Packet Classification using Tuple Space Search

Subhash Suri

Computer Science
Washington Univ., St. Louis

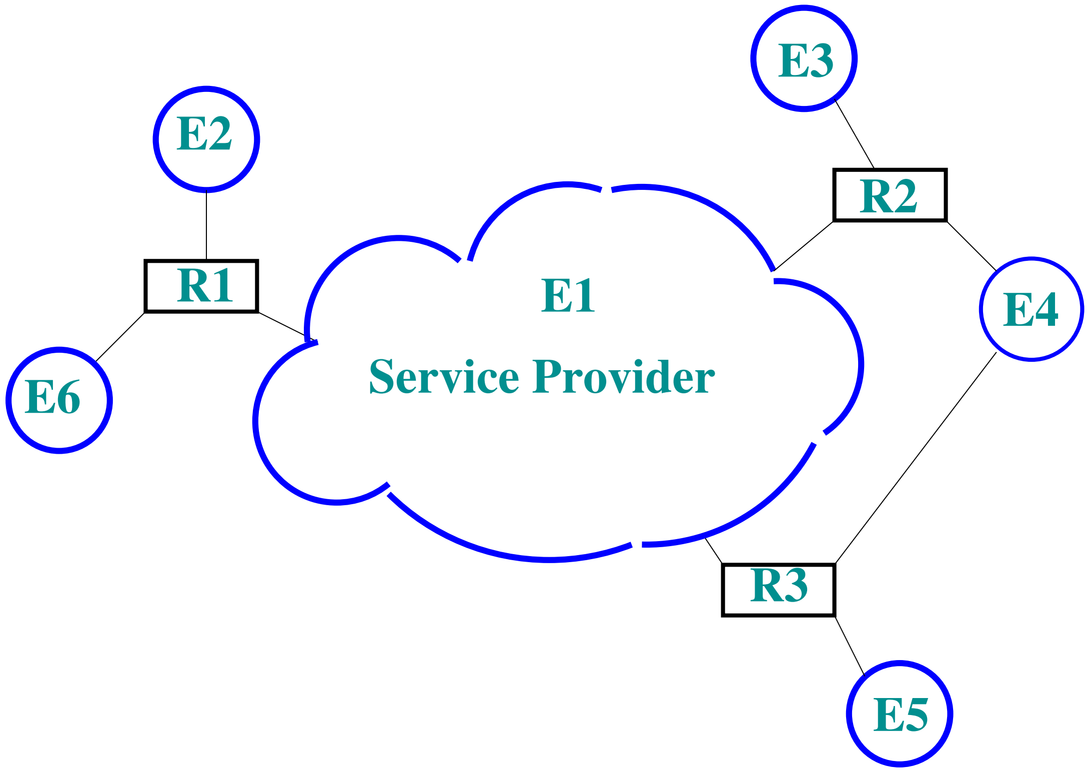
(with V. Srinivasan and G. Varghese)

Sept. 2 1999

ACM SIGCOMM '99

Packet Classification

- **Packet classification useful for:**
  1. **Access control**
  2. **Policy-based routing**
  3. **Virtual private networks**
  4. **Monitoring and billing**



Examples

- $(D_1, \ S_1) \quad \rightarrow \quad$ **"Reserve 50 Mbps".**
- $(D_2, \ S_2, \ \textbf{video}) \quad \rightarrow \quad$ **"Use Link L".**
- $(D_3, \ *, \ 23, \ *, \ \textbf{tcp}) \quad \rightarrow \quad$ **"Drop".**

Problem Formulation

Input: N **Filters, each with cost.**

Query: Cheapest filter matching H**.**

$$
\begin{aligned}
F_1 &= [00*,\ 10*,\ \mathbf{80},\ *,\ \mathbf{tcp}] \\
F_2 &= [0*,\ 1*,\ < \mathbf{1024},\ *,\ *] \\
F_3 &= [0*,\ *,\ *,\ *,\ \mathbf{tcp}]
\end{aligned}
$$

- **BMF for** $[000,\ 100,\ 80, 20, \mathbf{tcp}]$ **is** F_1**.**
- **BMF for** $[010,\ 100,\ 6000, 20, \mathbf{tcp}]$ **is** F_3**.**

