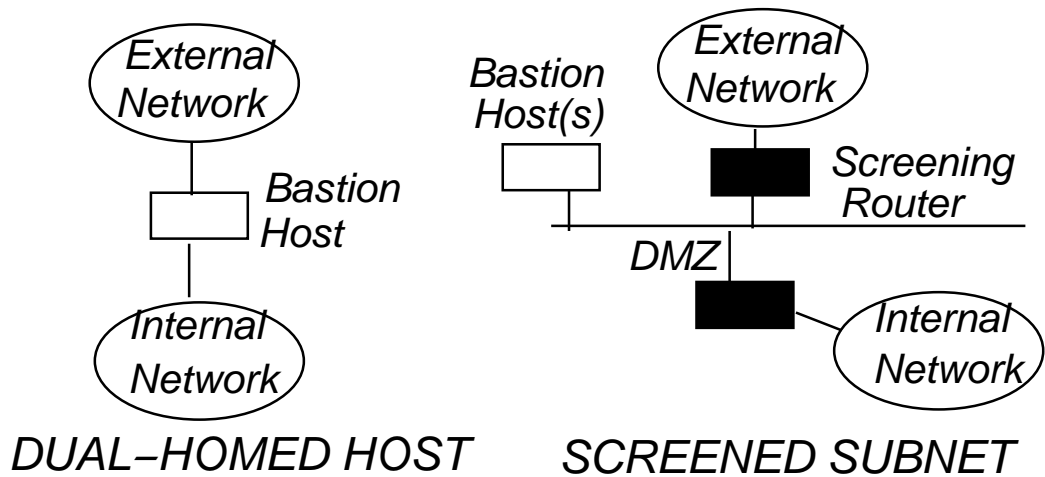


# Fast Filter Lookups

George Varghese

June 13, 2000

# Firewall Configurations



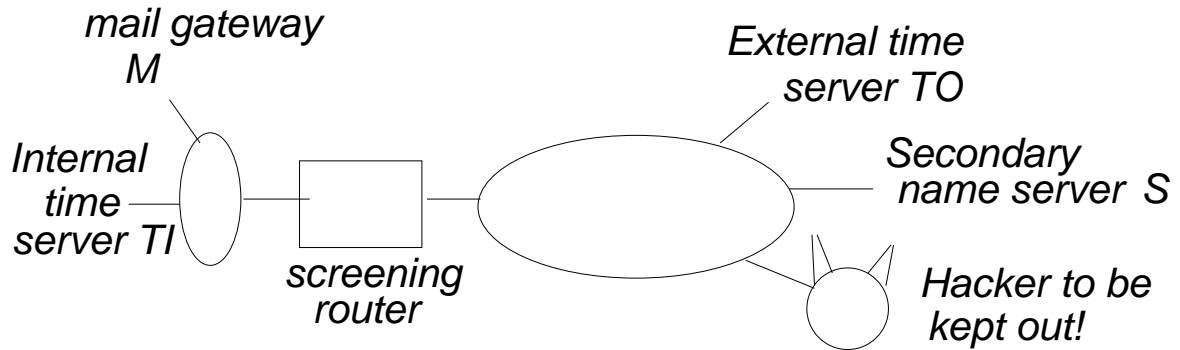
- Firewalls use *packet filtering* (Telnet, SMTP) and *application level gateways* (Web, FTP).
- Proxies are specialized for each service, require modifications to clients, and need screening routers anyway. Need to improve screening routers.

## Internet Message Format

<i>Destination</i> (32)	<i>Source</i> (32)	<i>Protocol</i> (8)	<i>Dest Port</i> (16)	<i>Source Port</i> (16)
----------------------------	-----------------------	------------------------	--------------------------	----------------------------

