Packet Classification Using Multidimensional Cutting

Systems & Networking

Sumeet Singh, Florin Baboescu, George Varghese

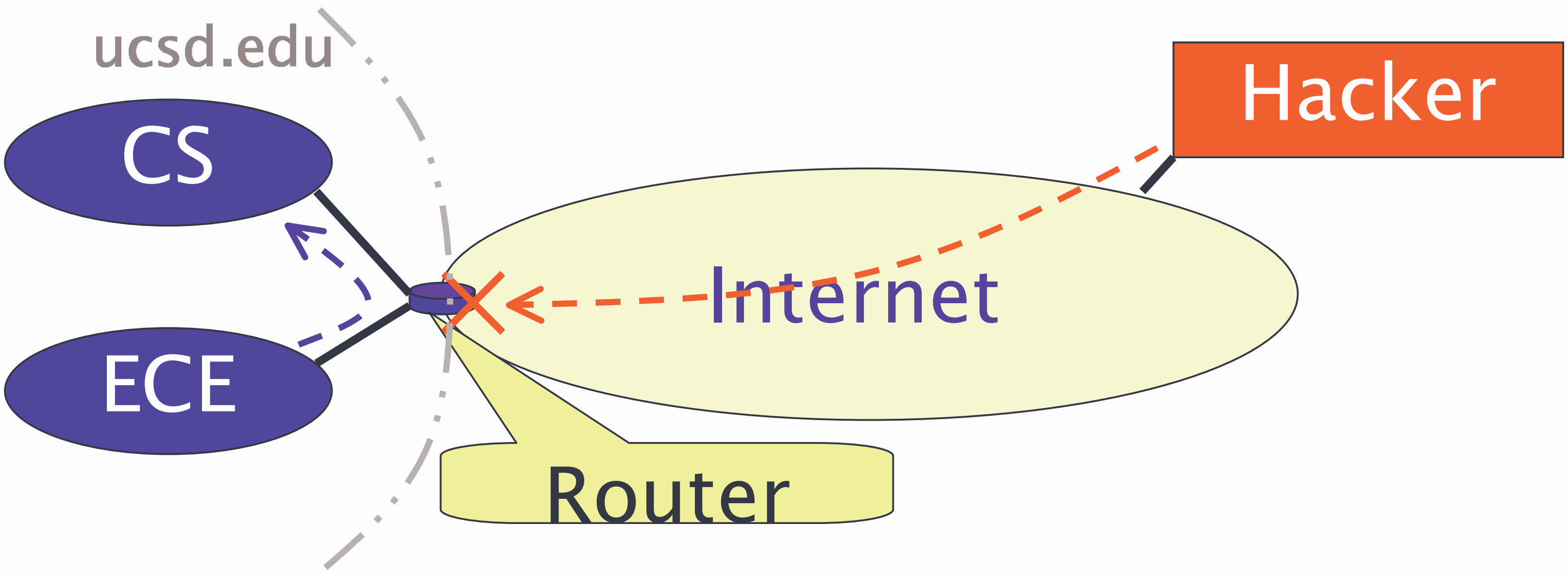
University of California, San Diego (UCSD)

&

Jia Wang

AT&T Labs- Research

Packet Classification (forwarding based on multiple fields)



Rules	Destination	Source	Destination Port	Action
Rule1	*cs*	*ece*	*	*10Gbps*
Rule2	*	*hacker*	*NetBios*	*Deny*

Classifier → A set of predicates (rules).

Packet Classification → Finding the Action associated with the highest priority rule (matching all dimensions) in the classifier.

Rules of the Game

- Fast search **speed**
  (4–32ns/pkt throughput)
- Low **storage** requirements
  (less than several Mbits)
- **Scalability** in the number of rules
  (up to 100K rules)
- **Scalability** in the number of fields
  (five fields or more)

Packet Classification : A Crowded Space

1998
Bit Vector

1999
Grid of Tries, Crossproducting
RFC, HiCuts

2000
FIS Trees

2001
ABV

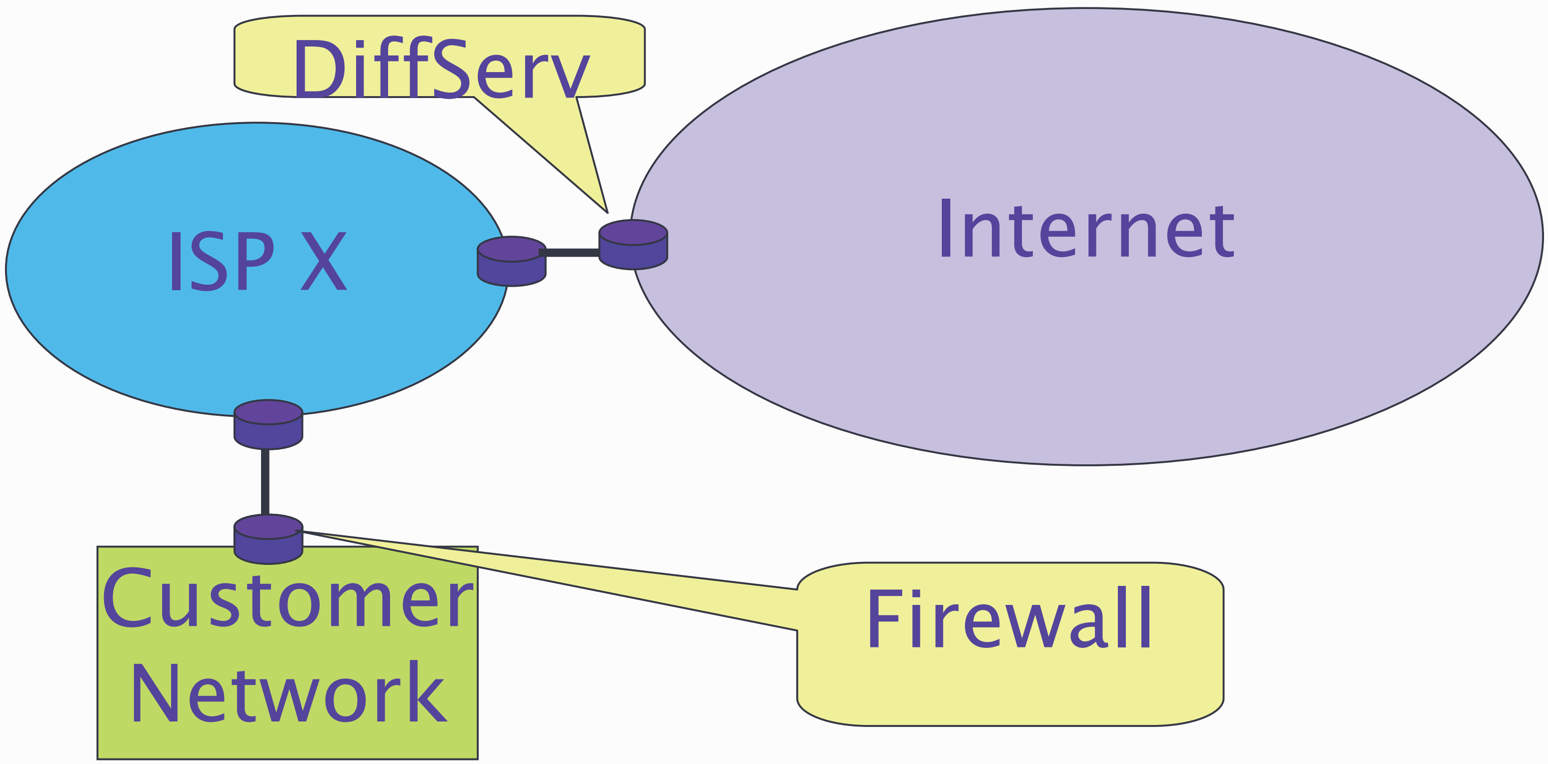
2003
HyperCuts *(this paper)*

Why yet another paper on Packet Classification?

Three Reasons for another solution

A. Increasing importance of Packet classification.
B. Inadequate performance of existing schemes:
   - CAMs
   - Algorithmic solutions
C. Possibility of new ideas.

