FPL 2010
milano, italy

USC

USC Viterbi
School of Engineering

Decision Forest: A Scalable Architecture for Flexible Flow Matching on FPGA

Weirong Jiang, Viktor K. Prasanna
University of Southern California
Norio Yamagaki
NEC Corporation
September 1, 2010

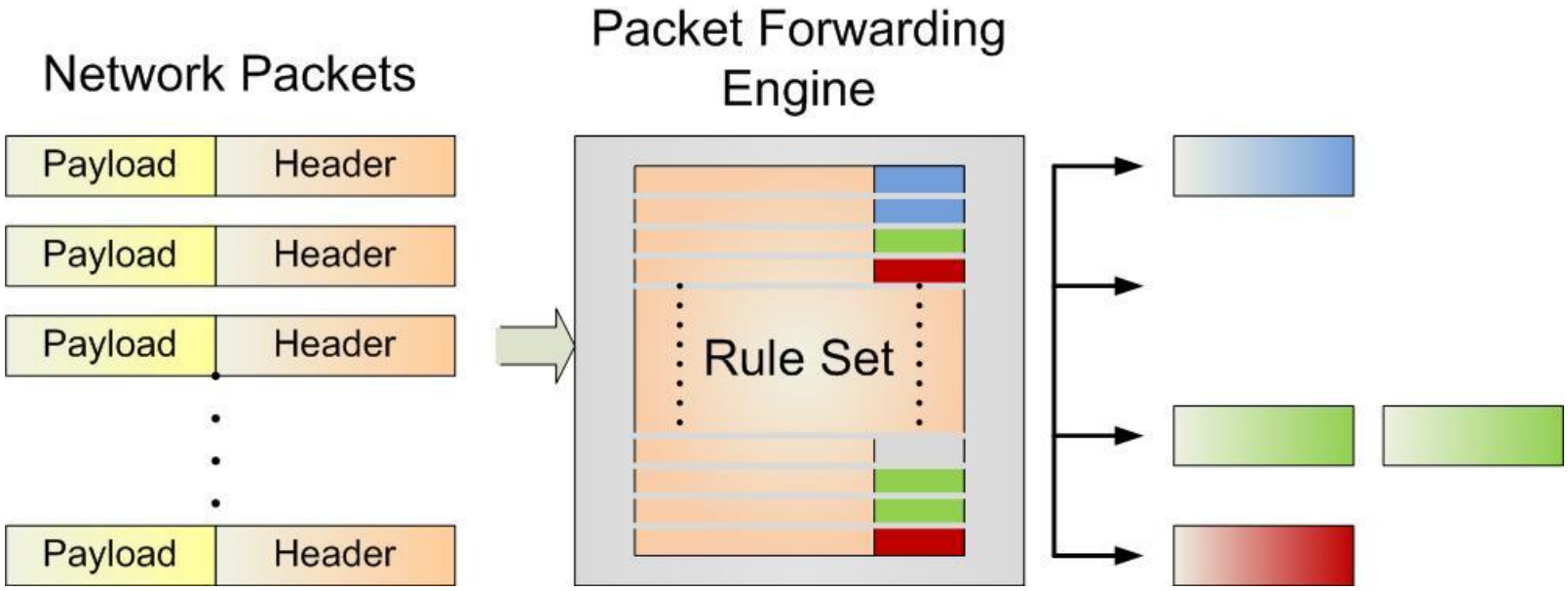
Outline

- Background
- Related Work
- Our Solution
- Conclusions

Background

USC Viterbi School of Engineering

- **Packet Forwarding in Network Routers**
  - **Matching packets against a (large) set of forwarding rules**
    - Each rule specifies the matching conditions and resulting actions
  - **Take corresponding actions of the matching rule**
    - Forward to a specified interface / Drop packets



Traditional Packet Forwarding

USC Viterbi School of Engineering

- **IP Lookup**
  - **Look up the routing table using the destination IP address of a packet**
  - **Each route entry: {prefix | next-hop}**
    - Prefix: represent a subnet
  - **Longest prefix matching**

Routing table:

Prefixes	Next-hop interface
128.123.*.*	Port 0
128.123.111.*	Port 2

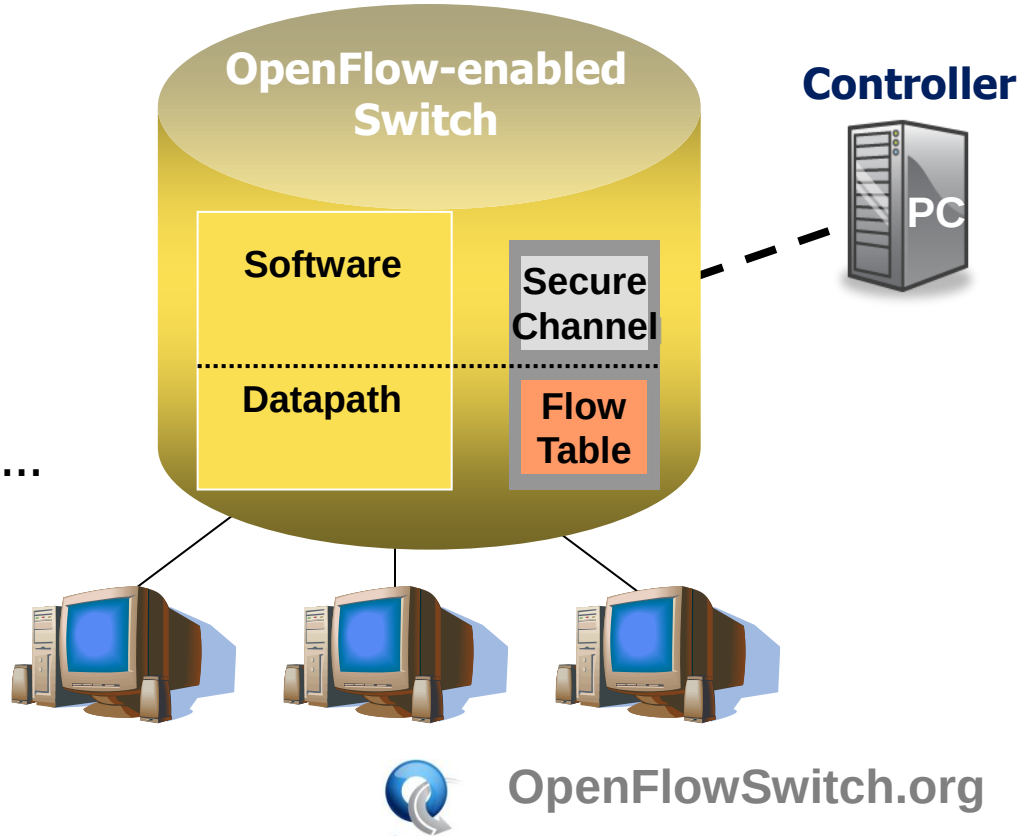
IP:
128.123.111.233 →

- **Other Packet Forwarding Problems**
  - **MAC forwarding**
  - **Packet classification**
  - **etc.**

Flexible Flow Matching

USC Viterbi School of Engineering

- **Motivation**
  - **Router Supporting Network Virtualization**
    - Multiple virtual networks employing different types of forwarding rules
- **OpenFlow Switching**
  - **Centralized control**
  - **Flexible flow table**
  - **Initiated in 2008**
    - 60+ Universities in US
    - NEC, Juniper, HP, Cisco, ...