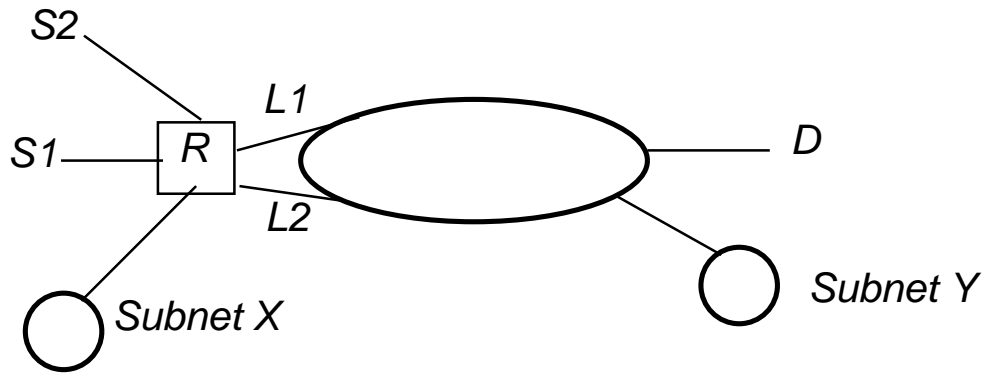
- Destination and Source address correspond to sending host and receiving computers (like telephone numbers)
- Protocol field most commonly indicates TCP (like certified mail with acks) or UDP (like regular U.S. mail).
- Destination Port is like destination extension. Some well known extensions (e.g., like 12 is extension of marketing department)

# Sample Firewall Database



<i>Destination</i>	<i>Source</i>	<i>Destination Port</i>	<i>Source Port</i>	<i>flags</i>	<i>comments</i>
M	*	25	*	*	allow inbound mail
M	*	53	*	UDP	allow DNS access
M	S	53	*	*	secondary access
M	*	23	*	*	incoming telnet
TI	TO	123	123	UDP	NTP time info
*	Net	*	*	*	outgoing packets
Net	*	*	*	TCP ack	return ACKs OK
*	*	*	*	*	block everything!

## Not just firewalls



**Database at Router R**

To	From	TrafficType	Forwarding Directive
D	S1	Video	Forward via L1
*	S2	*	Drop all traffic
Y	X	*	Reserve 50 Mbps

- Trend in RSVP (resource allocation) and QoS Routing.

## Simplest Solution

- The simplest solution is based on *ternary CAMs* which allow matching with don't care bits. Ternary CAMs that handle 32,000 words (with don't cares) of 32 bits exist and ones with 64,000 are being announced.
- However, that requires another chip on board. Can also use other hardware solutions such as those due to SwitchOn networks. ([www.switchon-net.com](http://www.switchon-net.com)) that allows up to 16K rules on Source, Destination, Protocol, DestPort and Source Port.

## Plan of Attack

Let us first consider packet classification on two fields (say source and destination) to build intuition. Useful for traffic measurement of traffic between networks and for multicast forwarding. So our plan is:

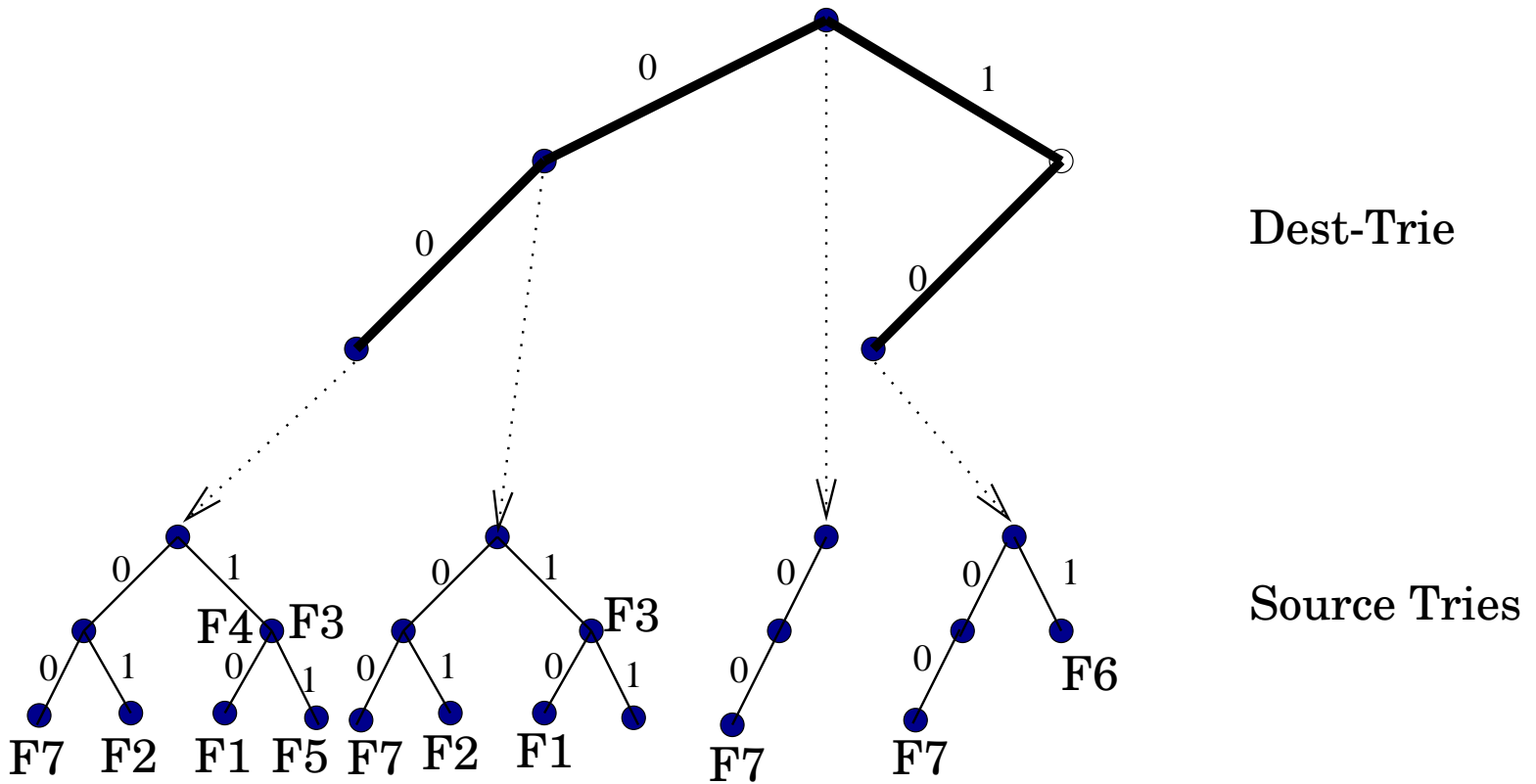
- **1.0 Two dimensional Filters:** Set pruning tries, backtracking tries, grid-of-tries, and rectangle search
- **2.0 General Filters:** Extended grid-of-tries, and schemes based on divide and conquer.

## 1.0 Two Dimensional Filters

## 2D Example

Filter	Destination	Source
$F_1$	0*	10*
$F_2$	0*	01*
$F_3$	0*	1*
$F_4$	00*	1*
$F_5$	00*	11*
$F_6$	10*	1*
$F_7$	*	00*