A) Increasing Importance of Packet Classification



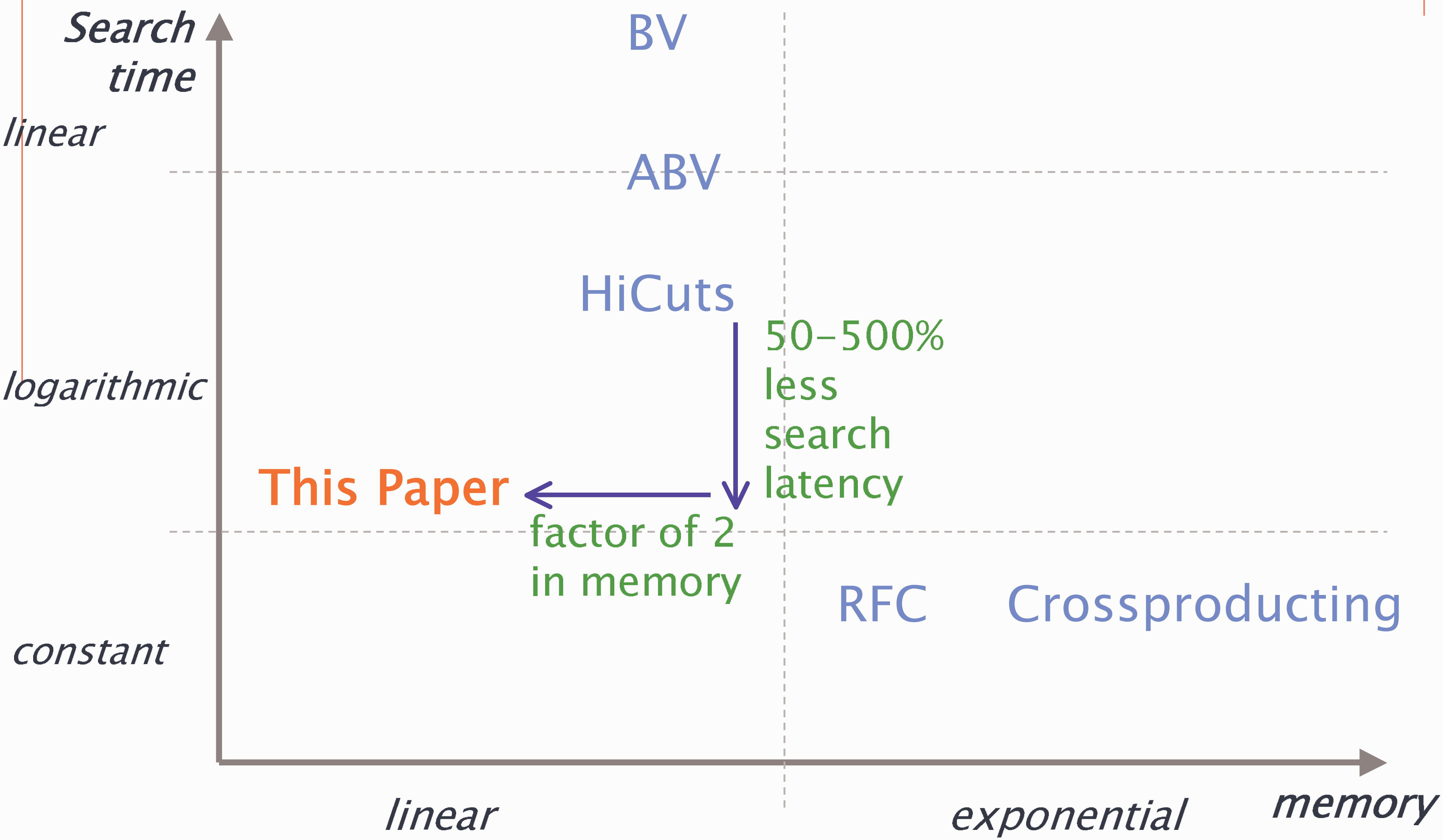
- Increased demand for new services
  - QoS
  - Security
- Increased speed
  - In 2004, 21% of edge routers will be OC-192 (10Gbps)

B) Inadequate Performance of CAM based solutions

- Content Addressable Memory
  - Hardware Solution (using parallelism)
  - Widely used in the Industry
- Pros:
  - Low latency and high throughput
  - Simple on-chip management scheme
- Cons:
  - High power (heat!)
  - Large die size (more board space)
  - High cost (compared to SRAM based solutions)
  - All fields must be expressed into a prefix format

An algorithmic solution may be a contender!

B) Inadequate Performance of Existing Algorithmic Schemes



C) Possibility of New Ideas

- Main Idea:
  - Increasing degrees of freedom involved in decision tree approaches to classification, by using hypercubes to partition the search space instead of hyperplanes.

Outline

1. Introduction
2. Geometric View of Packet Classification
3. Basic Decision Tree Approaches
4. Basic HyperCuts
5. HyperCuts Optimizations
6. Experimental Results
7. Conclusion

Geometric View of Packet Classification

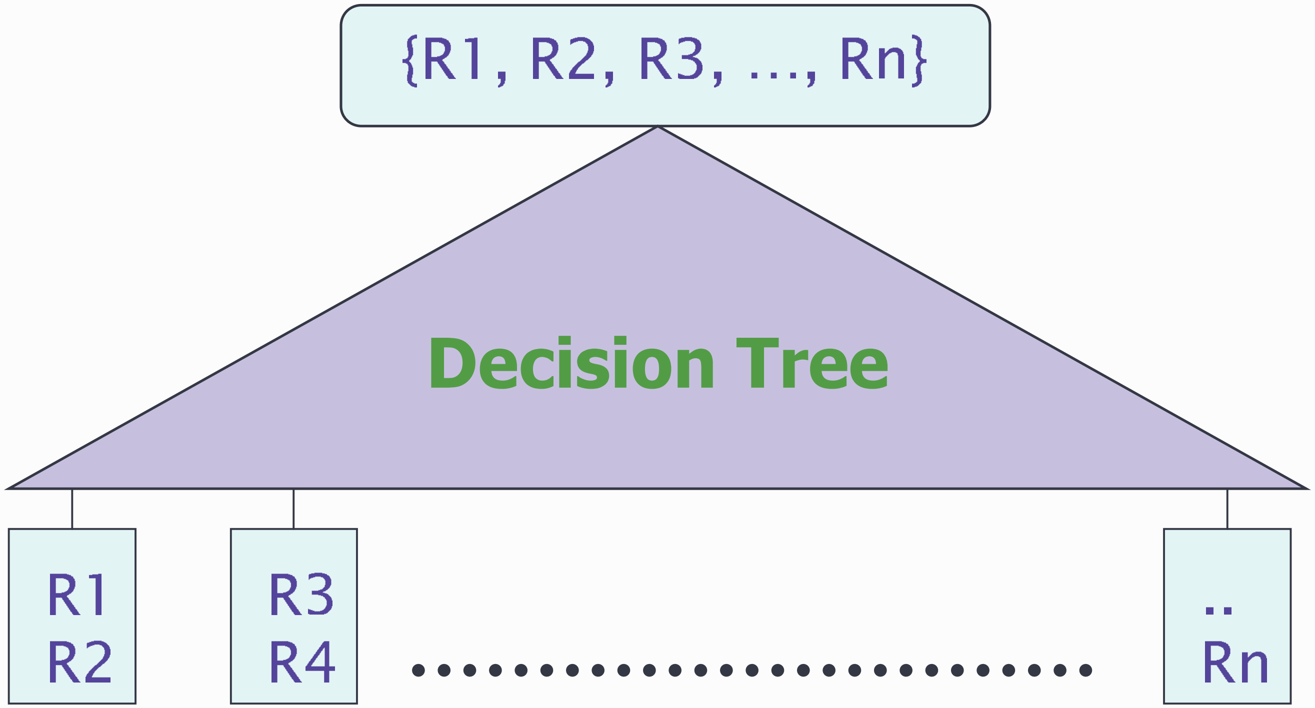
Rules	Source	Destination
$Rule_1$	0 – 127	0 – 127
$Rule_2$	192	0 – 255
$Rule_3$	32	160

