbb{C}(p) \).

We can think of a packet classifier \( \mathbb{C} \) as defining a many-to-one mapping function from \( \Sigma \) to \( DS \). Two packet classifiers \( \mathbb{C}_1 \) and \( \mathbb{C}_2 \) are equivalent, denoted \( \mathbb{C}_1 \equiv \mathbb{C}_2 \), if and only if they define the same mapping function from \( \Sigma \) to \( DS \); that is, for any packet \( p \in \Sigma \), we have \( \mathbb{C}_1(p) = \mathbb{C}_2(p) \). A rule is redundant in a classifier if and only if removing the rule does not change the semantics of the classifier. Furthermore, we define the equivalence relation that classifier \( \mathbb{C} \) defines on each field domain and the resulting equivalence classes. We use the notation \( \Sigma_{-i} \) to denote the set of all \( (d-1) \)-tuple packets over the fields \( (F_1, \ldots, F_{i-1}, F_{i+1}, \ldots, F_d) \) and \( p_{-i} \) to denote an element of \( \Sigma_{-i} \). Then we use \( \mathbb{C}(p_i, p_{-i}) \) to denote the decision that packet classifier \( \mathbb{C} \) makes for the packet \( p \) that is formed by combining \( p_i \in D(F_i) \) and \( p_{-i} \).

Definition 2.1 (Equivalence Class). Given a packet classifier \( \mathbb{C} \) over fields \( F_1, \ldots, F_d \), we say that \( x, y \in D(F_i) \) for \( 1 \leq i \leq d \) are equivalent with respect to \( \mathbb{C} \) if and only if \( \mathbb{C}(x, p_{-i}) = \mathbb{C}(y, p_{-i}) \) for any \( p_{-i} \in \Sigma_{-i} \). It follows that \( \mathbb{C} \) partitions \( D(F_i) \) into equivalence classes. We use the notation \( \mathbb{C}\{x\} \) to denote the equivalence class that \( x \) belongs to as defined by classifier \( \mathbb{C} \).

In a typical packet classifier rule, the fields of source IP, destination IP, and protocol type are specified in prefix format, which can be directly stored in TCAMs; however, the remaining two fields of source port and destination port are specified as ranges (i.e., non-negative integer intervals), which are typically converted to prefixes before being stored in TCAMs. This leads to range expansion, the process of converting a non-prefix rule to prefix rules. In range expansion, each field of a rule is first expanded separately. The goal is to find a minimum set of prefixes such that the union of the prefixes corresponds to the range (see Algorithm 1). For example, if one 3-bit field of a rule is the range \([1, 6]\), a corresponding minimum set of prefixes would be 001, 01*, 10*, 110. The worst-case range expansion of a \( w \)-bit range results in a set containing \( 2w - 2 \) prefixes [Gupta and McKeown(2001)]. The next step is to compute the cross product of the set of prefixes for each field, resulting in a potentially large number of prefix rules.

2.1 Firewall decision diagrams

A crucial data structure required for this work is the Firewall Decision Diagram (FDD) [Gouda and Liu(2004)]. A Firewall Decision Diagram (FDD) with a decision set \( DS \) and over fields \( F_1, \ldots, F_d \) is an acyclic and directed graph that has the following five properties: (1) There is exactly one node that has no incoming edges. This node is called the root. The nodes that have no outgoing edges are called terminal nodes. (2) Each node \( v \) has a label, denoted \( F(v) \), such that
Input: An interval \( \text{Interval} = (a, b) \) and a prefix aligned interval \( \text{test} = (c, d) \) s.t. \( a, b, c, d \in \mathbb{N} \).

Output: A list of prefix aligned ranges.

1. Let \( i = (e, f) \) be the intersection of \( \text{Interval} \) and \( \text{test} \);
2. if \( i \) is empty then
   return an empty list;
3. else
   if \( i = \text{test} \) then
      return a list that contains only \( \text{test} \);
   else
      Split \( \text{test} \) into two prefix intervals \( \text{low} \) and \( \text{high} \);
      return the concatenation of \( \text{GetPrefixes(Interval,low)} \) and \( \text{GetPrefixes(Interval,high)} \);
   end
end

Algorithm 1: GetPrefixes(Interval, test = \((0, 2^{32} - 1)\))

\[
F(v) \in \begin{cases} 
\{F_1, \ldots, F_d\} & \text{if } v \text{ is a nonterminal node,} \\
\text{DS} & \text{if } v \text{ is a terminal node.}
\end{cases}
\]

(3) Each edge \( e:u \to v \) is labeled with a nonempty set of integers, denoted \( I(e) \), where \( I(e) \) is a subset of the domain of \( u \)'s label (i.e., \( I(e) \subseteq D(F(u)) \)).
(4) A directed path from the root to a terminal node is called a decision path. No two nodes on a decision path have the same label.
(5) The set of all outgoing edges of a node \( v \), denoted \( E(v) \), satisfies the following two conditions:
   (i) \( \text{Consistency: } I(e) \cap I(e') = \emptyset \) for any two distinct edges \( e \) and \( e' \) in \( E(v) \).
   (ii) \( \text{Completeness: } \bigcup_{e \in E(v)} I(e) = D(F(v)) \).