## The First Idea: Set Pruning Tries

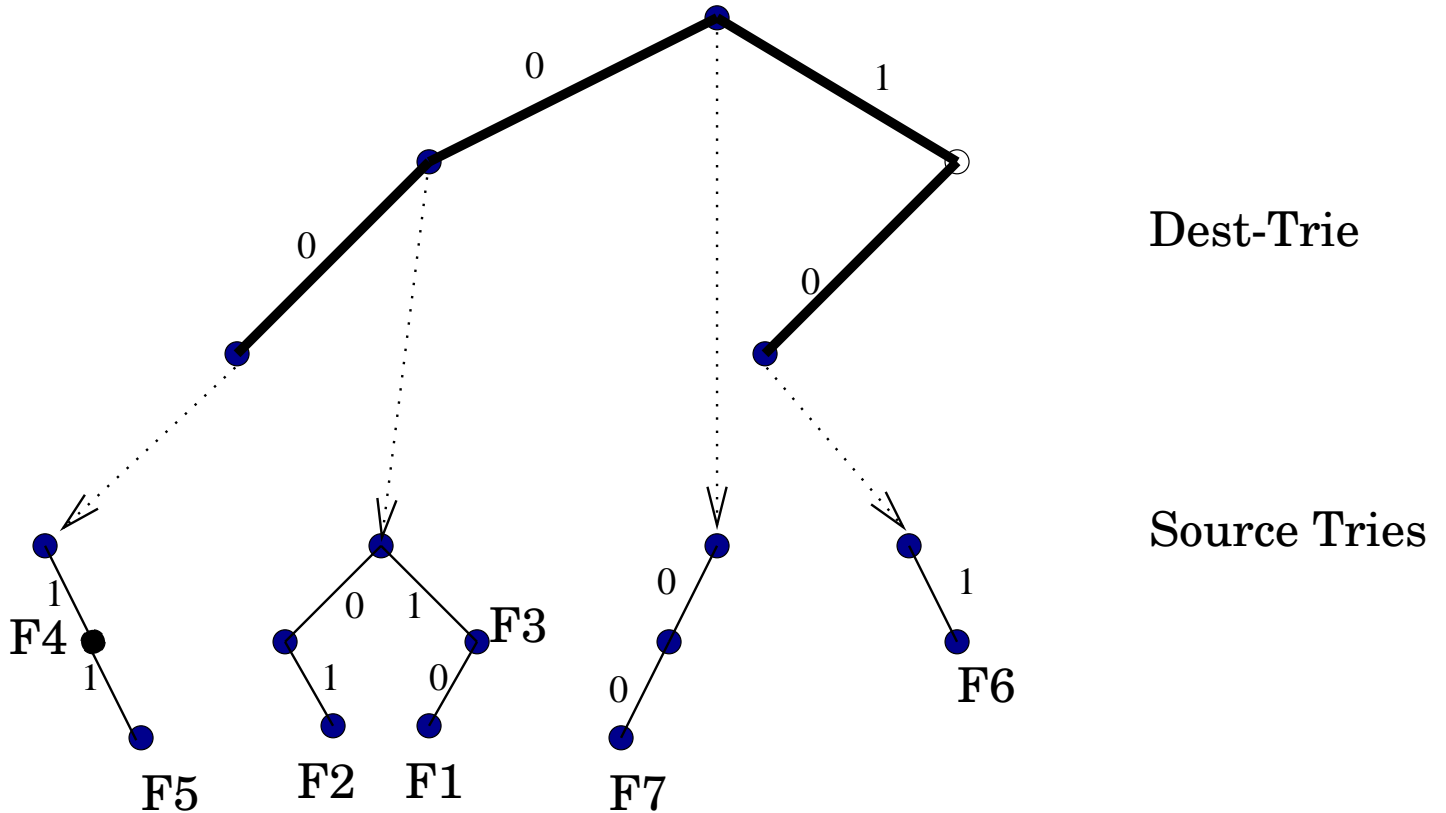


- Each valid destination prefix  $D$  has a pointer to a source trie containing the source prefixes that belong to filters whose destination field is a *prefix* of  $D$ .
- Can take  $\Theta(N^2)$  memory for  $N$  filters in the worst case.

## Worst-case example for large storage

Filter	Destination	Source
$F_1$	$D_1$	*
$F_2$	$D_2$	*
	$\vdots$	
$F_{N/2}$	$D_{N/2}$	*
$F_{N/2+1}$	*	$S_1$
$F_{N/2+2}$	*	$S_2$
	$\vdots$	
$F_N$	*	$S_N$

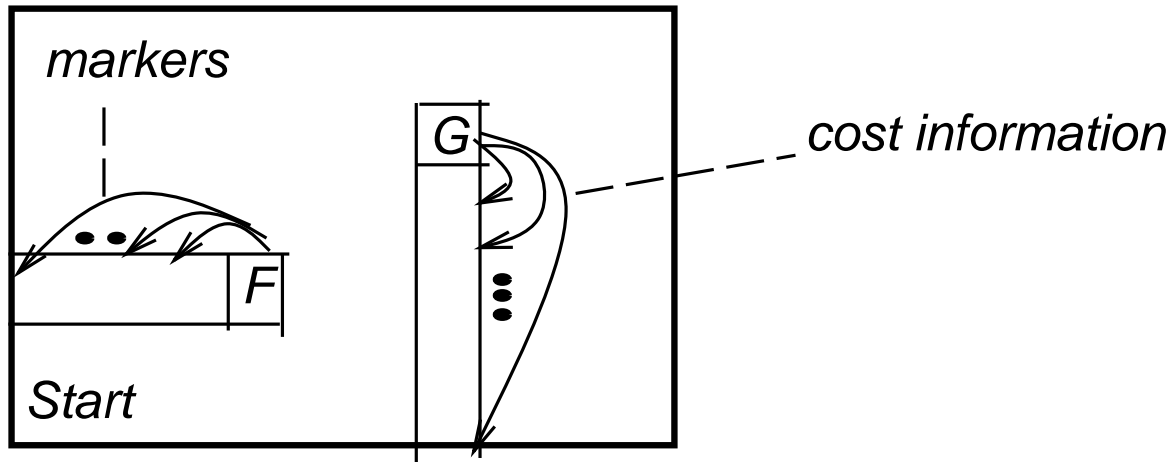
## Backtracking Search



- Each valid destination prefix  $D$  has a pointer to a source trie containing the source prefixes that belong to filters whose destination field is *equal* to  $D$ .
- Now need to backtrack when searching and can take  $W^2$  time (900 memory accesses). Works quite well in practice.