Prefixes represented as ranges

Outline

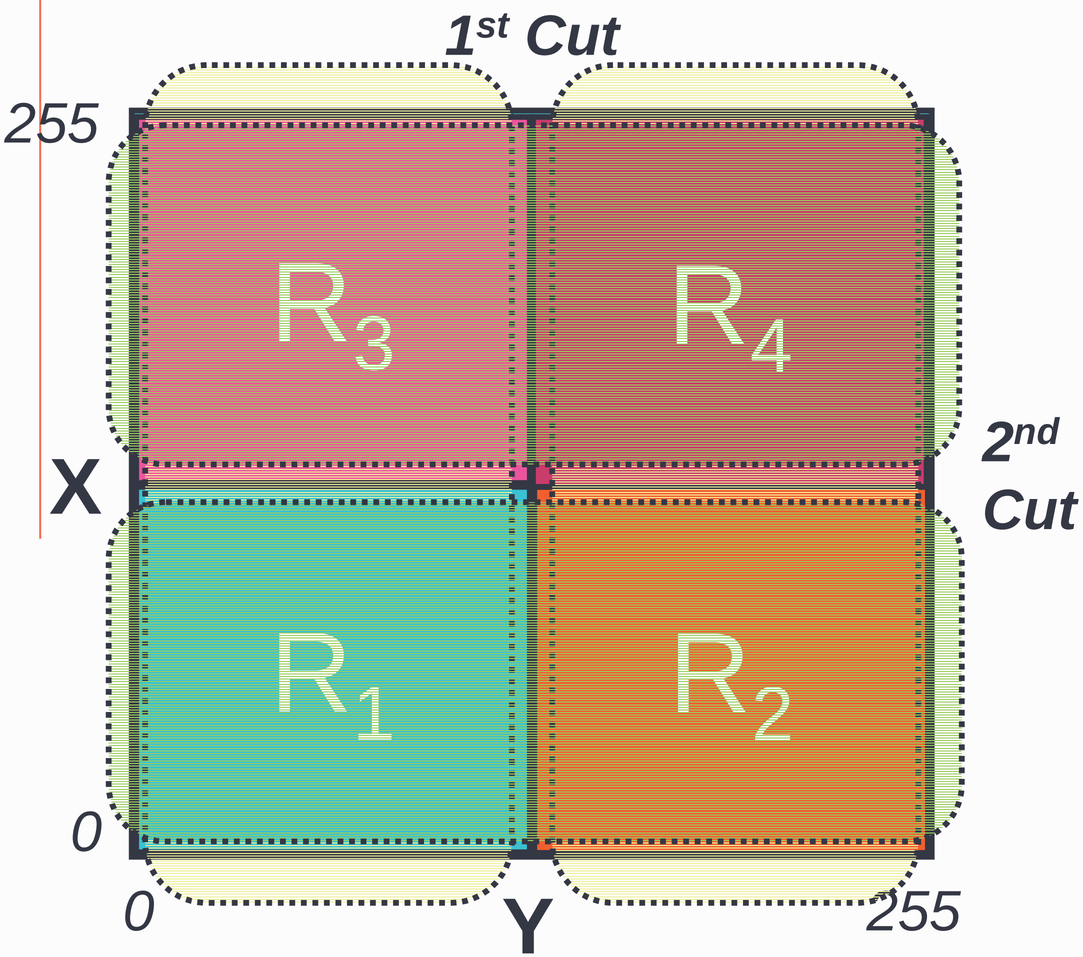
1. Introduction
2. Geometric View of Packet Classification
3. Basic Decision Tree Approaches
4. Basic HyperCuts
5. HyperCuts Optimizations
6. Experimental Results
7. Conclusion

Decision Tree Based Classification

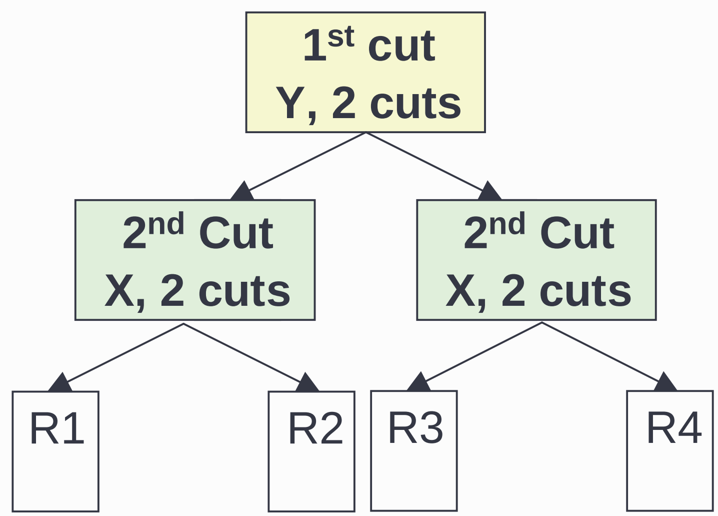


Pioneered by Woo and Gupta-McKeown

HiCuts:Using single-dimension cutting



R_1	0*	0*
R_2	0*	1*
R_3	1*	0*
R_4	1*	1*

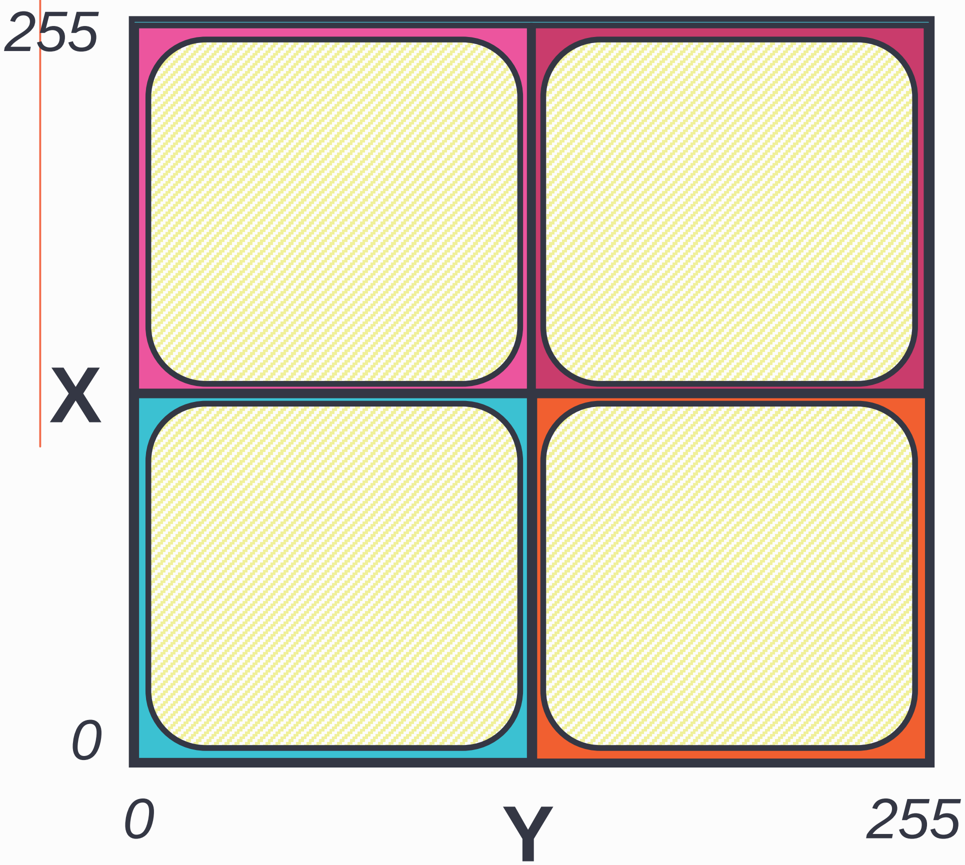


Gupta-McKeown 99

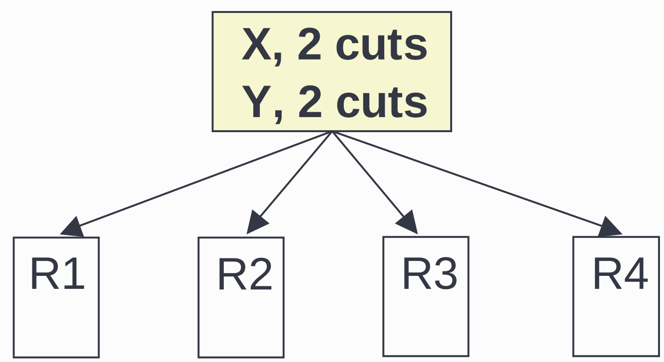
Outline

1. Introduction
2. Geometric View of Packet Classification
3. Basic Decision Tree Approaches
4. Basic HyperCuts
5. HyperCuts Optimizations
6. Experimental Results
7. Conclusion