Metrics: Lookup and update time in terms of number of filters, N**, and number of fields,** K**.**

Previous Results

SCHEME	GOOD	BAD
Grid-of-Tries (SVSW '98)	2 Dim	W^{K-1} Search
Crossproducting (SVSW '98)	Fast	N^K memory
Bit Vector (LS '98)	Fast	Hardware N^2 memory
Quadtree (BSW '99)	Update	W^{K-1} Search
Ternary CAMs	Fast	Cost, size

Contributions

1. **Tuple Space Search**
2. **Pruned Tuple Space**
3. **Simulation Results**
4. **Rectangle Search**
5. **Lower Bounds for tuple space**
6. **Conclusions**

Tuple Space

IP Address Lookup

Prefix	Link
01*	L1
0110*	L2
0011*	L3
10*	L2
0101*	L3

Hash Table	Prefix Set
T2	01*, 10*
T4	0110*, 0011*, 0101*

- **BMP for** 0111 **is** $01*$**.**

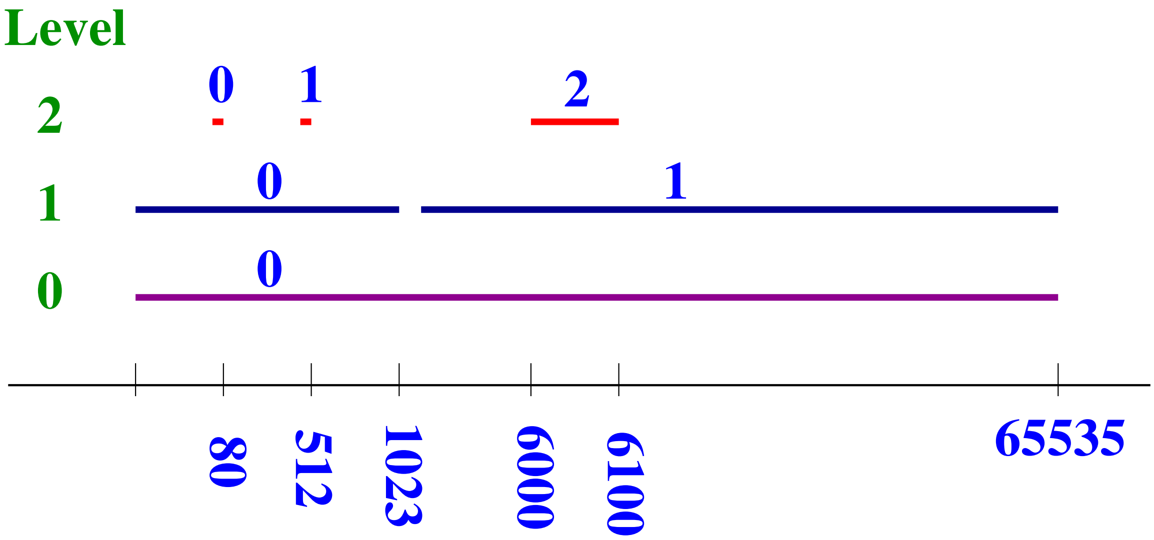
Tuple Space for Filters

Filters	Tuple
(10*, 1100*, TCP) (01*, 0110*, UDP)	[2, 4, 8]
(10011*, *, *)	[5, 0, 0]

- **Tuple is a vector $[\ell_1, \ldots, \ell_K]$.**
- **Hash Key for $[2, 4, 8]$.**