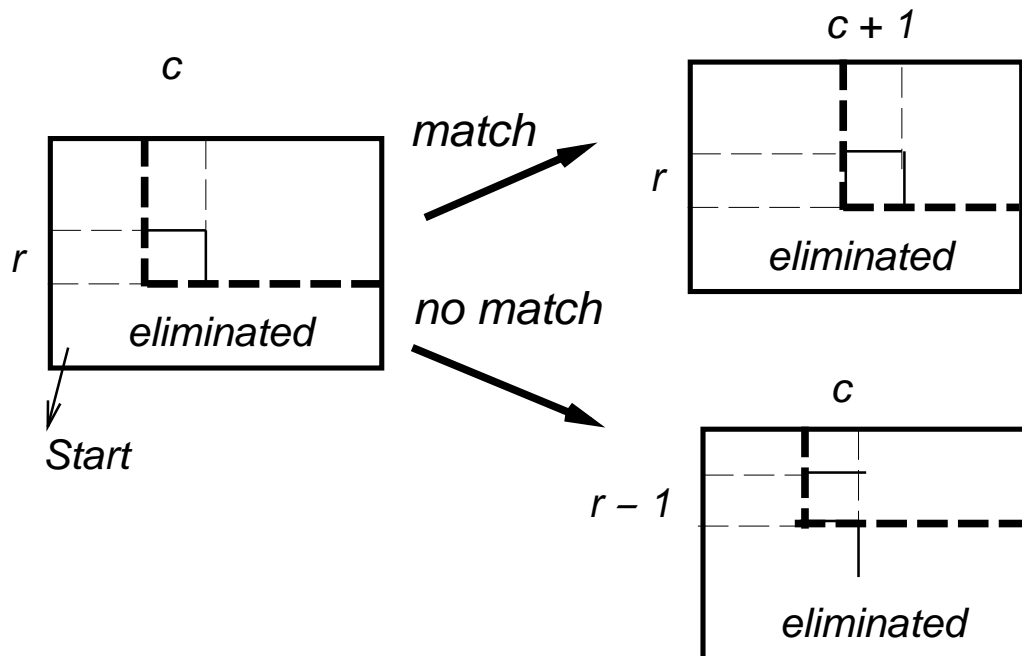


## Using Hashing: Rectangle Search



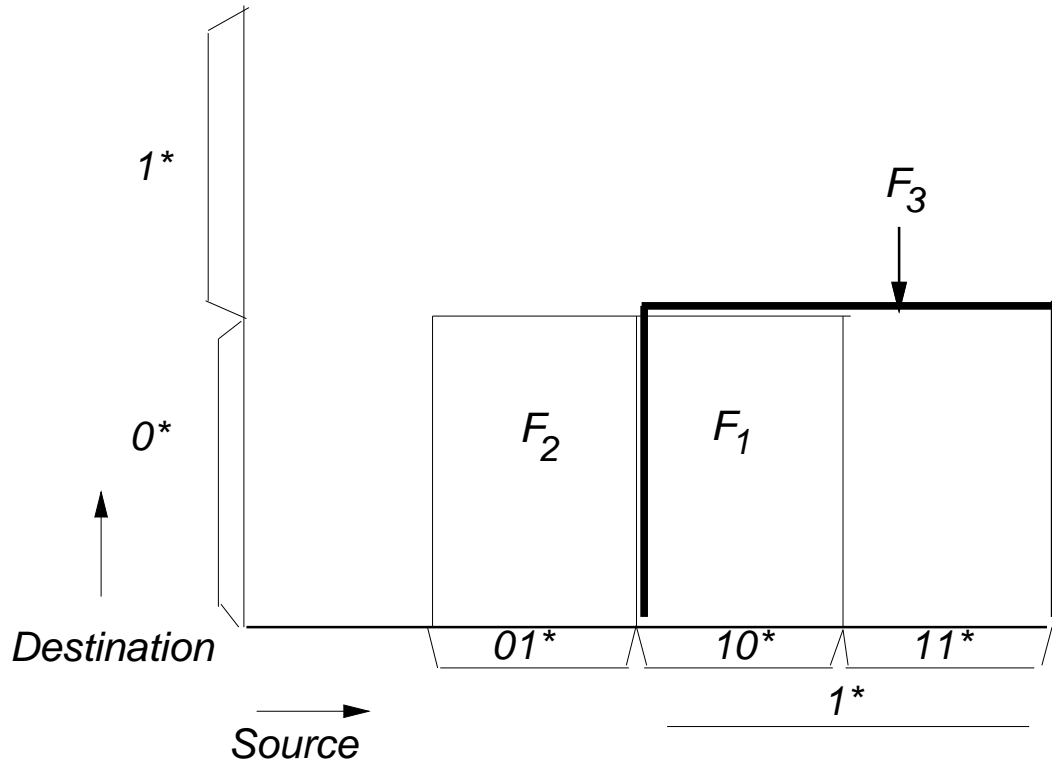
- Generalization of binary search on prefix lengths. Multiple hashing using markers and precomputation to reduce hashes.