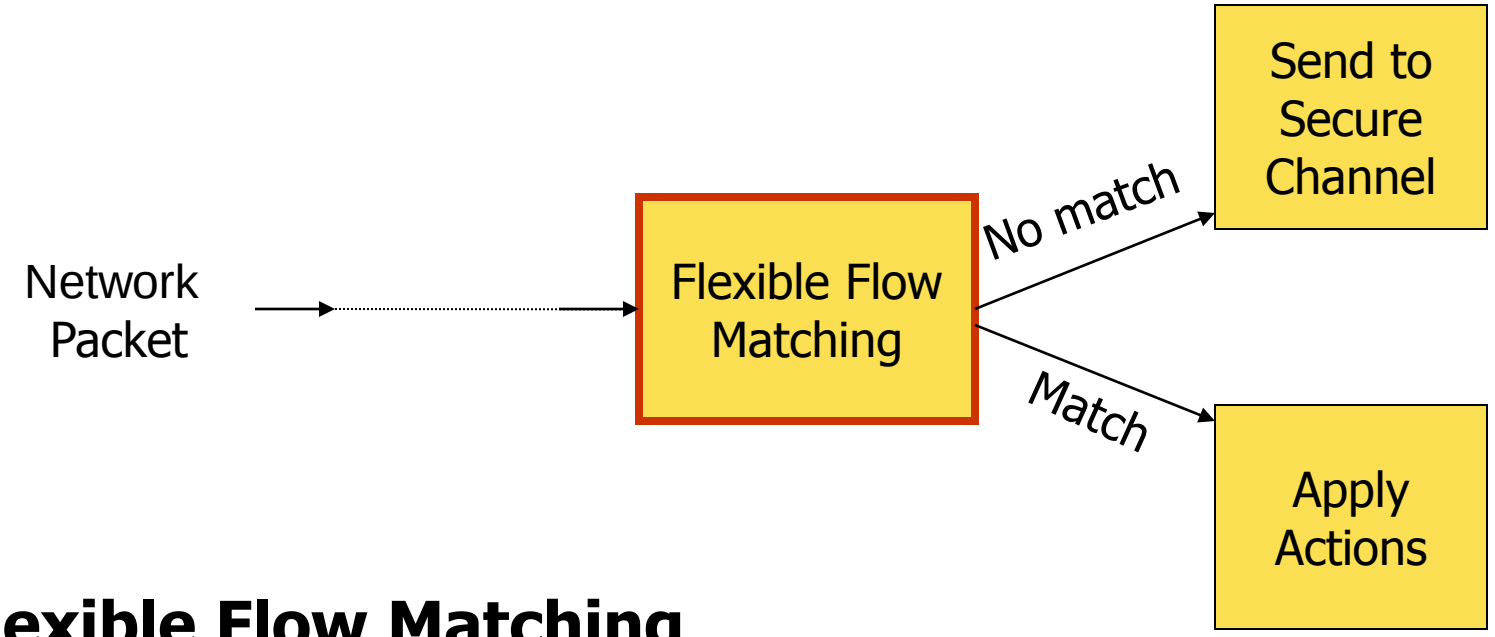


Example OpenFlow Rules

USC Viterbi School of Engineering

Forwarding types	Ingress Port	Eth src	Eth dst	Eth type	VLAN ID	VLAN prior	IP src	IP dst	IP Prtl	IP ToS	TCP sport	TCP dport	Action
Ethernet Switching	*	*	00: 1F	*	*	*	*	*	*	*	*	*	Port 6
IP Routing	*	*	*	*	*	*	*	1.2.3.4	*	*	*	*	Port 9
Application Firewall	*	*	*	*	*	5	*	*	*	*	*	22	Drop
Flow Switching	Port 2	00: 23	00: 42	0800	vlan1	*	1.2.3.4	5.6.7.8	4	*	1743 4	80	Port 3
Port + Ethernet + IP	Port 3	00: 23	*	0800	*	7	*	2.3.4.5	4	0	*	*	Port 13

OpenFlow-enabled Switch / Router



- **Flexible Flow Matching**
  - **12-tuple Packet Header Fields**
  - **Exact/ Wildcard/ Prefix match**
  - **Priority**
    - If a packet matches multiple entries, the one with the highest priority is used.

OpenFlow Switching Consortium

USC Viterbi School of Engineering

- **Initiated in 2008**
  - **60+ Universities in US**
  - **NEC, Juniper, HP, Cisco, …**
- **Goal: Put an Open platform in the hands of researchers/students to test new ideas at scale**
- **Network Programmability**
  - **"Enabling Innovation in Campus Networks"**
- **Network Virtualization**
  - **"Enables atomic access to (virtualized) resources"**

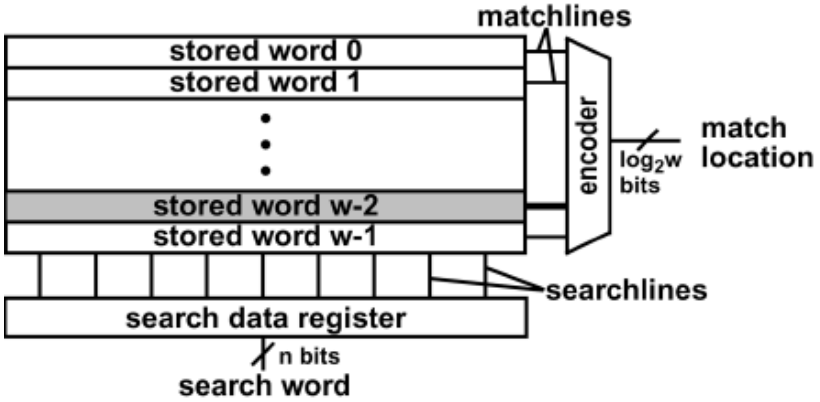
Challenges

- **Throughput**
  - **e.g. 40 Gbps, i.e. 125 million (40 byte) packets per second**
- **Wide Rules**
  - **12-field -->12-dimensional search**
- **Sparse Wildcards**
  - **Difficult for hashing**
  - **State of the art solution: Linear search**

- **Current OpenFlow Research**
  - **Focus on functionality**
- **Little work**
  - **to address the performance challenges**

Related Work: TCAM

- **Ternary Content Addressable Memory (TCAM)**
  - **De facto solution for packet forwarding engines**
    - Store an entry in a word
    - Parallel search on all words
    - Brute-force
  - **Scalable?**
    - High power/energy consumption
    - Low clock rate
    - Large area



	TCAM (18 Mb)	SRAM (18 Mb)
Clock rate	266 MHz	450 MHz
Power	12 ~ 15 Watts	1.2 ~ 1.6 Watts
Cell size	16 transistors	6 transistors

Motivation

- **Packet Classification → Flexible Flow Matching**
  - **Packet classification: 5-tuple**
    - IP src/dst, Port src/dst, IP protocol
  - **OpenFlow: 12-tuple**
- **Algorithmic Solutions → SRAM-based Pipelines**
  - **Decision tree -based packet classification algorithm**
  - **Decision forest: multiple decision trees**
- **Modern FPGAs**
  - **Massive parallelism**
  - **Large amount of dual-port RAMs**
  - **High clock rate, e.g. 550 MHz (Xilinx Virtex-5)**