Using multidimensional cutting

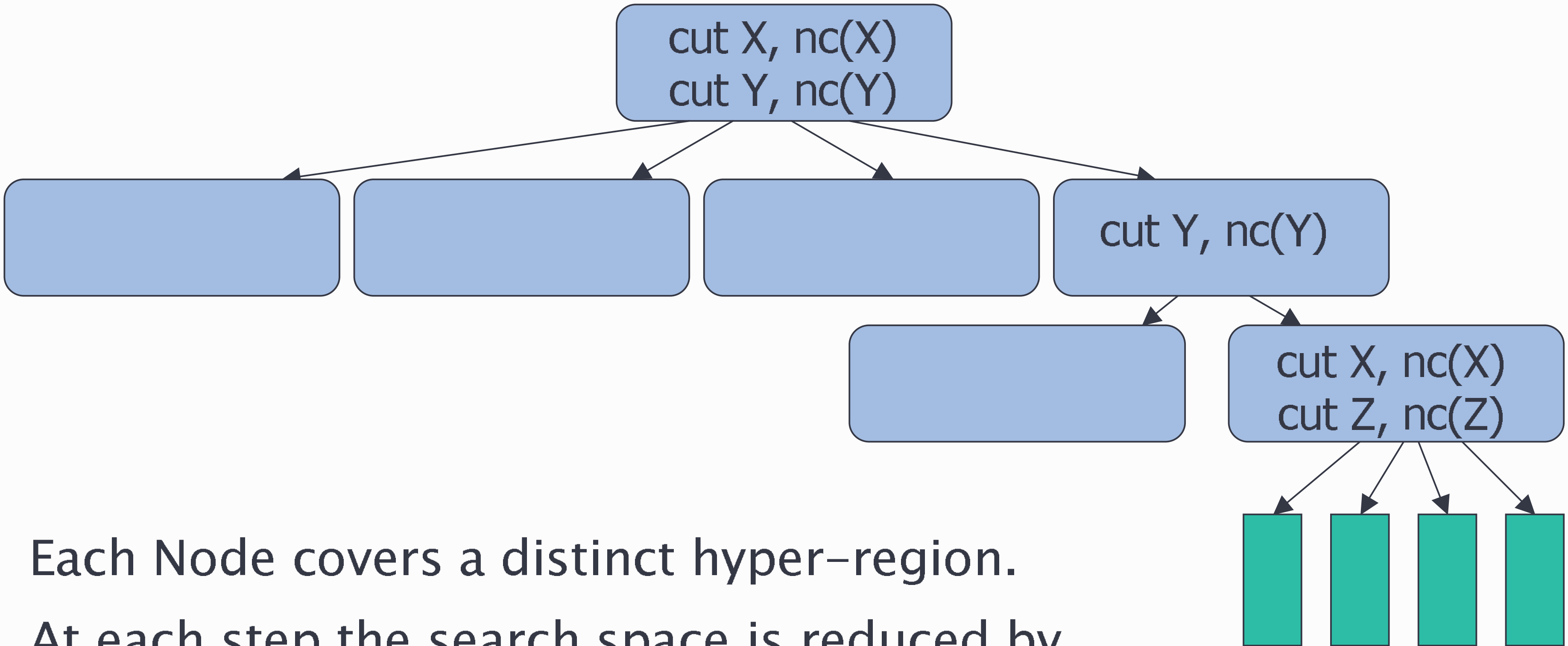


R_1	0*	0*
R_2	0*	1*
R_3	1*	0*
R_4	1*	1*



Cuts are equal size ranges on each dimension, for easy array indexing.
The number of cuts in each dimension may be different.

A HyperCuts Decision Tree

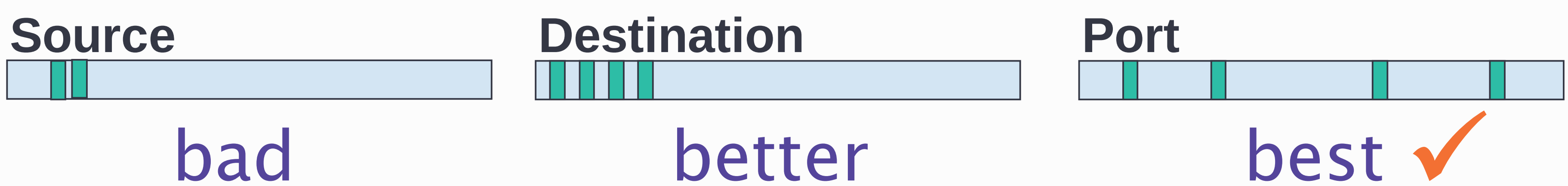


- Each Node covers a distinct hyper-region.
- At each step the search space is reduced by cutting a node (across *k*-dimensions).
- All child-nodes of the same parent cover non-overlapping hyper-regions of same size.
- Leaf-Nodes have a small number of rules represented in a list.

Building the HyperCuts decision tree Step 1: Selecting the Dimensions

- **Challenge:**
  - To pick the dimensions which will lead to the most uniform distribution of the rules when the node is cut into sub-nodes.

- **Idea:**
  - Pick dimensions with highest entropy.



Recall: cuts are equal size ranges for easy array indexing!

Building the HyperCuts decision tree Step 2: Selecting the # of cuts

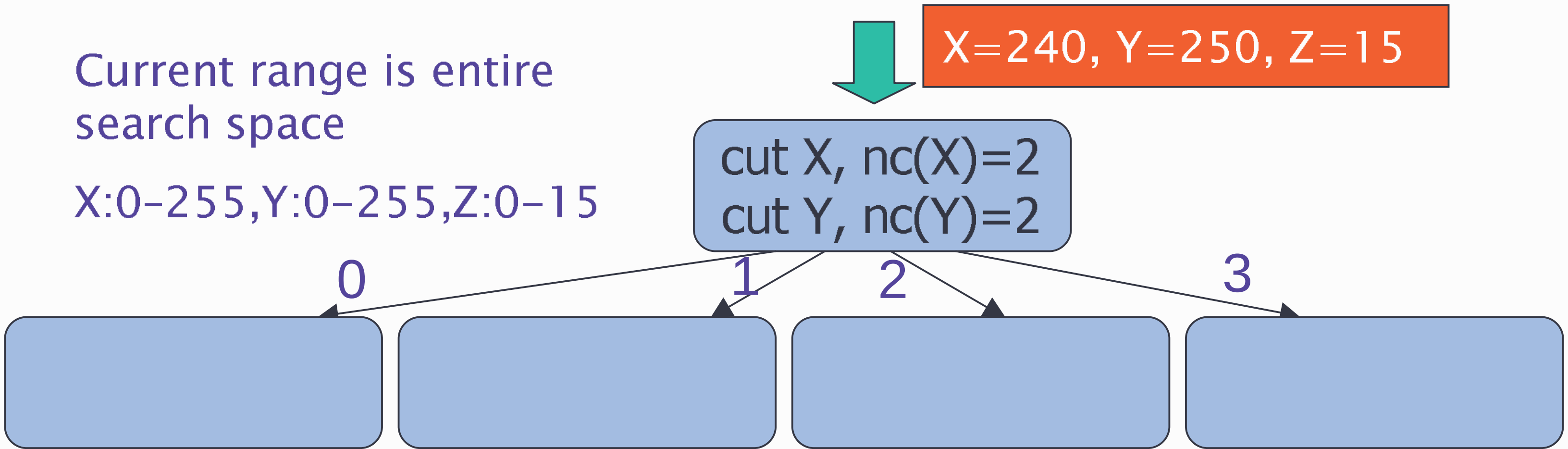
- Goal 1: Minimize search time while keeping space roughly linear
- Strategy 1: Look for multi-dimensional cut that:
  - Minimizes number of rules allocated to any child node
  - Maximum number of Children (cuts) allocated to a node are limited by (space factor * $\sqrt{\#\text{rules in node}}$).