## Rectangle Search Proof



- Each probe eliminates a row or a column. Since we have only  $2W$  rows and columns takes only  $2W$  time.

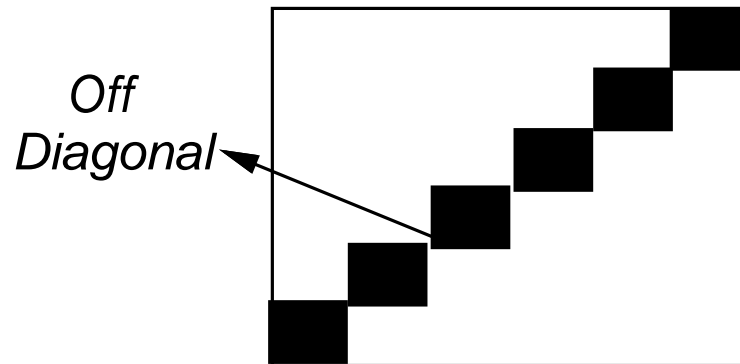
## Geometric View of Filters



- For example,  $F_1 = 0^*, 10^*$  is the box whose projection on the Destination axis is the range corresponding to  $0^*$  and whose projection on the Source axis is the range corresponding to  $10^*$ .
- Used to adapt algorithms for 2D point locations from computational geometry by Lucent. Poor performance.

