University of Massachusetts Amherst

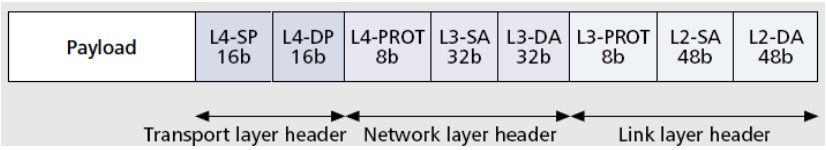
ECE697AA – Lecture 21

Routers: Flow Classification Algorithms

Tilman Wolf
Department of Electrical and Computer Engineering
11/20/08

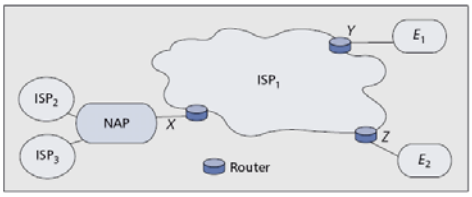
Packet Classification

- What is packet classification?
  - Categorization of packets into flows
- Why do we need classification?
  - Firewalling / NAT
  - Quality of service
  - Defenses against DoS attacks
- How does classification work?
  - Flows are identified by multiple fields



Packet Classification Example

- Example uses of classification



Service	Example
Packet filtering	Deny all traffic from ISP$_3$ (on interface X) destined to E$_2$.
Policy routing	Send all voice-over-IP traffic arriving from E$_1$ (on interface Y) and destined to E$_2$ via a separate ATM network.
Accounting and	Treat all video traffic to E$_1$ (via interface Y) as highest priority and perform accounting for the traffic sent this way.
Traffic rate limiting	Ensure that ISP$_2$ does not inject more than 10 Mb/s of e-mail traffic and 50 Mb/s of total traffic on interface X.
Traffic shaping	Ensure that no more than 50 Mb/s of Web traffic is injected into ISP$_2$ on interface X.

- Example rules:

Flow	Relevant packet fields
Email and from ISP2	Source link-layer address, source transport port number
From ISP$_2$	Source link-layer address
From ISP$_3$ and going to E$_2$	Source link-layer address, destination network-layer address

Packet Classification Example

- Rules with four dimensions:

Rule	Network-layer destination (address/mask)	Network-layer source (address/mask)	Transport-layer destination	Transport-layer protocol	Action
R1	152.163.190.69/255.255.255.255	152.163.80.11/255.255.255.255	*	*	Deny
R2	152.168.3.0/255.255.255.0	152.163.200.157/255.255.255.255	eq www	udp	Deny
R5	152.163.198.4/255.255.255.255	152.163.160.0/255.255.252.0	gt 1023	tcp	Permit
R6	0.0.0.0/0.0.0.0	0.0.0.0/0.0.0.0	*	*	Permit

- Classification results:

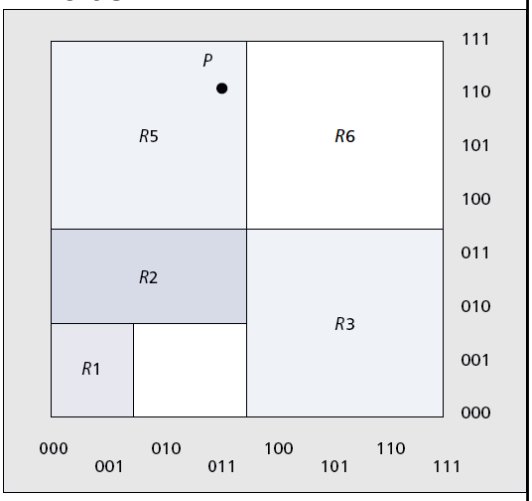
Packet header	Network-layer destination	Network-layer source	Transport-layer destination	Transport-layer protocol	Best matching rule, action
P1	152.163.190.69	152.163.80.11	www	tcp	*R1*, deny
P2	152.168.3.21	152.163.200.157	www	udp	*R2*, deny
P3	152.168.198.4	152.163.160.10	1024	tcp	*R5*, permit

- Descending rule priority (by convention)

Geometric View of Rules

- Illustration of rules on 2 fields:

Rule	F1	F2
R_1	00*	00*
R_2	0*	01*
R_3	1*	0*
R_4	00*	0*
R_5	0*	1*
R_6	*	1*



- Complexity of locating point in N rectangles
  - O(log N) time and O(N^d) space
  - O((log N)$^{d-1}$) time and O(N) space