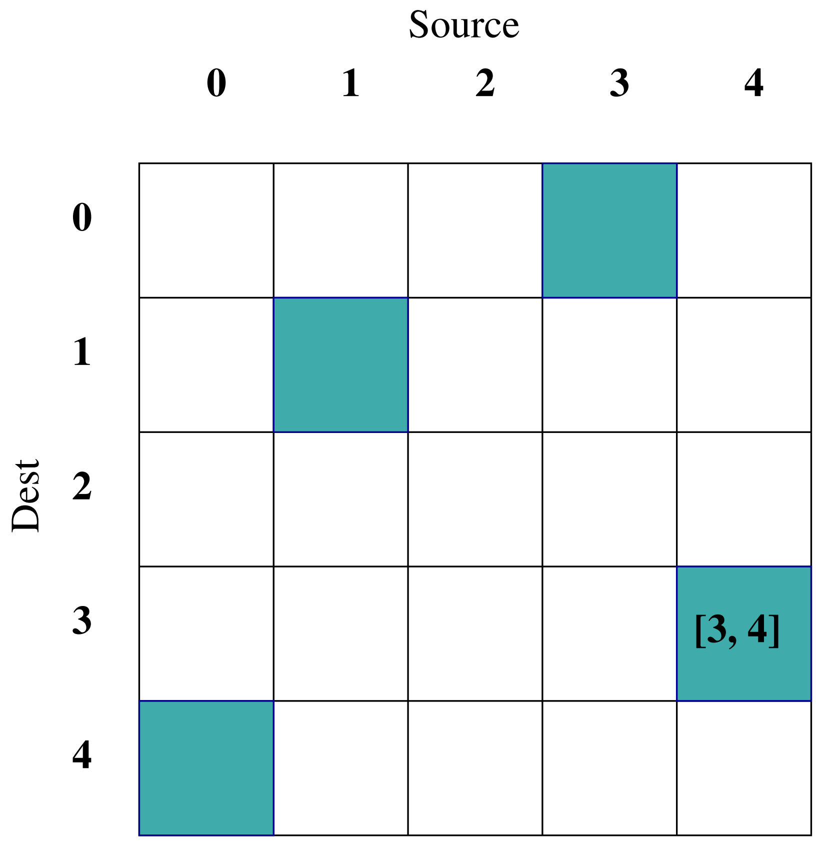
Port Ranges

- $\{*,$ **80, 512,** $\leq$ **1023,** $\geq$ **1024, 6000-6100** $\}$.



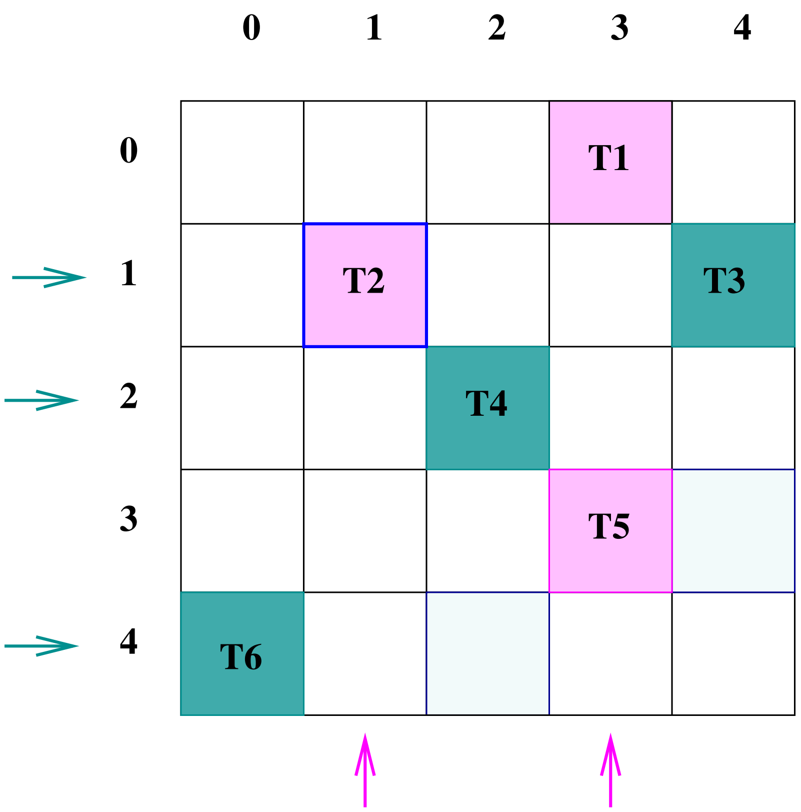
- **Port** 512 **maps to** $(2, 1)$.
- $[01*, 1100*, 512, *, \textbf{tcp}] \quad \rightarrow \quad [2, 4, 2, 0, 8]$.
- **Hash key is 01 1100 1 0 tcp**