## **2.0 General (at least 5) Dimensional Filters**

## Beyond 2D: The Bad News

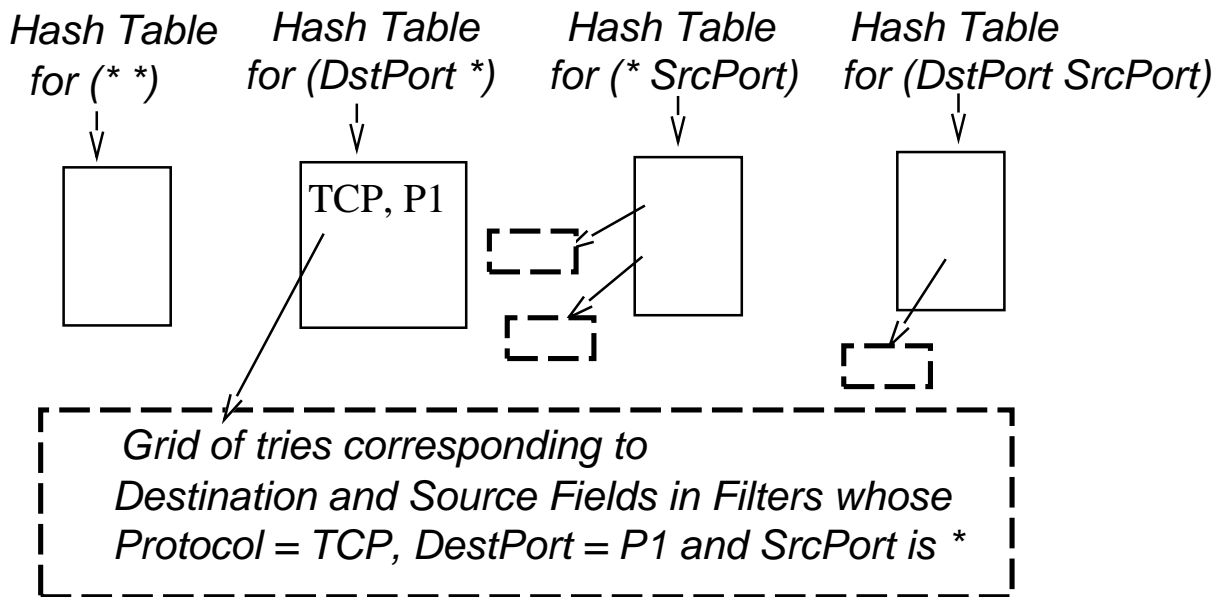


- At least  $2W$  hashes required for 2D filters.
- At least  $\frac{W^{(k-1)}}{k!}$  hashes required for  $k$  dimensions using linear memory equal to 17462 hashes!

## Beyond 2D: The Good News

- **O1, Prefix containment is rare:** It is fairly rare to have prefixes that are prefixes of other prefixes such as  $00^*$  and  $0001^*$ . Maximum prefixes = 7.
- **O2, Many fields are not general ranges:** Most ranges are ( $\geq 1024$  and  $< 1024$ ).
- **O3, Number of disjoint classification regions is small:** Number of distinct regions in geometric view is small in real databases and much less than  $N^{k-1}$  which is worst case.

## Idea 1: Extending Grid-of-tries



- Can do it using 4 grid-of-tries lookups and 4 hashes assuming only wildcard port ranges.
- Allowing  $> 1024$  and  $< 1024$  we can do it with 10 hash tables but only 5 grid-of-tries searches and 5 hashes. Common case.

## Algorithmic Approach: Divide and Conquer

$$4 * 4 * 5 * 2 * 3 = 480$$

<i>Destination Prefixes</i>	<i>Source Prefixes</i>	<i>DstPort Prefixes</i>	<i>SrcPort Prefixes</i>	<i>Flags Prefixes</i>
M	S	25	123	UDP
T1	TO	53	Default	TCP-ACK
Net	Net	23		Default
Default	Default	123		
		Default		

- Natural to divide into simpler problems (find best matching prefix for each field) and then combine (as in say merge sort).
- How to combine? Best in each field concatenated does not produce best filter.

## Range to Prefix Matches

- Can rewrite an arbitrary range as the union of logarithmic *prefix ranges*. range  $[8, 12]$  can be expressed as the union of the two prefix ranges  $10^*$  and  $1100^*$ .
- Write range as at most two anchored ranges and keep halving range. Ranges in  $[0, 2^k]$  need at most  $2k$  prefix ranges.
- For 16 bit port numbers, the range  $\leq 1023$  can be expressed using the prefix range  $000000^*$ . The range  $> 1023$  on the other hand can be expressed using 5 prefix ranges  $00001^*$  (1K - 2K),  $0001^*$  (2K-4K),  $001^*$  (4K-8K),  $01^*$  (8k-16K), and  $1^*$  (16K - 64K).

## Lucent Scheme

Destination Prefixes	Source Prefixes	DstPort Prefixes	SrcPort Prefixes	Flags Prefixes
M   11110111	S   11110011	25   10000111	123   11111111	UDP   11111101
T1   00001111	TO   11011011	53   01100111	*   11110111	TCP   10110111
Net   00000111	Net   11010111	23   00010111		*   10110101
*   00000101	*   11010011	123   00001111		
		*   00000111		

- Store an  $N$  bit vector with each field value  $M$ .
- Bit  $j$  is set for value  $M$  in field  $i$  if value  $M$  matches Filter  $j$  in field  $i$ .
- Have to AND 5  $N$  bit vectors and find first bit set.