Complexity of Packet Classification

- High-dimensional problem
  - Typically 4+ fields considered
- Overlapping rules
  - Rule order needs to be considered
- Range matches
  - Can be decomposed into multiple prefixes
- Number of rules
  - Order of thousands
  - Typically not as large as number of prefixes
- Performance constraints
  - Classification at line speed for every packet
  - Memory limited to few Megabytes

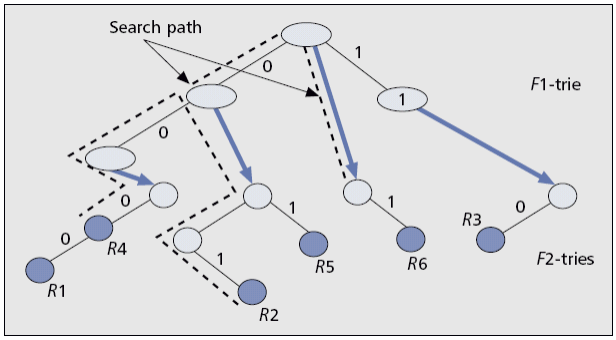
Classification by Linear Search

- Match packet against rules in order of priority
  - First match is highest matching rule
- Space: O(N)
  - Simple list of rules
- Time: O(Nd)
  - Linear growth with number of rules
- Simple, but poor scalability

Classification by Hierarchical Tries

- Use d-dimensional hierarchical trie
  - For each dimension use prefix trie as for lookups
  - Search F1-trie and recursively F2-trie on next tree pointer
- Search: (000,010)

Rule	*F1*	*F2*
R_1	00*	00*
R_2	0*	01*
R_3	1*	0*
R_4	00*	0*
R_5	0*	1*
R_6	*	1*

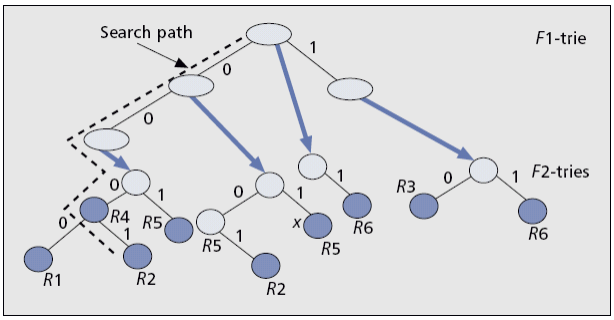


- Space: O(NdW)
- Time: O(W^d)

Classification by Set-Pruning Tries

- Avoid recursive search by replicating rules
  - Search F1 for longest prefix
  - Continue to F2 and search for longest prefix
- Search: (000,010)

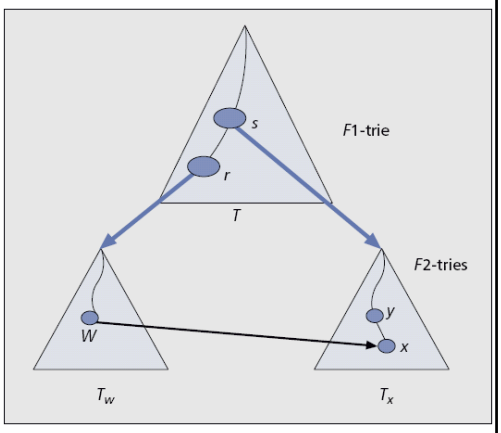
Rule	F1	F2
R_1	00*	00*
R_2	0*	01*
R_3	1*	0*
R_4	00*	0*
R_5	0*	1*
R_6	*	1*



- Space: $O(N^d dW)$
- Time: O(dW)

Classification by Grid-of-Tries

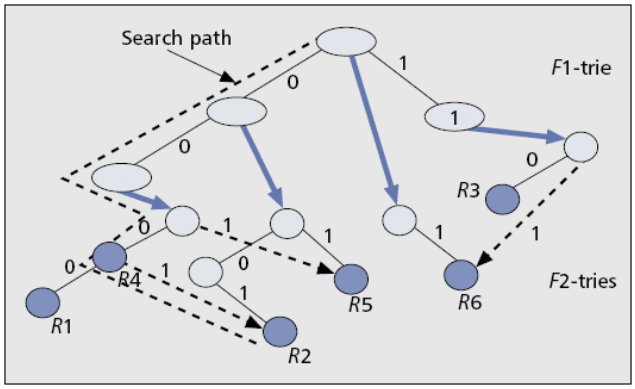
- Attempt to store rule only once and use pointers
  - "Switch pointer" switches F2-trie
- Only possible if
  - Bit string within F2-trie is identical for w and x
  - Node w and node x use same prefix component from F1
  - w does not have child
  - s is closest ancestor to r
- Space: O(NW)
- Time: O(W)
- Works only for 2D