We define a full-length ordered FDD as an FDD where in each decision path all fields appear exactly once and in the same order. For ease of presentation, we use the term “FDD” to mean “full-length ordered FDD” if not otherwise specified. Given a packet classifier \( \mathbb{C} \), the FDD construction algorithm in [Liu and Gouda(2004)] can convert it to an equivalent full-length ordered FDD \( f \). Figure 2.1(a) contains a sample classifier, and Figure 2.1(b) shows the resultant FDD from the construction process shown by Algorithms 2 and 3.

Input: A packet classifier \( f : \langle r_1, r_2, \ldots, r_n \rangle \)

Output: A \( f \) for packet classifier \( f \)

1. Build a path from rule \( r_1 \). Let \( v \) denote the root. The label of the terminal node is \( \langle 1 \rangle \);
2. for \( i = \{2, \ldots, n\} \in C \) do
   APPEND( \( v, r_i, 1, i \) );
end

Algorithm 2: FDD Construction Algorithm

After an FDD \( f \) is constructed, we can reduce \( f \)'s size by merging isomorphic subgraphs. A full-length ordered FDD \( f \) is reduced if and only if it satisfies the
following two conditions: (1) no two nodes in $f$ are isomorphic; (2) no two nodes have more than one edge between them. Two nodes $v$ and $v'$ in an FDD are isomorphic if and only if $v$ and $v'$ satisfy one of the following two conditions: (1) both $v$ and $v'$ are terminal nodes with identical labels; (2) both $v$ and $v'$ are nonterminal nodes and there is a one-to-one correspondence between the outgoing edges of $v$ and the outgoing edges of $v'$ such that every pair of corresponding edges have identical labels and they both point to the same node. A reduced FDD is essentially a canoni-
2.2 One-Dimensional Classifier Minimization

The special problem of weighted one-field TCAM minimization is used as a building block for multi-dimensional TCAM minimization. Given a one-field packet classifier \( f \) of \( n \) prefix rules \( \langle r_1, r_2, \ldots, r_n \rangle \), where \( \{ \text{Decision}(r_1), \text{Decision}(r_2), \ldots, \text{Decision}(r_n) \} = \{ d_1, d_2, \ldots, d_z \} \) and each decision \( d_i \) is associated with a cost \( \text{Cost}(d_i) \) (for \( 1 \leq i \leq z \)), we define the cost of packet classifier \( f \) as follows:

\[
\text{Cost}(f) = \sum_{i=1}^{n} \text{Cost}(\text{Decision}(r_i))
\]

Based upon the above definition, the problem of weighted one-dimensional TCAM minimization is stated as follows.

**Definition 2.2.** Weighted One-Dimensional Prefix Minimization Problem Given a one-field packet classifier \( f \) where each decision is associated with a cost, find a...
prefix packet classifier \( f_2 \in \{f_1\} \) such that for any prefix packet classifier \( f \in \{f_1\} \), the condition \( \text{Cost}(f_2) \leq \text{Cost}(f) \) holds.

The problem of one-dimensional prefix minimization (with uniform cost) has been studied in [Draves et al(1999)Draves, King, Venkatachary, and Zill, Suri et al(2003)Suri, Sandholm, and Warkhede] in the context of compressing routing tables. I generalize the dynamic programming solution in [Suri et al(2003)Suri, Sandholm, and Warkhede] to solve the weighted one-dimensional TCAM minimization. There are three key observations:

1. For any one-dimensional packet classifier \( f \) on \( \{\ast\}^w \), we can always change the predicate of the last rule to be \( \{\ast\}^w \) without changing the semantics of the packet classifier. This follows from the completeness property of packet classifiers.

2. Consider any one-dimensional packet classifier \( f \) on \( \{\ast\}^w \). Let \( f' \) be \( f \) appended with rule \( \{\ast\}^w \rightarrow d \), where \( d \) can be any decision. The observation is that \( f \equiv f' \).

This is because the new rule is redundant in \( f' \) since \( f \) must be complete. A rule in a packet classifier is redundant if and only if removing the rule from the packet classifier does not change the semantics of the packet classifier.

