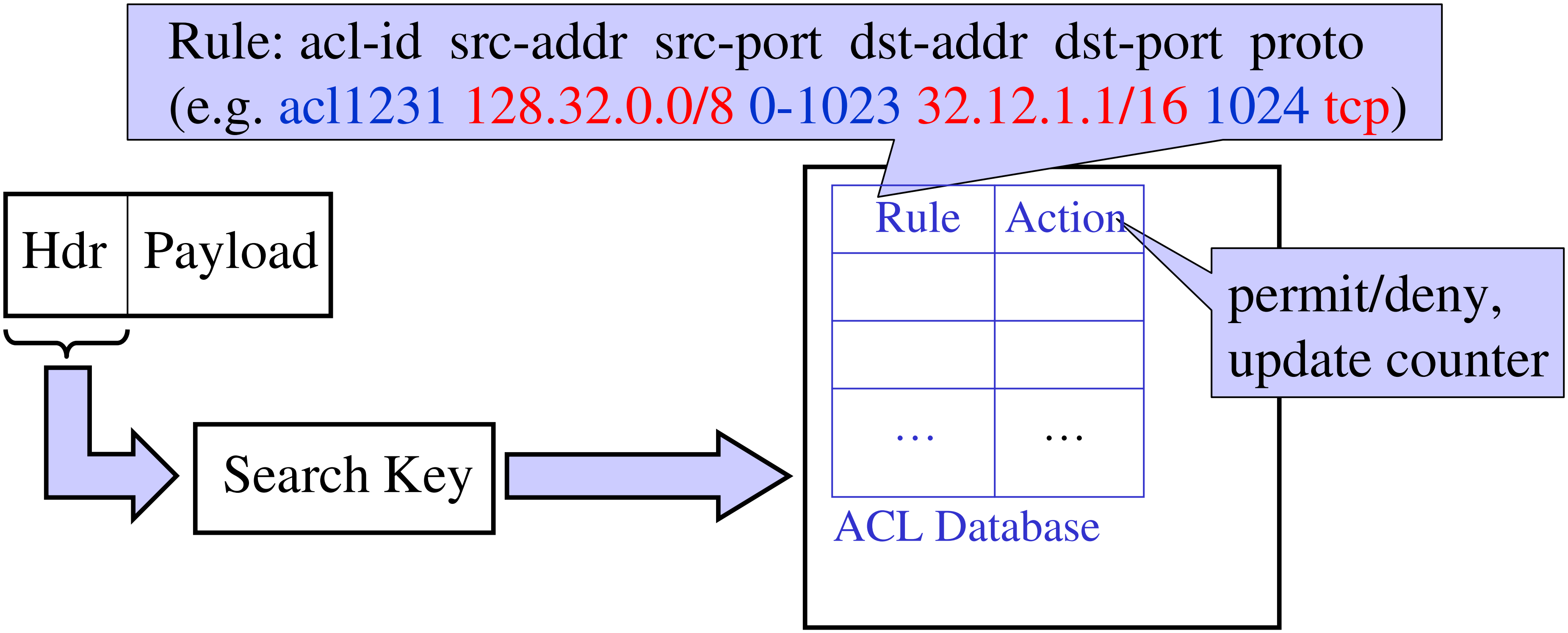
Algorithms for Advanced Packet Classification with Ternary CAMs