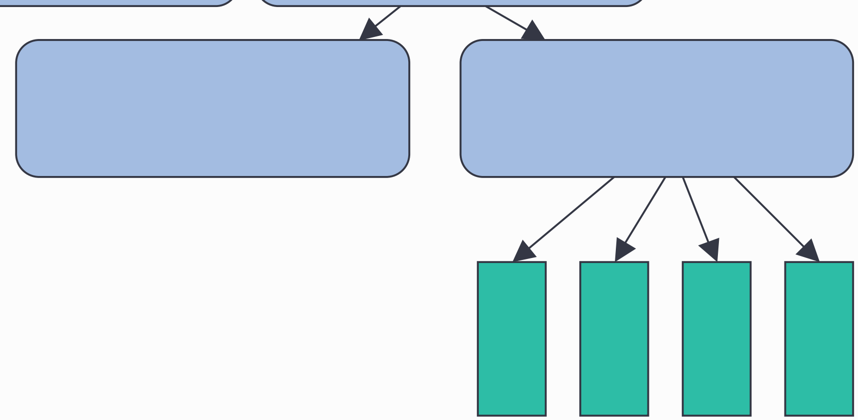
- Goal 2: Avoid exponential time to create a good decision tree
- Strategy 2: Use a greedy strategy which:
  - Determines the optimal cut in each dimension
  - Considers only combinations of these locally optimal cuts

Search algorithm for a HyperCuts decision tree

Current range is entire search space

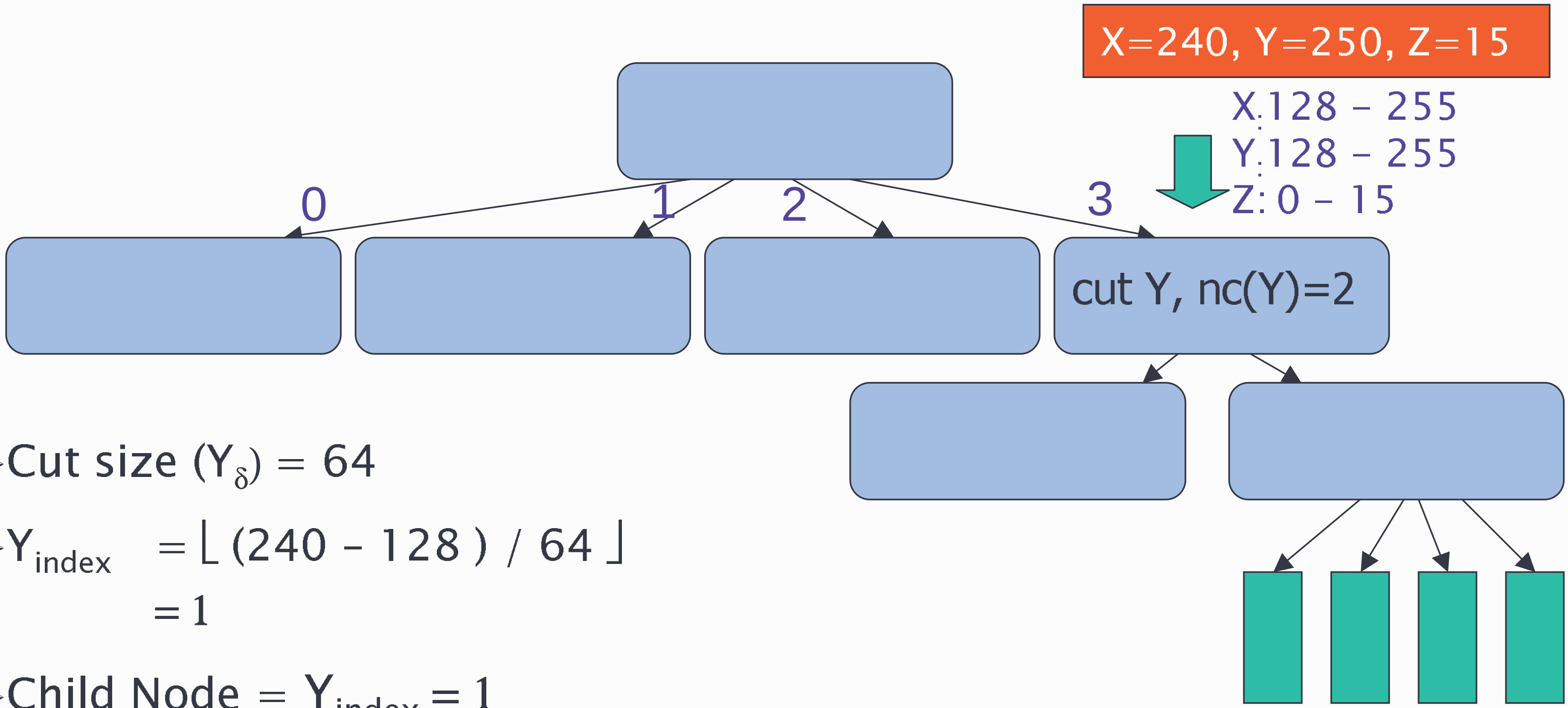
X:0-255,Y:0-255,Z:0-15





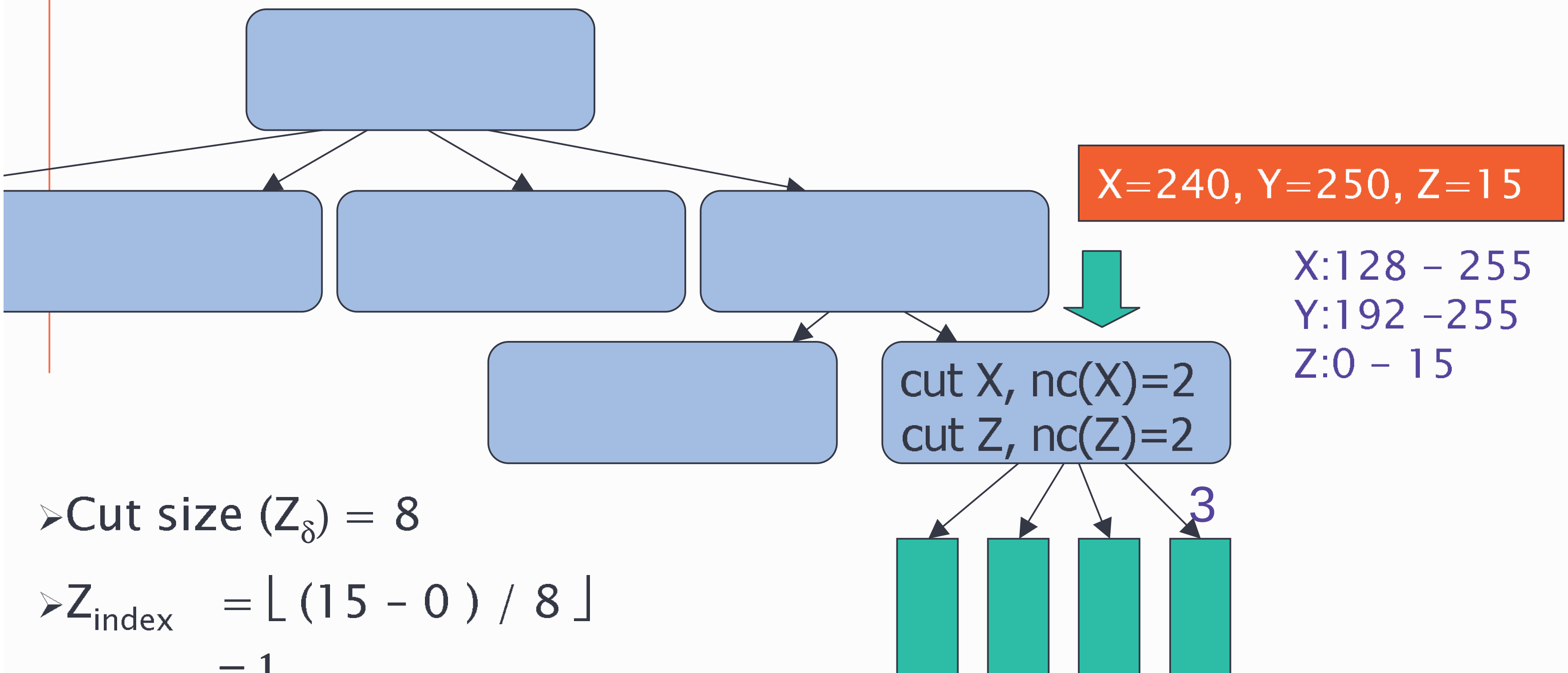
- X_δ = 128 (cut size in X dimension)
- $X_{index} = \lfloor (X_{header} - X_{min}) / X_\delta \rfloor$
  $= \lfloor (240 - 0) / 128 \rfloor$
  $= 1$
- $Y_{index} = \lfloor (250 - 0) / 128 \rfloor = 1$
- Child Node = $Y_{index} * nc(Y) + X_{index}$
  $= (1 * 2) + 1 = 3$