## Pruned Tuple Search

<i>Destination Prefixes</i>		<i>Source Prefixes</i>		<i>DstPort Prefixes</i>		<i>SrcPort Prefixes</i>		<i>Flags Prefixes</i>	
M	1110111	S	1110011	25	1000111	123	1111111	UDP	1111101
T1	0001111	TO	1101011	53	0110111	*	1110111	TCP	1010111
Net	0000111	Net	1100111	23	1000111			*	1010101
*	0000101	*	1100011	123	0001111				
				*	0000111				

<i>BIT</i>	<i>TUPLE</i>
7 (MSB)	(32, 0, 16, 0, 0)
6	(32, 0, 16, 0, 8)
5	(32, 32, 16, 0, 8)
4	(32, 32, 16, 16, 8)
3	(0, 24, 0, 0, 0)
2	(24, 0, 0, 0, 0)
1	(0, 0, 0, 0, 0)

- Hash only on tuples in intersection

## CrossProducts

<i>Num</i>	<i>CrossProduct</i>	<i>Matching Rule</i>
1	M, S, 25, 123, UDP	Rule 1
2	M, S, 25, 123, TCP-ACK	Rule 1
3	M, S, 25, 123, default	Rule 1
4	M, S, 25, default, UDP	Rule 1
5	M, S, 25, default, TCP-ACK	Rule 1
6	M, S, 25, default, default	Rule 1
•	•            •            •	•
•	•            •            •	•
479	default,default,default,default,TCP-ACK	Rule 8
480	default,default,default,default,default	Rule 8

- (Theorem): Best matching filter for crossproduct  $C$  is best matching filter for packet  $P$
- Example: UDP packet to  $M$  from  $S$ , dest port 53 and source port 57.

# Code

**Build DataStructure:** (\* called whenever a rule changes \*)  
  **For**  $i = 1 \dots K$  (\* K is number of packet fields \*)  
    Let  $S_i$  be the set of distinct fields in field  $i$  of any Rule  
     $Table[i] := BuildPrefix(S_i)$  (\* build best matching prefix table for field  $i$ \*)  
     $CrossProducts := \{X_1, \dots, X_K \text{ such that } X_i \in S_i\}$   
  **For** each  $C \in CrossProducts$   
    Find the earliest matching filter  $F$  that matches  $C$  using linear search  
     $HashInsert(C, F, CrossProductTable)$  (\* insert rule for  $C$  \*)

**FirewallSearch(P)** (\* called on arrival of packet  $P$  \*)  
  **For**  $i = 1 \dots K$   
     $M_i := PrefixLookup(P[i], Table[i]);$   
   $C := M_1 M_2 \dots M_K;$  (\* crossproduct for  $P$  \*)  
  Return ( $HashLookup(C, CrossProductTable)$ ); (\* return matching filter \*)

## On Demand Cross-Producing

- Do not build cross-product tables at start. Only build prefix lookup tables. Avoids huge memory needs of  $N^k$ .
- When a search in table fails, go ahead and compute cross-product and then insert in cross-product table. Remove stale entries periodically. Much better than caching full headers.
- Example: web accesses from internal site to external network to destinations  $D_1, \dots, D_M$  correspond to *two* cross-product terms  $(*, Net, *, *, TCP-ACK)$  and  $(*, Net, *, *, *)$ . Full-header caching will result in  $2M$  distinct entries.

## Equivalenced Cross-Producing (Gupta-Mckeown)

<i>Destination Source Prefix Pairs</i>	<i>Filter Bitmap</i>	<i>Class Number</i>
M, S	11110011	C1
M, TO	11010011	C2
M, Net	11010111	C3
M, *	11110011	C1
T1, S	00000011	C4
T1, TO	00001011	C5
T1 Net	00000111	C6
T1, *	00000011	C4
Net, S	00000011	C4
Net, TO	00000011	C4
Net, Net	00000111	C6
Net, *	00000011	C4
*, S	00000001	C7
*, TO	00000001	C7
*, Net	00000100	C8
*, *	00000001	C7

- Forming partial cross-products of first two columns. Assigns cross-products into same EqId if they have same filter bitmap. 16 partial cross-products only form 8 classes.

## Conclusions

- Hardware based solutions are quite attractive today for use as coprocessor with a network processor in a router.
- Efficient solutions for 2D based on say grid-of-tries. Works well.
- General problem hard but can exploit special features of real databases.
  - On-demand cross-producting is caching based on prefix combinations which has better hit rate than a full header cache.
  - Equivalenced cross-producting is fast but still require 16-30 Mbytes of memory for 100-1000 filters.
  - Extended grid-of-tries works reasonably if port ranges are not arbitrary.
  - Lucent scheme works well for up to 8K filters in hardware.
  - Backtracking works surprisingly well in practice.