Packet Classification Example

USC Viterbi School of Engineering

Example Rule Set

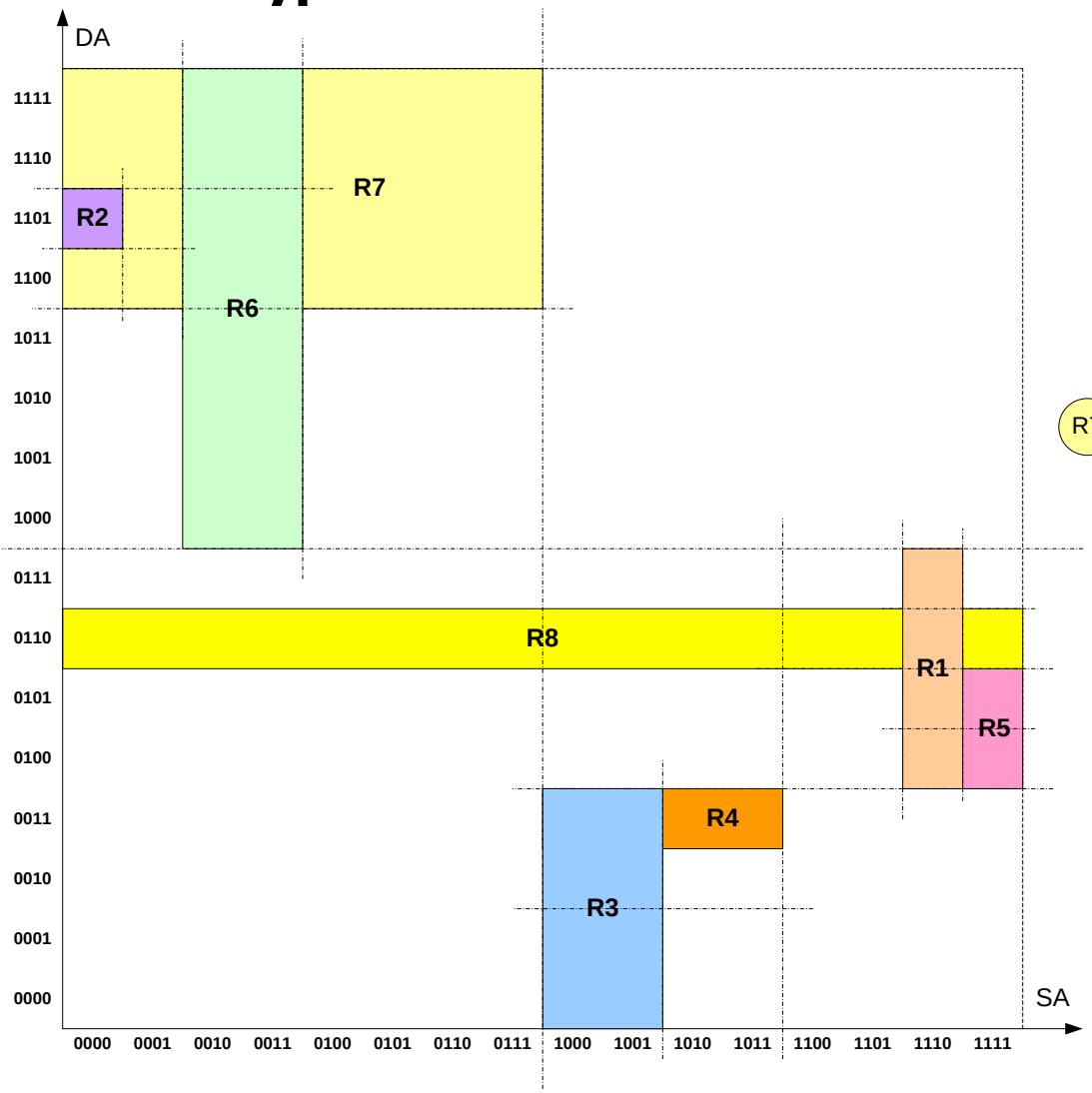
Rule	SA (8-bits)	DA (8-bit)	SP (4-bit)	DP (4-bit)	Prtl (2-bit)	Priority
R1	1110*	01*	[1:3]	[6:7]	1	1
R2	0000*	1101*	[10:13]	[2:6]	2	1
R3	**100***	**00***	**[2:5]**	**[6:7]**	**3**	2
R4	**10***	**0011***	**[0:3]**	**[6:7]**	**3**	3
R5	111*	010*	[2:5]	[6:7]	0	3
R6	001*	1*	[1:3]	[6:7]	1	4
R7	0*	11*	[1:3]	[6:7]	2	5
R8	*	0110*	[0:15]	[0:7]	3	6