Classification by Grid-of-Tries

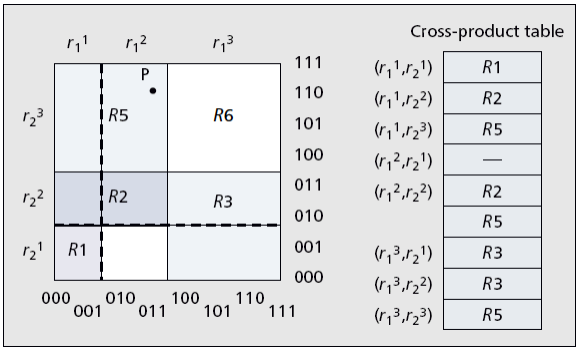
- Example of grid-of-tries
- Search: (000,010)

Rule	F1	F2
R_1	00*	00*
R_2	0*	01*
R_3	1*	0*
R_4	00*	0*
R_5	0*	1*
R_6	*	1*



Classification by Cross-Producting

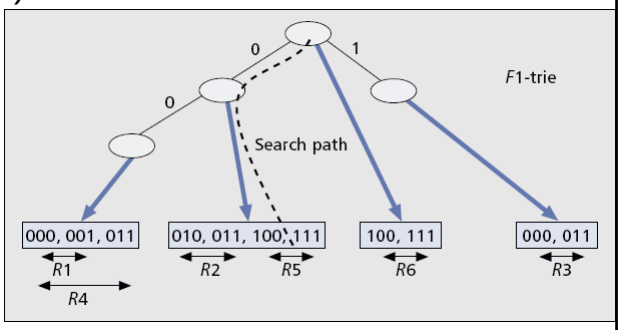
- Rules are partitioned into ranges in each dimension
  - Cross-product table contains best matching rule
  - All possible range combinations precomputed
- Lookup
  - Find range in each dimension
  - Ranges are index into cross-product table
- Space: $O(N^d)$
- Time: $O(dt_{RL})$
- Good for small, static classifiers



2D Classification Scheme

- Restrictions on rules:
  - 1st dimension prefix
  - 2nd dimension arbitrary range
- Search: (011,110)

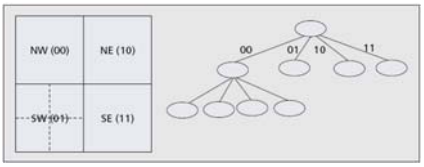
Rule	F1	F2
R_1	00*	00*
R_2	0*	01*
R_3	1*	0*
R_4	00*	0*
R_5	0*	1*
R_6	*	1*



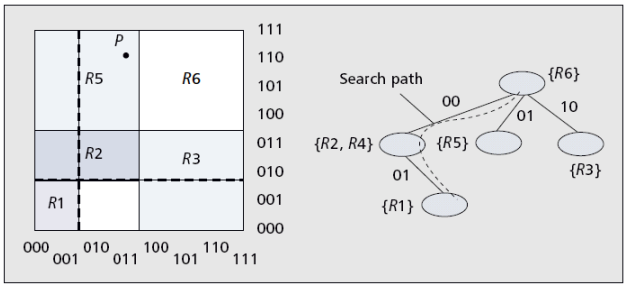
- Space: O(NW)
- Time: O(Wlog N)

Classification by Area-Based Quadtree

- Area-based quadtree (AQT)
  - Four children represent quadrants of 2D subspace
- Search: (001,010)



Rule	F1	F2
R_1	00*	00*
R_2	0*	01*
R_3	1*	0*
R_4	00*	0*
R_5	0*	1*
R_6	*	1*



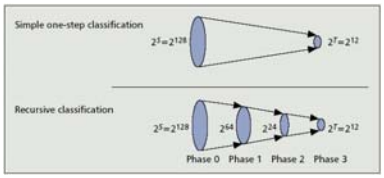
- Space: O(NW)
- Time: O(W)