Karthik Lakshminarayanan
UC Berkeley

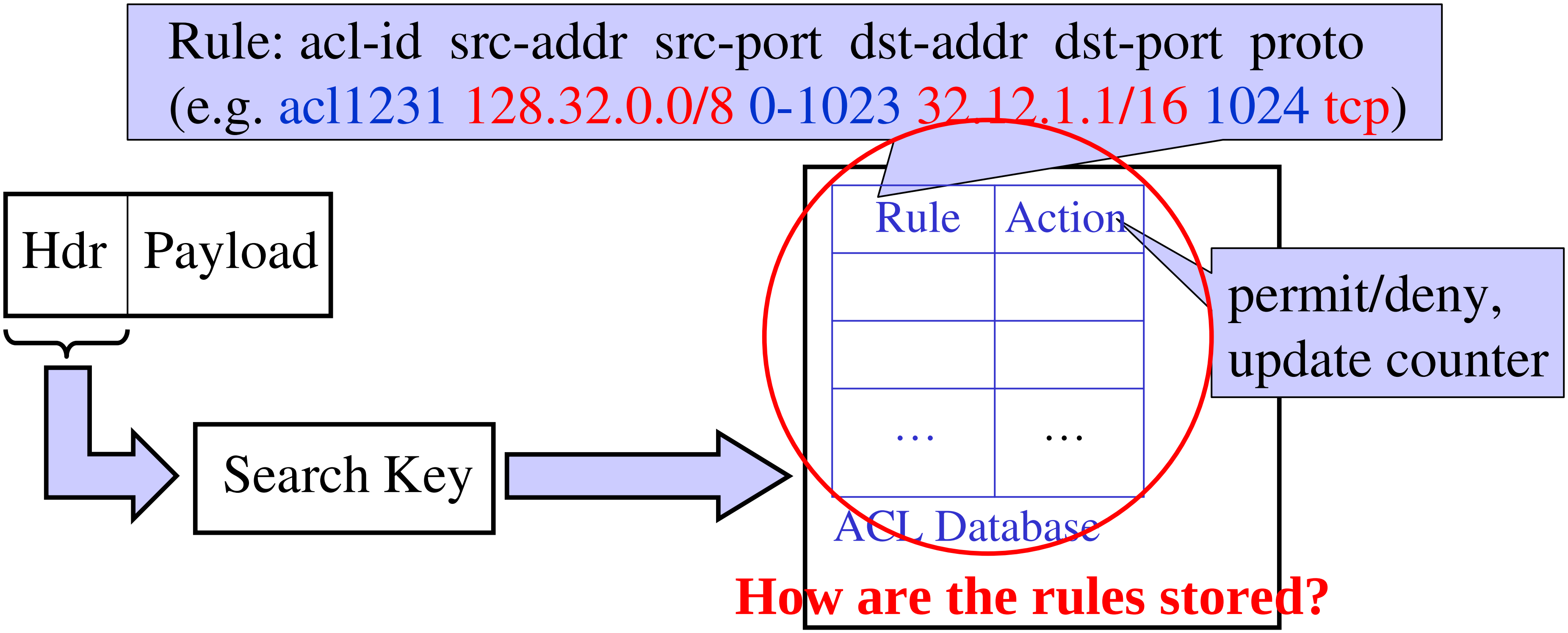
Joint work with
Anand Rangarajan and Srinivasan Venkatachary
(Cypress Semiconductor)

Packet Processing Environment



- Packet matches a set of rules based on the header
- Examples: routers, intrusion detection systems