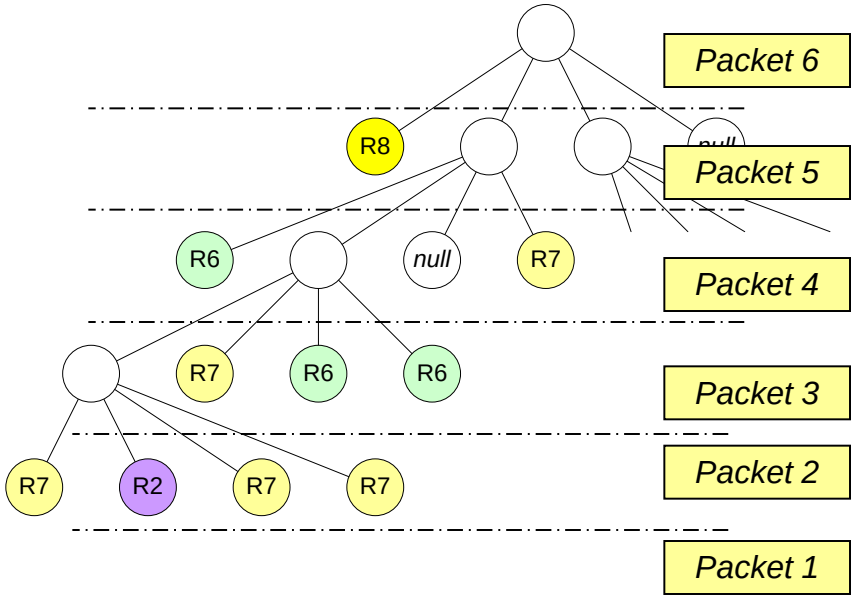
Packet

SA (8-bits)	DA (8-bit)	SP (4-bit)	DP (4-bit)	Prtl (2-bit)
10000000	**00110100**	**2**	**6**	**3**

USC Viterbi School of Engineering

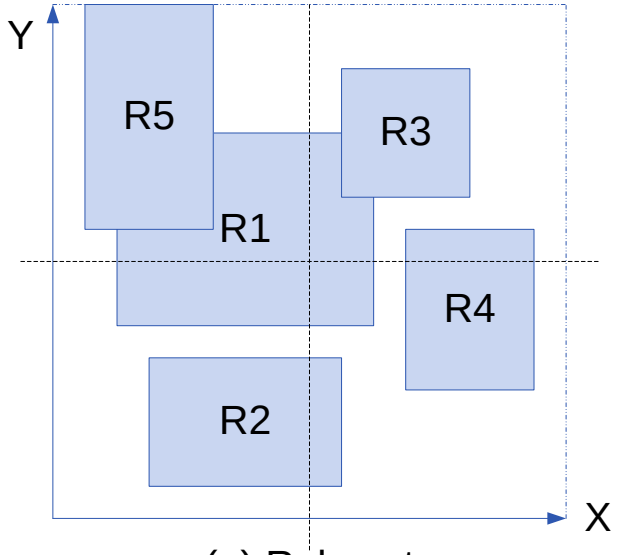
- **Decision-tree-based Algorithm**
  - **HyperCuts**



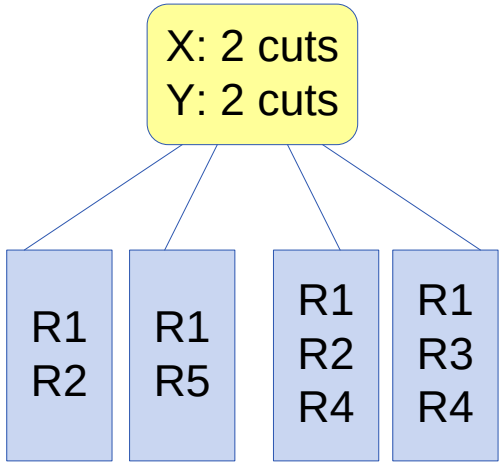


Drawbacks of Existing Algorithm

- **Memory Explosion**
  - **Due to rule duplication**
  - $O(N^D)$



(a) Rule set



(b) HyperCuts

- **Large Tree Depth**
  - **High latency**
- **Even worse for flexible flow matching!**
  - **D = 12**

Our Idea: Rule Set Partitioning

USC Viterbi School of Engineering

- **Example OpenFlow Table**

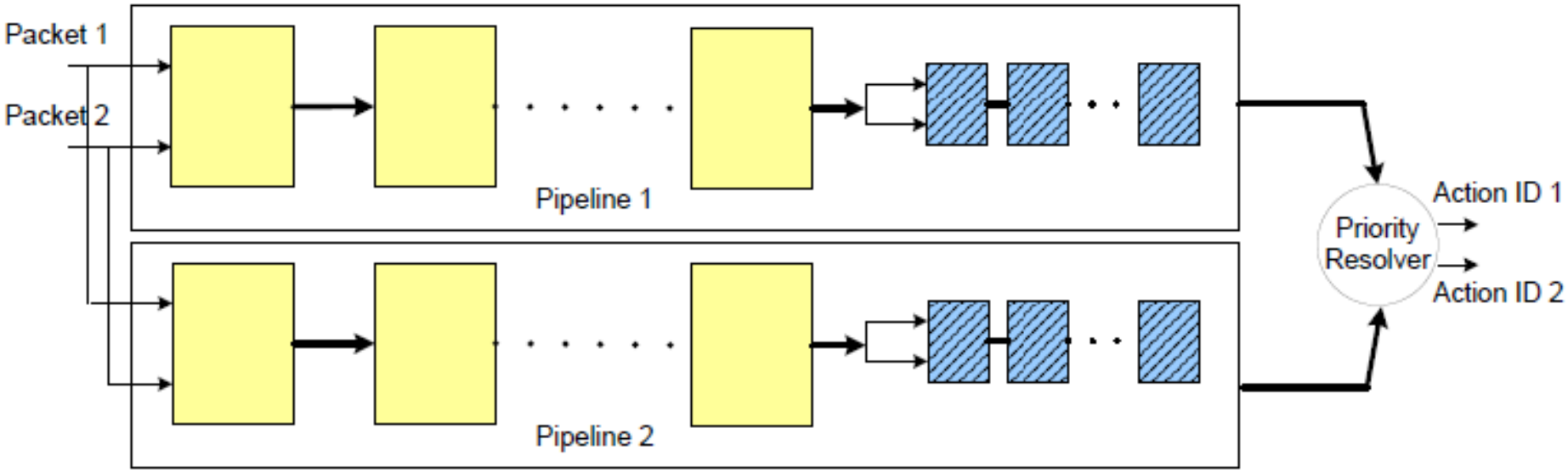
Rules	Switch port	MAC src	MAC dst	Eth type	VLAN ID	IP src	IP dst	IP port	TCP sport	TCP dport
R1	*	2	5	*	*	*	*	*	*	*
R2	*	*	*	*	*	111	223	*	*	*
R3	*	*	*	*	*	62	59	*	*	*
R4	*	6	9	*	*	*	*	*	*	*