Tuple Space Search



- $T[3,4] = \{(001, 1010), (101, 0010), \ldots\}$.
- **One hash per tuple.**
- **Few prefix lengths $\Longrightarrow$ Few Tuples.**

Tuple Pruning



T1 = (*, 000*)

T2 = (1*, 0*)

T3 = (1*, 0101*)

T4 = (11*, 01*)

T5 = (101*, 000*)

T6 = (1111*, *)

Header (1111, 0000)

List (D 1111) = (T2, T3, T4, T6)

List (S 0000) = (T1, T2, T5)

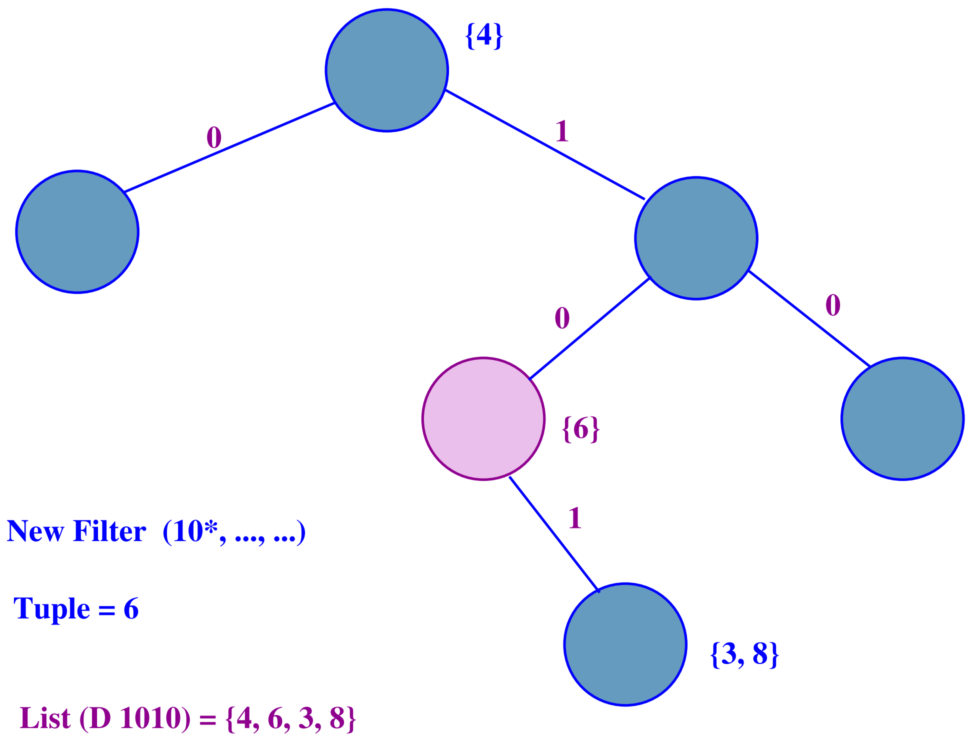
Intersected List = (T2)

Pruning on Multiple Fields

- **Header** $(DA,\ SA,\ DP,\ SP,\ R)$.
- **Determine BMP for header fields.**
- **Prune list is**

$$List(DA) \cap List(SA) \cdots \cap List(R)$$

Building Prune Lists



- **Build Prune List during BMP.**
- **Tuple** 6 **is in** $List(10)$ **only.**
- $List(1010) = \{4, 6, 3, 8\}$.