3. For any prefix \( P \in \{0,1\}^k\{\ast\}^{w-k} \) (0 \( \leq k \leq w \)), one and only one of the following conditions holds:

   a. \( P \in \{0,1\}^k0\{\ast\}^{w-k-1} \)

   b. \( P \in \{0,1\}^k1\{\ast\}^{w-k-1} \)

   c. \( P = \{0,1\}^k\{\ast\}^{w-k} \)

This property allows us to divide a problem of \( \{0,1\}^k\{\ast\}^{w-k} \) into two sub-problems: \( \{0,1\}^k0\{\ast\}^{w-k-1} \), and \( \{0,1\}^k1\{\ast\}^{w-k-1} \). This divide-and-conquer strategy can be applied recursively.

We formulate an optimal dynamic programming solution to the weighted one-dimensional TCAM minimization problem.

Let \( P \) denote a prefix \( \{0,1\}^k\{\ast\}^{w-k} \). We use \( P \) to denote the prefix \( \{0,1\}^k0\{\ast\}^{w-k-1} \), and \( \overline{P} \) to denote the prefix \( \{0,1\}^k1\{\ast\}^{w-k-1} \).

Given a one-dimensional packet classifier \( f \) on \( \{\ast\}^w \), we use \( f_P \) to denote a packet classifier on \( P \) such that for any \( x \in P \), \( f_P(x) = f(x) \), and we use \( f_{\overline{P}} \) to denote a similar packet classifier on \( \overline{P} \) with the additional restriction that the final decision is \( d \).

\( C(f_P) \) denotes the minimum cost of a packet classifier \( t \) that is equivalent to \( f_P \), and \( C(f_{\overline{P}}) \) denotes the minimum cost of a packet classifier \( t' \) that is equivalent to \( f_{\overline{P}} \) and the decision of the last rule in \( t' \) is \( d \).

Given a one-dimensional packet classifier \( f \) on \( \{\ast\}^w \) and a prefix \( P \) where \( P \subseteq \{\ast\}^w \), \( f \) is consistent on \( P \) if and only if \( \forall x, y \in P \), \( f(x) = f(y) \).

The dynamic programming solution to the weighted one-dimensional TCAM minimization problem is based on the following theorem. The proof of the theorem shows how to divide a problem into sub-problems and how to combine solutions to sub-problems into a solution to the original problem.
Theorem 2.1. Given a one-dimensional packet classifier \( f \) on \( \{\ast\}^w \), a prefix \( \mathcal{P} \) where \( \mathcal{P} \subseteq \{\ast\}^w \), the set of all possible decisions \( \{d_1, d_2, \ldots, d_z\} \) where each decision \( d_i \) has a cost \( w_{d_i} \) \((1 \leq i \leq z)\), we have that

\[
C(f_{\mathcal{P}}) = \min_{i=1}^{z} C(f_{d_i})
\]

where each \( C(f_{d_i}) \) is calculated as follows:

1. If \( f \) is consistent on \( \mathcal{P} \), then

\[
C(f_{d_i}) = \begin{cases} 
    w_{f(x)} & \text{if } f(x) = d_i \\
    w_{f(x)} + w_{d_i} & \text{if } f(x) \neq d_i
\end{cases}
\]

2. If \( f \) is not consistent on \( \mathcal{P} \), then

\[
C(f_{d_i}) = \min \left\{ \begin{array}{l}
    C(f_{d_i}) + C(f_{d_{i+1}}) - w_{d_i} + w_{d_i}, \\
    \vdots \\
    C(f_{d_{i-1}}) + C(f_{d_{i}}) - w_{d_{i-1}} + w_{d_i}, \\
    C(f_{d_{i}}) + C(f_{d_{i+1}}) - w_{d_{i}}, \\
    \vdots \\
    C(f_{d_{z}}) + C(f_{d_{1}}) - w_{d_z} + w_{d_1}
\end{array} \right\}
\]

Proof. (1) The base case is when \( f \) is consistent on \( \mathcal{P} \). In this case, the minimum cost prefix packet classifier in \( \{f_{\mathcal{P}}\} \) is clearly \( (\mathcal{P} \rightarrow f(x)) \), and the cost of this packet classifier is \( w_{f(x)} \). Furthermore, for \( d_i \neq f(x) \), the minimum cost prefix packet classifier in \( \{f_{\mathcal{P}}\} \) with decision \( d_i \) in the last rule is \( (\mathcal{P} \rightarrow f(x), \mathcal{P} \rightarrow d_i) \) where the second rule is redundant. The cost of this packet classifier is \( w_{f(x)} + w_{d_i} \).