- **Motivation**
  - **Partition the flow table: {R1, R4} and {R2, R3}**
  - **Each subset built into a decision tree**
    - Each tree uses a small number of header fields
  - **Theoretically, memory requirement:**
    - $O(P(N/P)^D) = O(N^D/P^{D-1})$

Architecture: Decision Forest

USC Viterbi School of Engineering

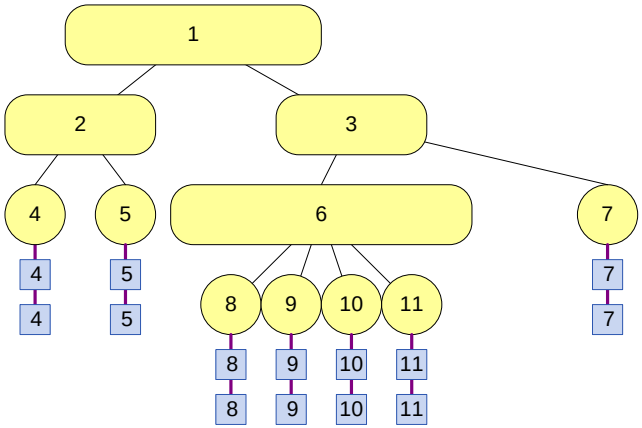
- **Multiple Depth-Bounded Decision Trees**
  - **Split out trees one-by-one**
- **Mapping Each Tree to a Linear Pipeline**
  - **P trees → P pipelines**
- **Dual-port RAMs**
  - **2x throughput**