Simulation Results

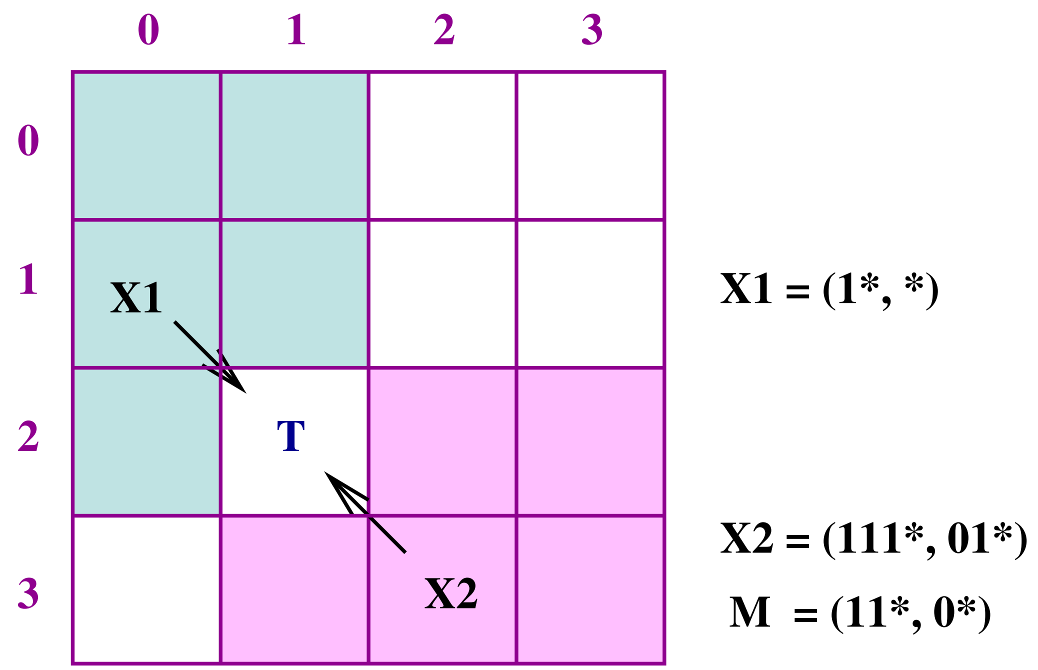
- Worst-case hashes by Pruned Search.

Database	Filters	Tuples	Pruned
Fwal-1	278	41	11
Fwal-2	158	28	6
Fwal-3	183	24	7
Fwal-4	68	15	5

- Random D-S pair filters.

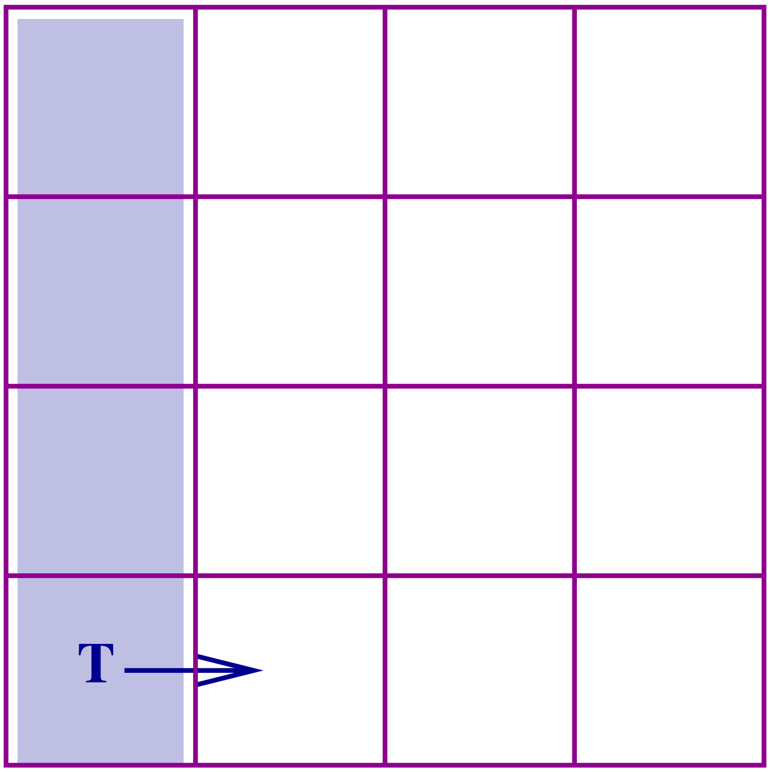
Size	Tuples	Dst-Prune	Src-Prune	Both-Prune
1K	53	1	1	1
5K	92	2	2	2
10K	104	3	3	2
50K	151	4	6	3
100K	186	6	8	4

Rectangle Search



- T **has a filter** $F = (10*, 1*)$.
- F **precomputes BMF from top-left.**
- **Filters in bot-right leave markers.**