Packet Processing Environment



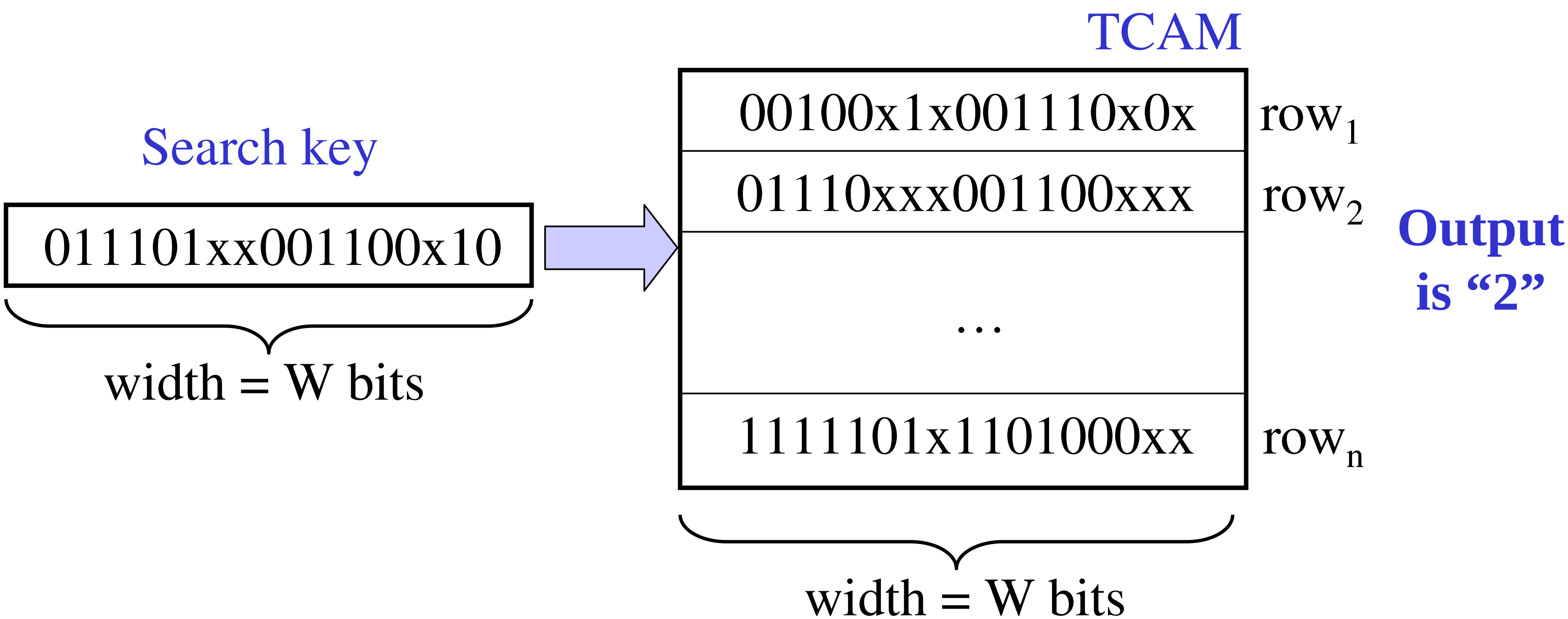
- TCAMs gaining widespread deployment
  - 6 million TCAM devices deployed
  - Used in multi-gigabit systems that have O(10,000) rules

Ternary Content Addressable Memory

- **RAM: input = address, output = value**
- **CAM: input = value, output = address**