Search algorithm for a HyperCuts decision tree



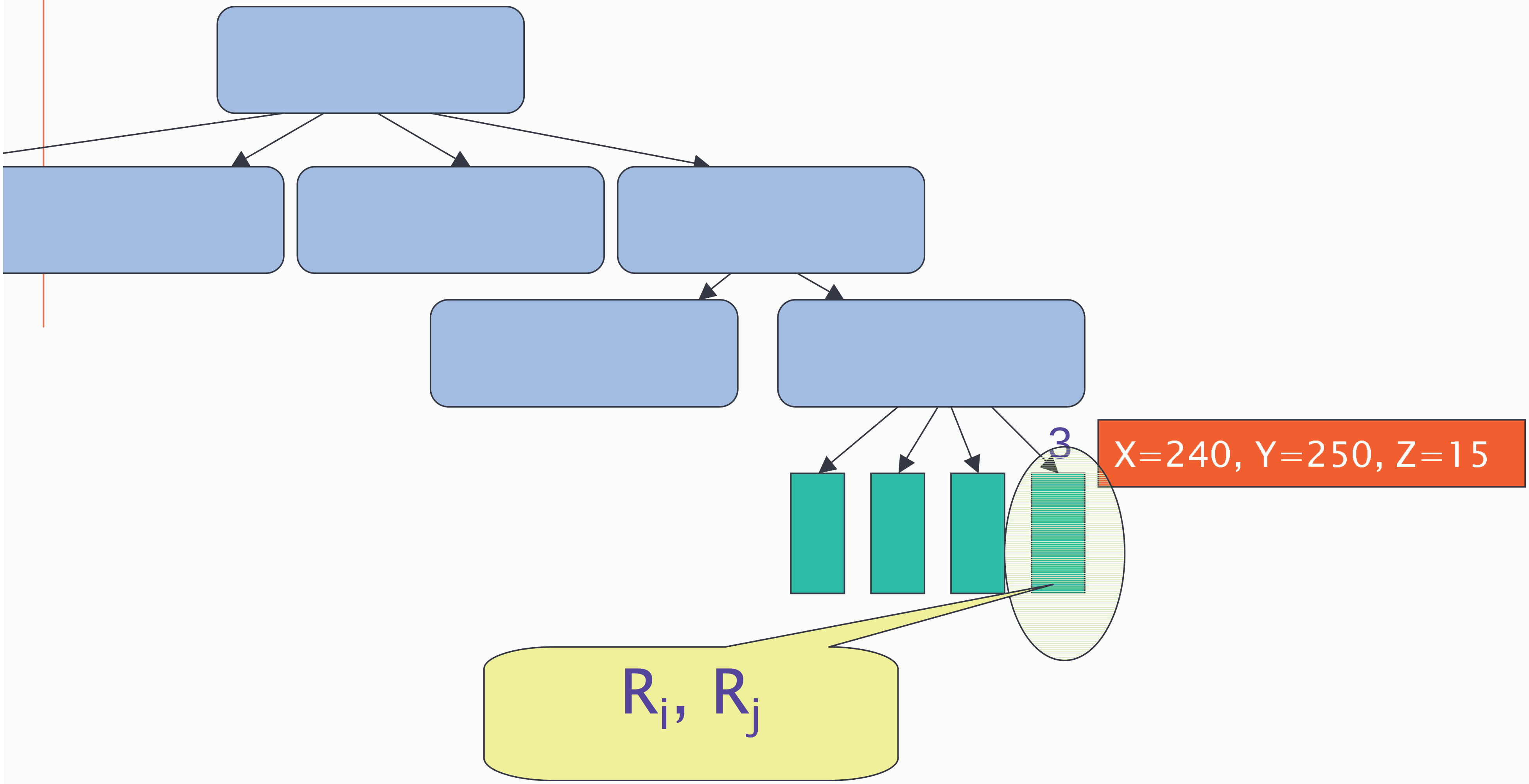
- Cut size (Y_δ) = 64
- Y_{index} $= \lfloor (240 - 128) / 64 \rfloor$
  $= 1$
- Child Node = $Y_{index} = 1$

Search algorithm for a HyperCuts decision tree



- Cut size (Z_δ) = 8
- $Z_{index} = \lfloor (15 - 0) / 8 \rfloor$
  $= 1$
- $X_{index} = \lfloor (240 - 128) / 64 \rfloor = 1$
- Child Node = $Z_{index} * nc(Z) + X_{index}$
  $= (1 * 2) + 1 = 3$

Search algorithm for a HyperCuts decision tree



Outline

1. Introduction
2. Geometric View of Packet Classification
3. Basic Decision Tree Approaches
4. Basic HyperCuts
5. HyperCuts Optimizations
6. Experimental Results
7. Conclusion