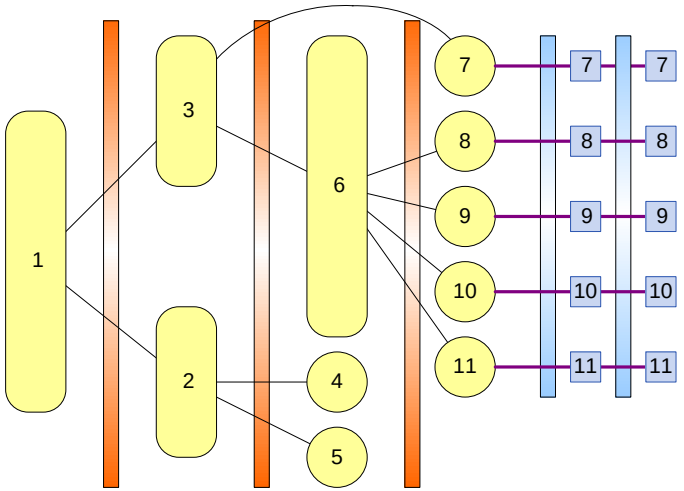
- P = 2

Tree-to-Pipeline Mapping

- **Decison Tree**
  - **Internal nodes**
  - **Rule list at each leaf**



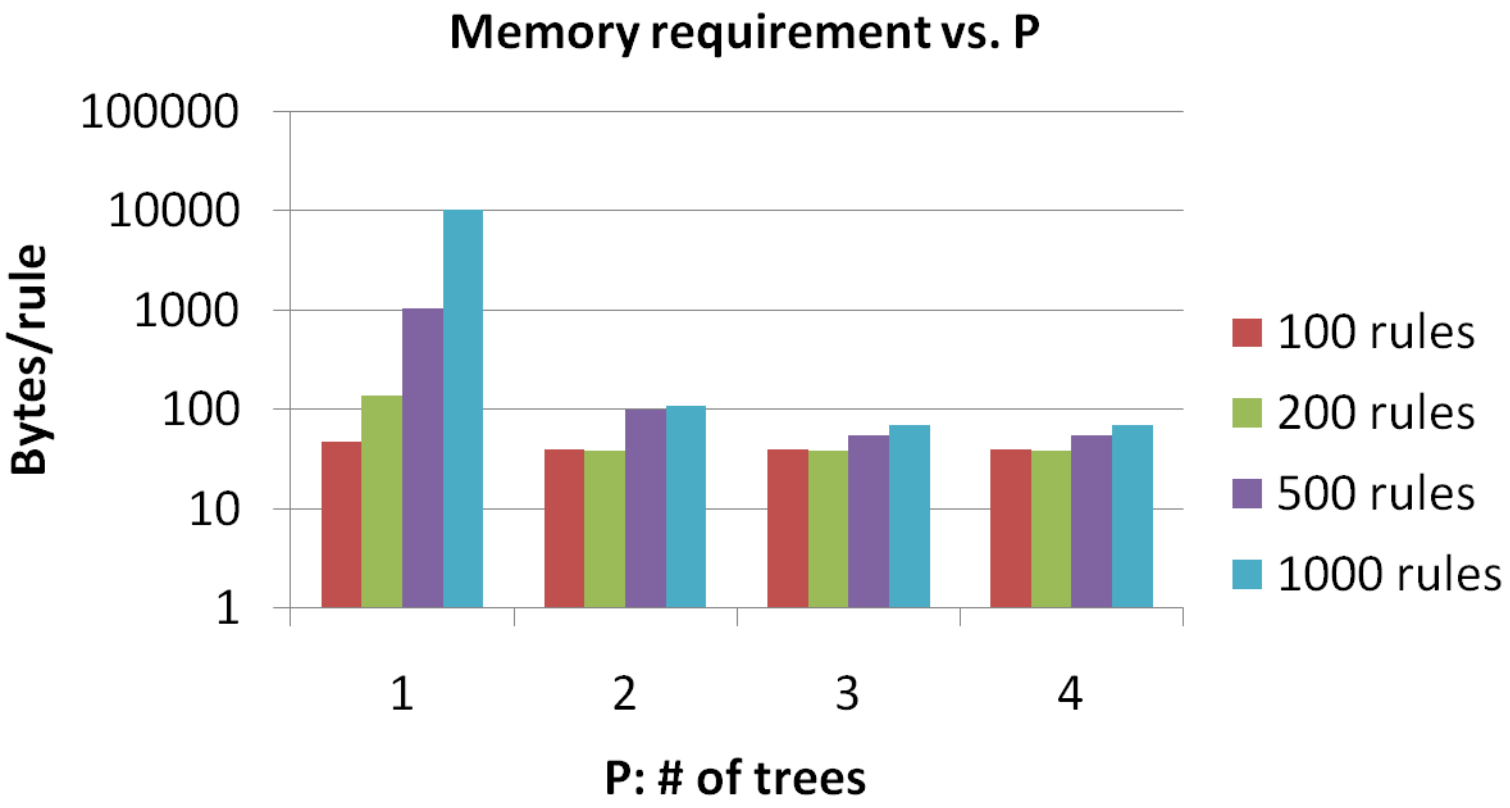
- **Linear Pipeline**
  - **Memory balancing**