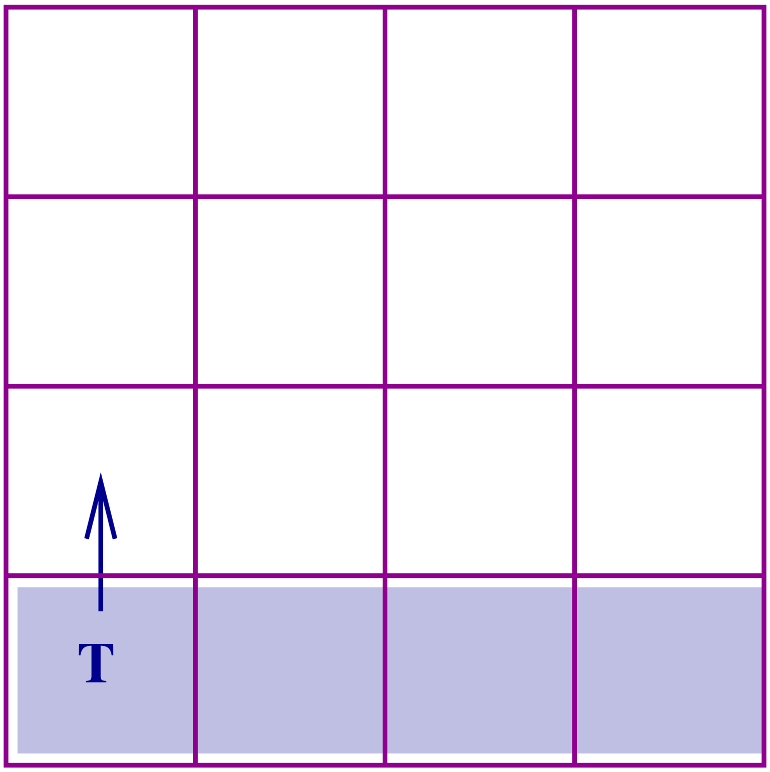
Rectangle Search: Match



F = (101, 010)

- **Probe with** $(101, *)$.
- **On match, skip column.**

No Match

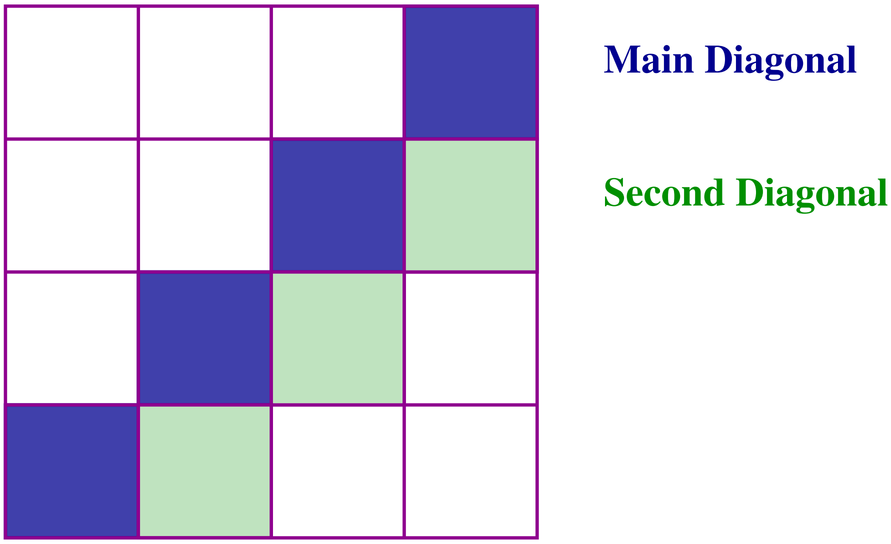


F = (101, 010)

- **Probe with** $(101, *)$.
- **On no match, skip row.**

Lower Bound

- **Rectangle Search makes $2W - 1$ probes in worst-case.**
- **Lower Bound proof shows this is the best possible in hash-based schemes.**



- **In worst-case, cells in main and second diagonals must be probed.**

Conclusions

- **Tuple space improves search.**
- **Pruned Tuple Search**
  1. **Has linear memory.**
  2. **Allows fast updates.**
  3. **Should be scalable.**
- **Rectangle Search worst-case optimal for 2D filters.**
- **Lower Bounds.**