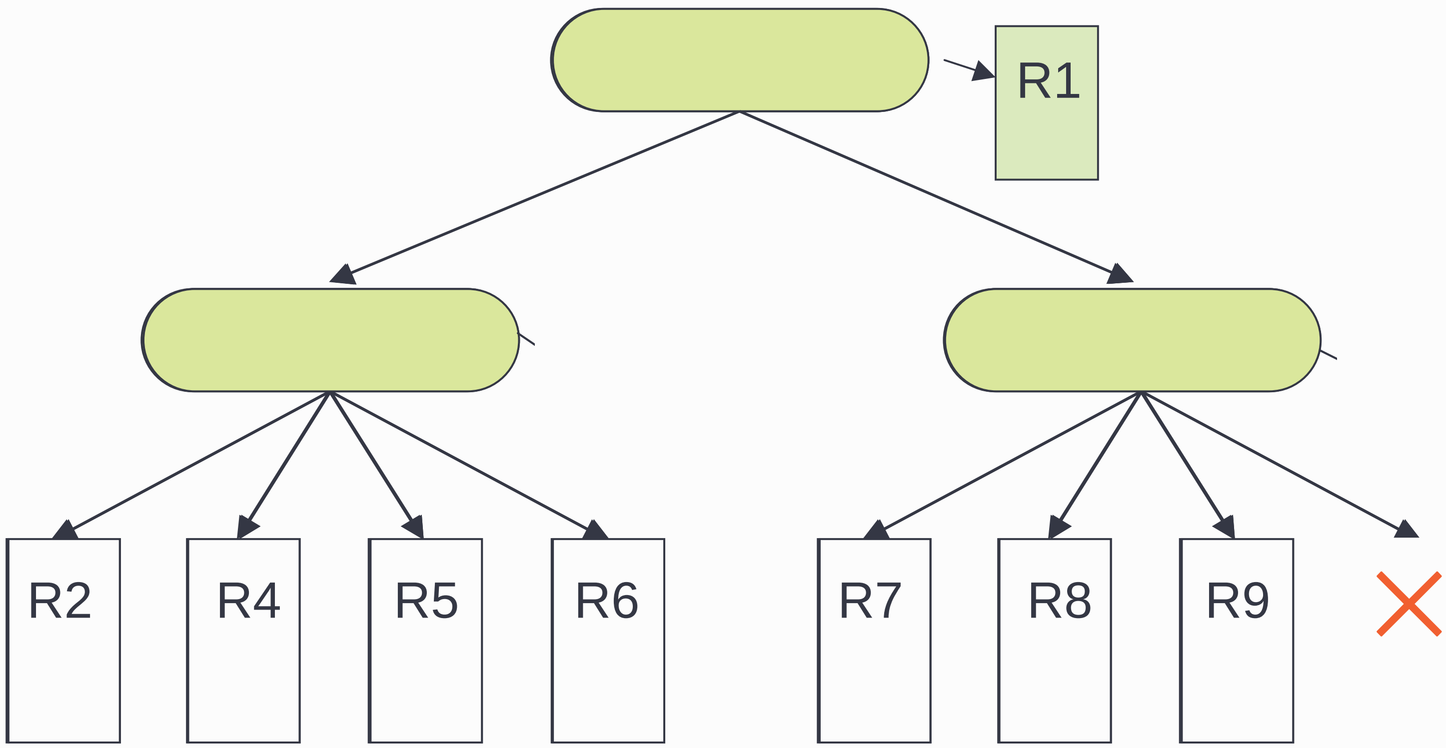
Optimizations for Space Reduction

- Two sources of memory wastage in basic HyperCuts

  – Space consumed by multidimensional arrays.
    Solutions: Node merging, Region compaction

  – Space consumed by *replicated rules*.
    Solutions: Eliminate Rule overlap, Rule Pushing

Rule Pushing



- Rule R1 exists in all child-nodes
- Push-up rule R1 to parent node
- Wild carded rules often get replicated like this.

Outline

1. Introduction
2. Geometric View of Packet Classification
3. Basic Decision Tree Approaches
4. Basic HyperCuts
5. HyperCuts Optimizations
6. Experimental Results
7. Conclusion

Evaluation Methodology

- Metrics:
  - Worst case search time in number of memory accesses
  - Memory size

- Real & Synthetic Classifiers:
  - Core routers (real from multiple Tier-1 ISPs)
  - Edge routers
  - Firewalls

Notes:

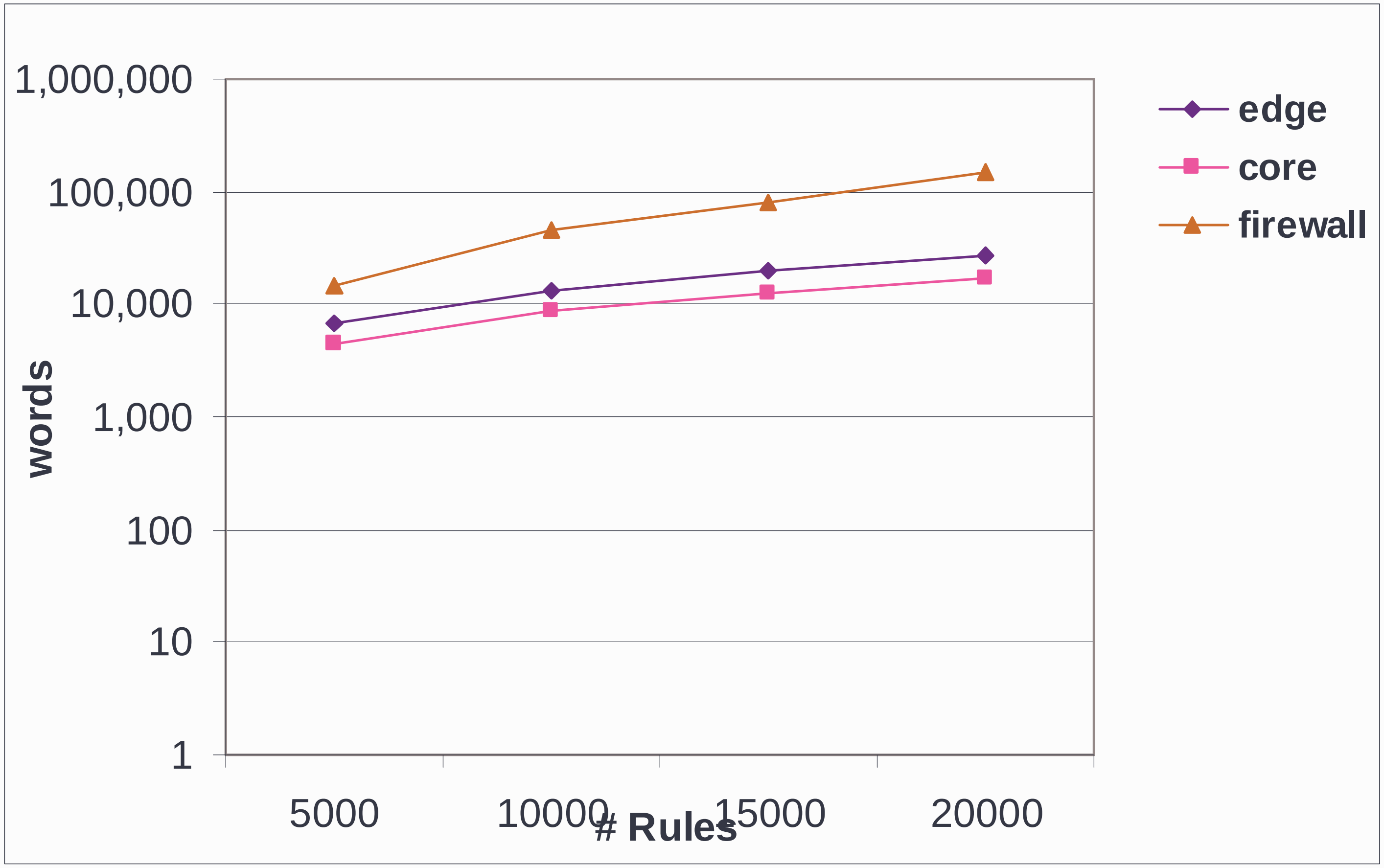
Each rule in the classifiers is a 5 Tuple:

Source Prefix, Destination Prefix, Source Port, Destination Port, Protocol

Evaluation
Real Classifiers

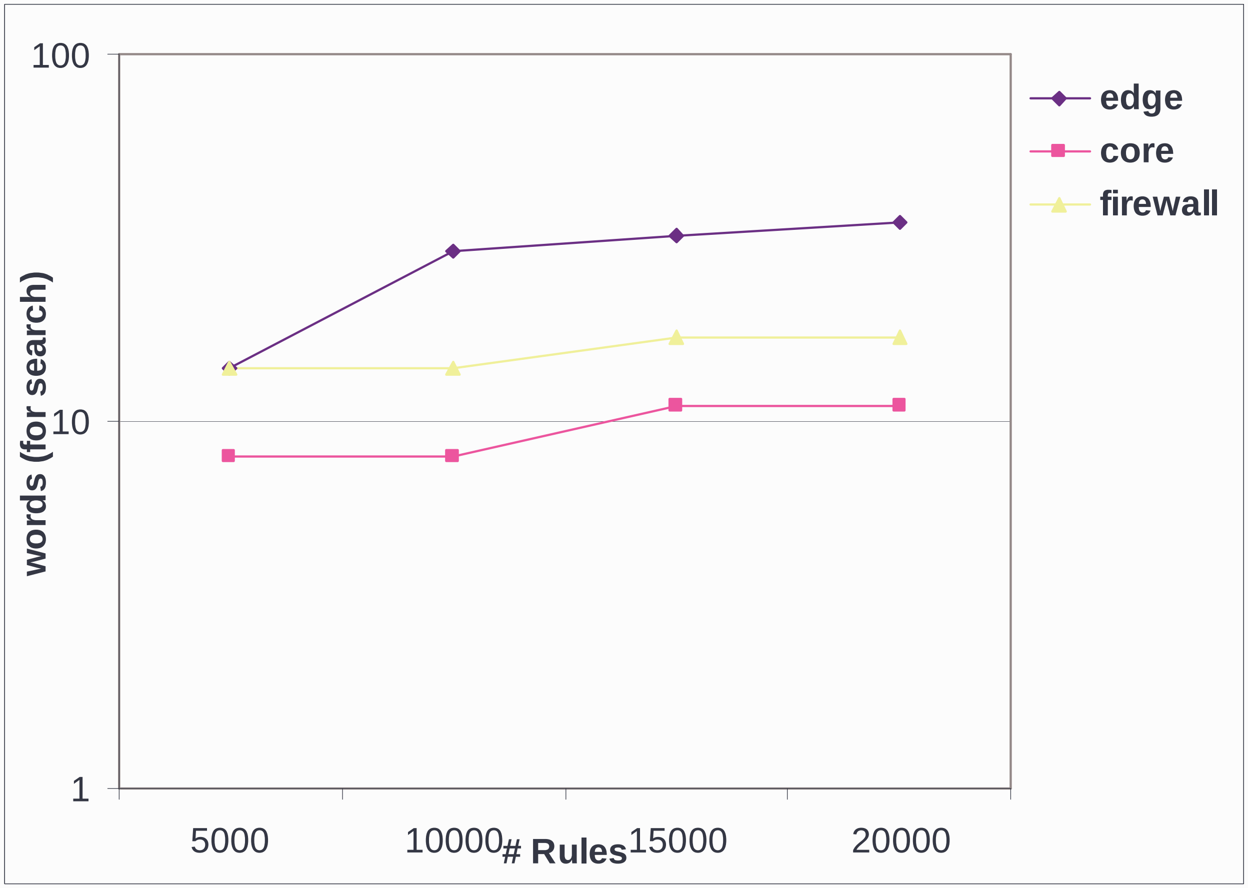
- HyperCuts optimized for memory has 50–500% better search time than HiCuts optimized for speed.
- HyperCuts optimized for speed uses 2 to 10 times less memory than HiCuts optimized for memory.
- Compared with other algorithms (e.g. RFC) for a database of 2800 rules HyperCuts uses 30 times less memory space, while the search speed decreases only by a factor of 50%.