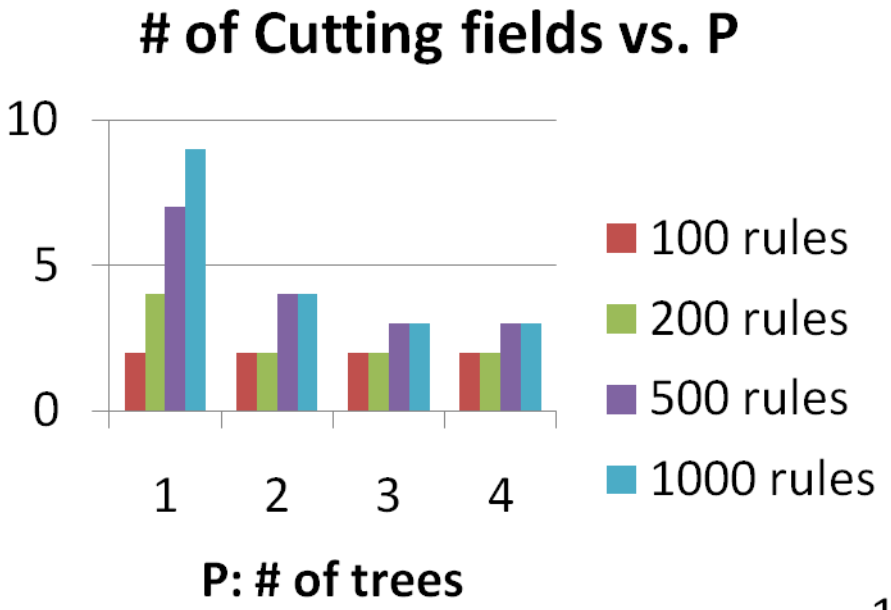
Performance Evaluation

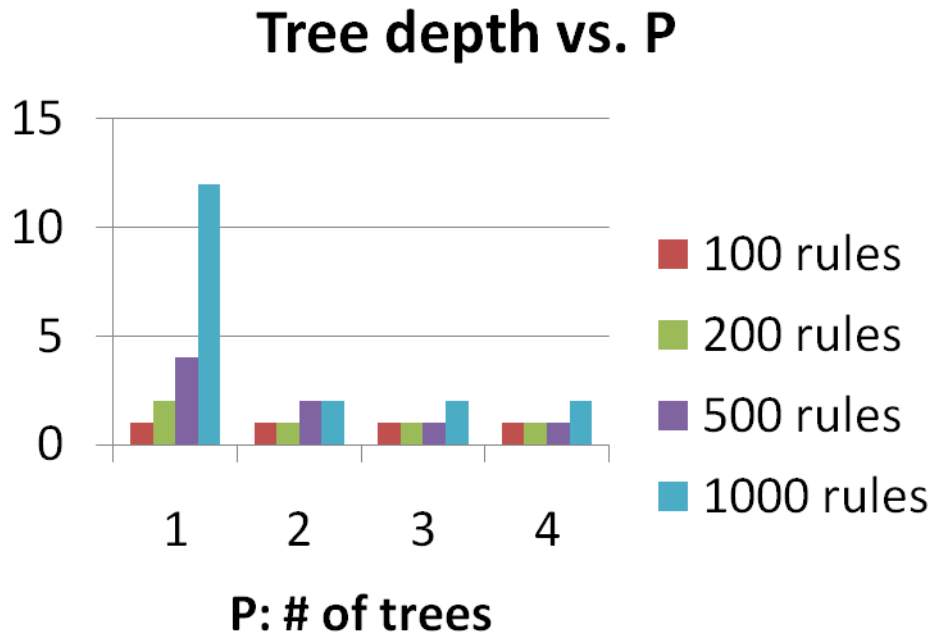
USC Viterbi School of Engineering

- **Using OpenFlow-like 12-tuple Rules**
  - **Randomly generated**



Performance Evaluation (2)





FPGA Implementation

USC Viterbi School of Engineering

- **Xilinx Virtex-5 XC5VFX200T**
  - **1K 12-tuple rules**
  - **125 MHz → 40 Gbps**

	Available	Used	Utilization
# of Slices	30,720	11,720	38%
# of 36Kb BlockRAMs	456	256	56%
# of User I/Os	960	303	31%

- **Comparison**

	# of complex rules	Throughput
Our design	1000	40 Gbps
TCAM on NetFPGA [Naous: ancs08]	32	4 Gbps

Summary

- **First attempt to address the performance challenges for flexible packet forwarding**
  - **Flexible forwarding itself is evolving**
- **Algorithmic Solution + Parallel Architecture**
  - **Decision Forest**
    - Partitioning the rule set
    - Multiple decision trees
  - **Reducing overall memory & resouce requirement**
- **FPGA is an attractive option for implementing network processing engines**
- **Future Work**
  - **Optimal partitioning?**

Q & A

- **Questions?**

Our Solution: Algorithms

USC Viterbi School of Engineering

Algorithm 1 Building the decision forest

Input: Rule set R.
Input: Parameters: $bucketSize$, $depthBound$, P.
Output: Decision forest: $\{T_i | i = 0, 1, \cdots, P-1\}$.

1: $i \leftarrow 0$, $R_i \leftarrow R$ and $split \leftarrow TRUE$.
2: **while** $i < P$ **do**
3: **if** $i == P - 1$ **then** {The last subset / tree}
4: $split \leftarrow FALSE$
5: **end if**
6: $\{T_i, R_{i+1}\} \leftarrow BuildTree\ (R_i,\ split,\ bucketSize,\ depthBound)$
7: $i \leftarrow i + 1$
8: **end while**

Our Solution: Algorithms (2)

USC Viterbi School of Engineering

Algorithm 2 Building the decision tree and the split-out set: $\{T, R_{ex}\} \leftarrow BuildTree(R, split, bucketSize, depthBound)$

Input: Rule set R.
Input: Parameters: $split$, $bucketSize$, $depthBound$.
Output: Decision tree T and the split-out set R_{ex}.

1: Initialize the root node: $root.rules \leftarrow R$.
2: Push $root$ into $nodeList$.
3: **while** $nodeList \neq null$ **do**
4: $n \leftarrow Pop(nodeList)$
5: **if** $n.numrules < bucketSize$ **then**
6: n is a leaf node. Continue.
7: **end if**
8: **if** $n.depth == depthBound$ **then**
9: Assign to n the $bucketSize$ most specified rules from $n.rules$. Push remaining rules of $n.rules$ into R_{ex}. n is a leaf node. Continue.
10: **end if**
11: **for** $f \in OptFields(n)$ **do**
12: $nCuts[f] \leftarrow OptNumCuts(n, f)$
13: $n.numCuts$ *= $nCuts[f]$
14: **end for**
15: **if** $split$ is TRUE **then**
16: $r \leftarrow PotentialDuplicatedRule(n, nCuts)$
17: Push r into R_{ex}.
18: **end if**
19: **for** $i \leftarrow 0$ to $2^{n.numCuts} - 1$ **do**
20: $n_i \leftarrow CreateNode(n, nCuts, i)$
21: Push n_i into $nodeList$.
22: **end for**
23: **end while**

Depth bounding

Split-out