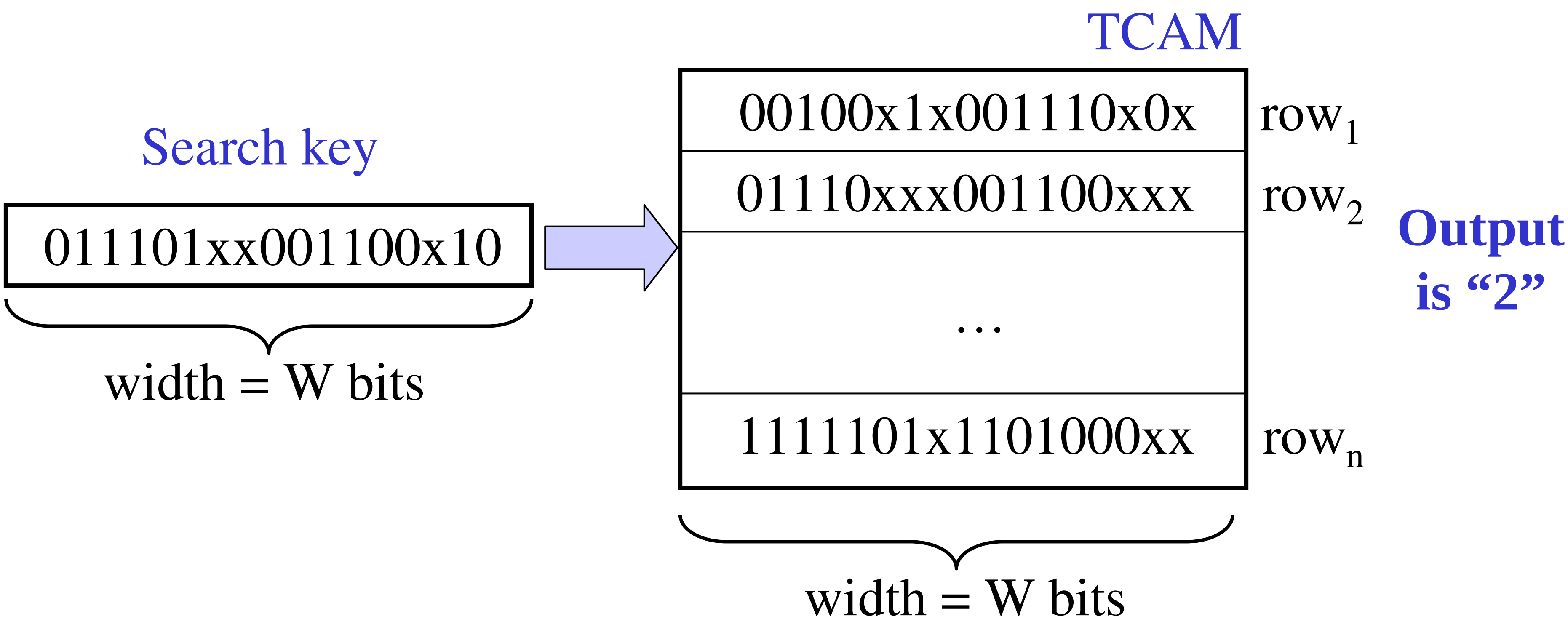
Ternary Content Addressable Memory

- Memory device with fixed-width arrays
- Each bit is 0, 1 or x (don't care)
- Search is performed against all entries in *parallel* and the *first result* is returned



Ternary Content Addressable Memory

- Benefits: Deterministic Search Throughput
  - single cycle search irrespective of search key



Problems

- Range Representation Problem

- Multimatch Classification Problem

No modifications to TCAMs and simple
→ Easy to deploy

Problems

- Range Representation Problem
- Multimatch Classification Problem

Range Representation Problem

- (Recall that rules contain prefixes and ranges)
- Representing prefixes in ternary is trivial
  - IP address prefixes present in rules
  - e.g. 128.32.136.0/24 would contain 8 ‘x’s at the end
- Representing arbitrary ranges is not easy though
  - port fields might contain ranges
  - e.g. some security applications may allow ports 1024-65535 only

Problem Statement: Given a range R, find the minimum number of ternary entries to represent R

Why is efficient range representation an important problem?

Statistic	1998 database	2004 database
Total number of rules	41190	215183
With single range field	4236 (10.3%)	54352 (25.3%)
With single non-"$\geq$ 1024" range field	553 (1.3%)	25311 (11.8%)
With two range fields	0 (0%)	3225 (1.5%)
Unique ranges in first field	62	270
Unique ranges in second field	0	37

Number of range rules has increased over time

Why is efficient range representation an important problem?

Statistic	1998 database	2004 database
Total number of rules	41190	215183
With single range field	4236 (10.3%)	54352 (25.3%)
With single non-"$\geq$ 1024" range field	553 (1.3%)	25311 (11.8%)
With two range fields	0 (0%)	3225 (1.5%)
Unique ranges in first field	62	270
Unique ranges in second field	0	37

Number of unique ranges have increased over time

Earlier Approaches – I

Prefix expansion of ranges:

- express ranges as a union of prefixes
- have a separate TCAM entry for each prefix

- Example: the range [3,12] over a 4-bit field would expand to:
  - 0011 (3), 01xx (4-7), 10xx (8-11) and 1100 (12)
  - expansion: the number of entries a rule expands to
- Worst-case expansion for a W-bit field is 2W-2
  - example: [1,14] would expand to 0001, 001x, 01xx, 10xx, 110x, 1110
  - 16-bit port field expands to 30 entries

Why is efficient range representation an important problem?