Evaluation
Synthetic classifiers (memory)



- Memory utilization grows linearly with increase in number of rules

Evaluation
Synthetic Classifiers (search)



- Search time does not grow worse than logarithmically

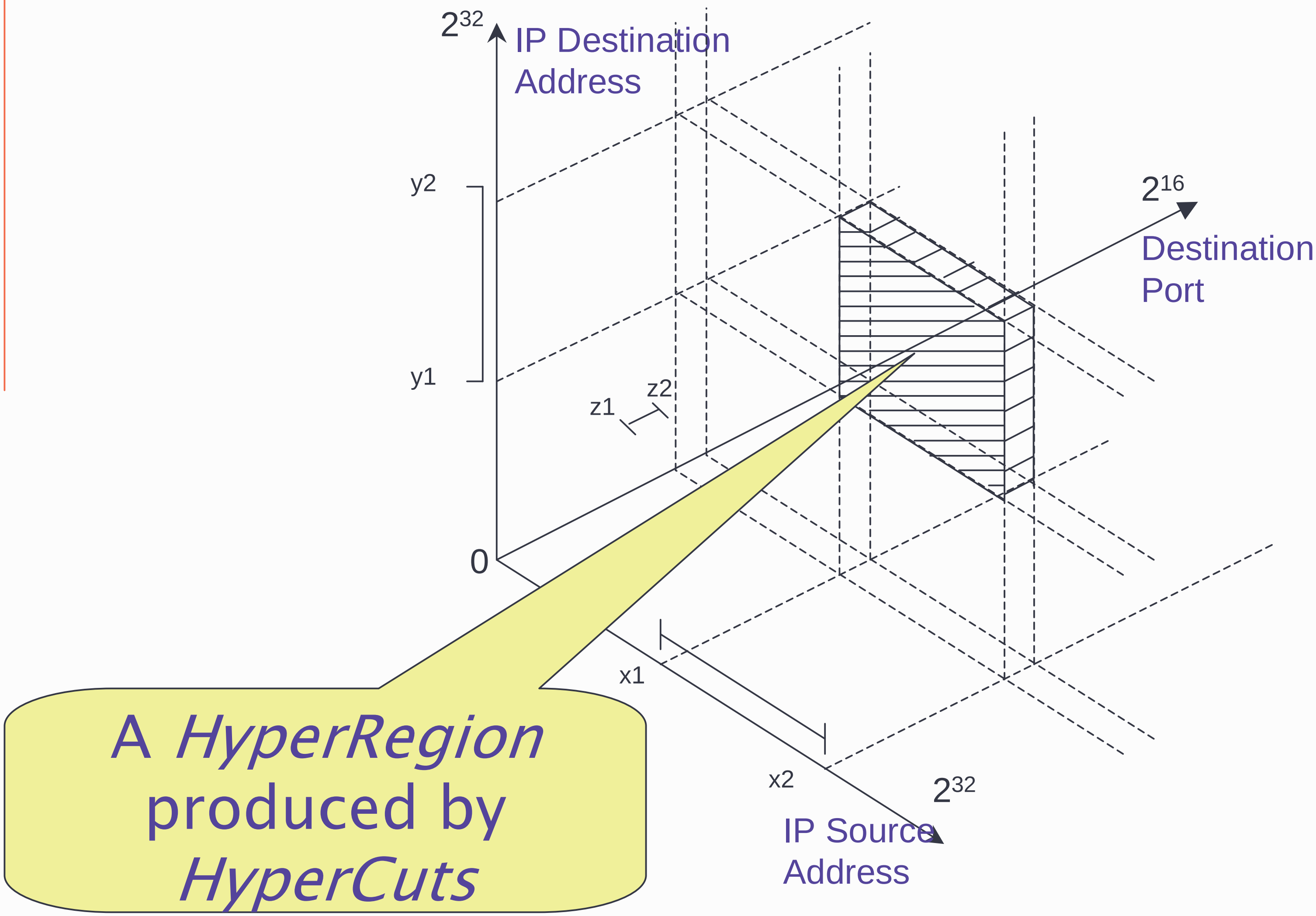
A word of caution

- Classifier characteristics differ between locations and between ISPs (Firewall, Edge, Core Router)
- Cutting across multiple dimensions in each step may not be a good idea:
  - Lose flexibility of adaptive decisions
- For 2-d classifers HyperCuts degenerates to HiCuts for best performance (i.e. select at most 1 dimension at every step)

Conclusion

- HyperCuts has linear space complexity and provides a latency that is at most logarithmic in the number of rules on real classifiers that we studied.
- The throughput of the algorithm can be improved by pipelining based on the depth of the tree.
- Based on initial evaluation, It seems that HyperCuts can be a practical contender compared to CAM based solutions.
- Future Direction:
  We have designed a pipeline architecture for hardware implementation of the algorithm, which we are evaluating.

Questions ?

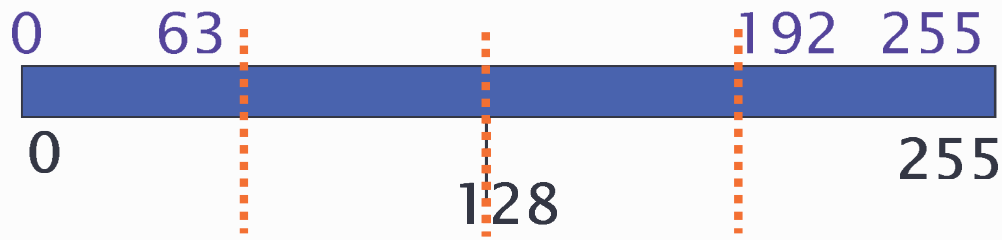


2^{32}
IP Destination Address
y2
y1
z1
z2
2^{16}
Destination Port
0
x1
x2
2^{32}
IP Source Address
A *HyperRegion* produced by *HyperCuts*

Decision Tree Based Algorithms

- **Idea**:
  - build a decision tree based on local optimization decisions at each node
- **Pros:**
  - Tree can be of relatively small height
  - Easy to pipeline
- **Cons:**
  - Difficult to predict the performance
  - Utilizing fancy heuristics and optimizations may
    - Increase search latency
    - Increase complexity of incremental updates.

What is a Cut?



4 cuts, cut size= 64