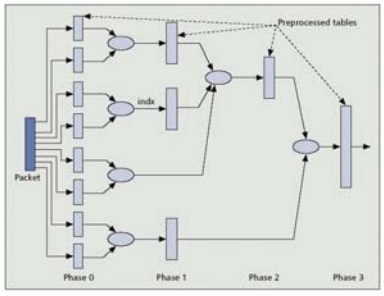
Heuristics

- Solutions to classification can be expensive
  - Worst case analysis
- Classification rules in real networks not worst case
  - Rules structure and redundancy
- Heuristics can exploit structure and redundancy
  - Possibly really bad for theoretical worst case

Recursive Flow Classification

- RFC repeats mapping of multiple fields into one
  - Index into tables decreases with each phase
  - Complex construction of tables
- Performance
  - 1600 rules work
  - Space increases significantly for 6000 rules
  - 10Gbps in hardware and 2.5Gbps in software

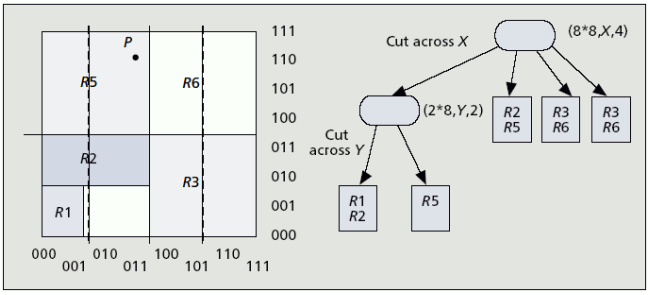




Hierarchical Intelligent Cutting

- HiCuts uses tree structure
  - Space partitioned into subspaces
  - Leaf nodes contain small numbers of rules
  - Sequential search of rules
- Tradeoff between space and query time
- Performance
  - 1700 rules
  - <1MB
  - <20 memory accesses

Rule	F1	F2
R_1	00*	00*
R_2	0*	01*
R_3	1*	0*
R_4	00*	0*
R_5	0*	1*
R_6	*	1*



Tuple Space Search

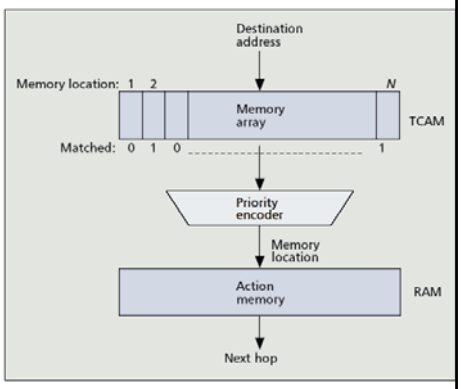
- Length of prefix in each rule determines tuple
  - Multiple rules with same tuple can be stored in hash table
    - » Same size prefix for all rules within that hash table
  - Search is exact match operation on table
- Space and time good for average case
  - Nondeterministic time due to hashing
  - Could become bad for some cases

Rule	Specification	Tuple
R1	(00*,00*)	(2,2)
R2	(0**,01*)	(1,2)
R3	(1**,0**)	(1,1)
R4	(00*,0**)	(2,1)
R5	(0**,1**)	(1,1)
R6	(***,1**)	(0,1)

Tuple	Hash table entries
(0,1)	{R6}
(1,1)	{R3,R5}
(1,2)	{R2}
(2,1)	{R4}
(2,2)	{R1}

Hardware-Based Algorithms

- Some hardware components exhibit useful features
  - Content addressable memories (CAMs) allow O(1) search
- Classification by Ternary CAMs (T-CAMs)
  - Prefixes stored as value and mask
  - Priority encoder picks one of multiple matches and yields index to action memory
  - Fast, but expensive and power hungry



Summary

- Performance overview:
  - Many algorithms perform better in average case

Algorithm	Worst-case time complexity	Worst-case storage complexity
Linear search	N	N
Ternary CAM	1	N
Hierarchical tries	W^d	NdW
Set-pruning tries	dW	N^d
Grid-of-tries	W^{d-1}	NdW
Cross-producting	dW	N^d
FIS-tree	$(l + 1)W$	$l \times N^{1+1/l}$
RFC	d	N^d
Bitmap-intersection	$dW + N/memwidth$	dN^2
HiCuts	d	N^d
Tuple space search	N	N

Homework

- Read:
  - Kurose & Ross: Chapter 7.5
- SPARK
  - Assessment quiz