(2) If \( f \) is not consistent on \( \mathcal{P} \), divide \( \mathcal{P} \) into \( \mathcal{P} \) and \( \overline{\mathcal{P}} \). The crucial observation is that an optimal solution \( f^* \) to \( \{f_{\mathcal{P}}\} \) is essentially an optimal solution \( f_1 \) to the sub-problem of minimizing \( f_{\overline{\mathcal{P}}} \) appended with an optimal solution \( f_2 \) to the sub-problem of minimizing \( f_{\mathcal{P}} \). The only interaction that can occur between \( f_1 \) and \( f_2 \) is if their final rules have the same decision, in which case both final rules can be replaced with one final rule covering all of \( \mathcal{P} \) with the same decision. Let \( d_x \) be the decision of the last rule in \( f_1 \) and \( d_y \) be the decision of the last rule in \( f_2 \). Then we can compose \( f^* \) whose last rule has decision \( d_i \) from \( f_1 \) and \( f_2 \) based on the following cases:

(A) \( d_x = d_y = d_i \): In this case, \( f \) can be constructed by listing all the rules in \( f_1 \) except the last rule, followed by all the rules in \( f_2 \) except the last rule, and then the last rule \( \mathcal{P} \rightarrow d_i \). Thus, \( Cost(f) = Cost(f_1) + Cost(f_2) - w_{d_i} \).

(B) \( d_x = d_y \neq d_i \): In this case, \( f \) can be constructed by listing all the rules in \( f_1 \) except the last rule, followed by all the rules in \( f_2 \) except the last rule, then rule \( \mathcal{P} \rightarrow d_x \), and finally rule \( \mathcal{P} \rightarrow d_i \). Thus, \( Cost(f) = Cost(f_1) + Cost(f_2) - w_{d_x} + w_{d_i} \).

(C) \( d_x \neq d_y, d_x = d_i, d_y \neq d_i \): We do not need to consider this case because \( C(f_{d_i}) + C(f_{\overline{\mathcal{P}}}) = C(f_{d_i}) + (C(f_{\overline{\mathcal{P}}}) + w_{d_i}) - w_{d_i} = C(f_{d_i}) + C(f_{\overline{\mathcal{P}}}) - w_{d_i} \).

(D) \( d_x \neq d_y, d_x \neq d_i, d_y = d_i \): Similarly, this case need not be considered.

(E) \( d_x \neq d_y, d_x \neq d_i, d_y \neq d_i \): Similarly, this case need not be considered.
Figure 2.2 shows the illustration of a one-dimensional TCAM minimization problem, where the black bar denotes decision “accept” and the white bar denotes decision “discard”. Figure 2.3 illustrates how the dynamic programming works on this example. Algorithm 4 shows the pseudocode for finding $C(f^d_{ij})$.

**Fig. 2.2** An example one-dimensional TCAM minimization problem

**Fig. 2.3** Illustration of dynamic program
2.2 One-Dimensional Classifier Minimization

**Input:** A *Universe* they contains the color information of each prefix in the domain, a prefix interval \( p \), and dictionary of weights for each color.

**Output:** A dictionary that contains the cost of the optimal prefix solution for keys

\[(\text{Prefix}, \text{Color}) \text{ where the prefix Prefix has the background color Color.}\]

Let \( c \) be the colors at prefix \( p \) on *Universe*;

Let \( \text{Colors} \) be the set of colors defined by ColorWeights;

if \( c \) is monochromatic then

Let \( \text{answer} \) be an empty dictionary;

for each color \( \in \text{Colors} \) do

if color \( \neq c \) then

\[\text{answer}[(p, \text{color})] = \text{ColorWeights}[c] + \text{ColorWeights}[\text{color}]\]

else

\[\text{answer}[(p, \text{color})] = \text{ColorWeights}[c]\]

end

end

else

Split \( p \) into low and high;

Let \( \text{answer} \) be a new dictionary that contains the keys from

ODPM(*Universe*, low, ColorWeights) and ODPM(*Universe*, low, ColorWeights);

for each color that is defined by ColorWeights do

\[\text{answer}[(p, \text{color})] = \min_{cc \in \text{Colors}} \begin{cases} 
\text{answer}[(\text{lowPrefix}, cc)] + \text{answer}[(\text{highPrefix}, cc)] - \text{ColorWeights}[cc] & \text{if color} = cc \\
\text{answer}[(\text{lowPrefix}, cc)] + \text{answer}[(\text{highPrefix}, cc)] - \text{ColorWeights}[cc] + \text{ColorWeights}[\text{color}] & \text{otherwise}
\end{cases}\]

end

return \( \text{answer} \);

**Algorithm 4:** ODPM(*Universe*, \( p \), ColorWeights)
Hardware Based Packet Classification for High Speed Internet Routers
Meiners, C.R.; Liu, A.X.; Tornq, E.
2010, XIII, 123 p., Hardcover