Statistic	1998 database	2004 database
Total number of rules	41190	215183
With single range field	4236 (10.3%)	54352 (25.3%)
With single non-"≥ 1024" range field	553 (1.3%)	25311 (11.8%)
With two range fields	0 (0%)	3225 (1.5%)
Unique ranges in first field	62	270
Unique ranges in second field	0	37

Two range fields – multiplicative effect

Earlier Approaches – II

Database-dependent encoding:

- observation: TCAM array has some unused bits
- use these additional bits to encode commonly occurring ranges in the database

- TCAMs with IP ACLs have ~ 36 extra bits
  - 144-bit wide TCAMs
  - 104-bits + 4-bits typically used for IP ACL rules

Earlier Approaches – II

Database-dependent encoding:

– observation: TCAM array has some unused bits

– use these additional bits to encode commonly occurring ranges in the database

- Example:

Address	Port	…		
12.123.0.0/16	20-24	…	→	**Set extra bit to 1**
32.12.13.0/24	1024-	…	→	**Set extra bit to x**
128.0.0.0/8	20-24	…	→	**Set extra bit to 1**

If search key falls in 20-24, set extra bit to 1, else set it to 0

Earlier Approaches – II

Database-dependent encoding:

- observation: TCAM array has some unused bits
- use these additional bits to encode commonly occurring ranges in the database

- Improved version: Region-based Range Encoding
- Disadvantages:
  - database dependent → incremental update is hard

Database-Independent Range Pre-Encoding (DIRPE)

- Key insight: use additional bits in a database independent way
  - wider representation of ranges
  - reduce expansion in the worst-case

DIRPE: Fence Encoding

- Fence encoding (W-bit field)
  - total of 2^W-1 bits
  - Encoding(0) = **0000000**
  - Encoding(2) = **0000011**
  - Encoding(4) = **0001111**
  - Encoding[2,4] = **000xx11**

Range	Encoding
$= i$	$0^{2^k-i-1}1^i$
$\geq i$	$x^{2^k-i-1}1^i$
$< i$	$0^{2^k-i}x^{i-1}$
$[i, j]$	$0^{2^k-1-j}x^{j-i}1^i$

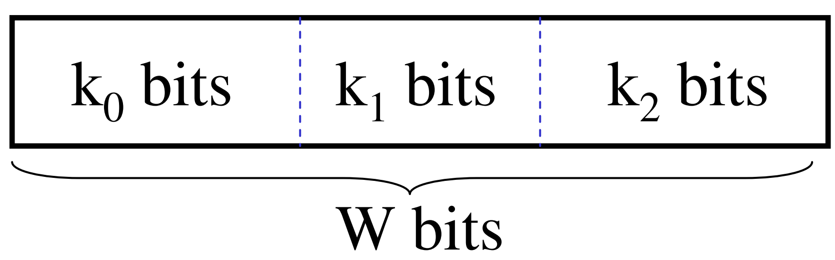
- Using 2^W-1 bits, fence encoding achieves an expansion of 1
- Theorem: For achieving a worst-case row expansion of 1 for a W-bit range, 2^W-1 bits are necessary

DIRPE: Using the Available Extra Bits

- Two extremes:
  - no extra bits → worst case expansion is 2W–2
  - 2^W–W–1 extra bits → worst case expansion is 1
- Is there something in between?
  - appropriate worst-case based on number of extra bits available

DIRPE: Splitting the Range Field

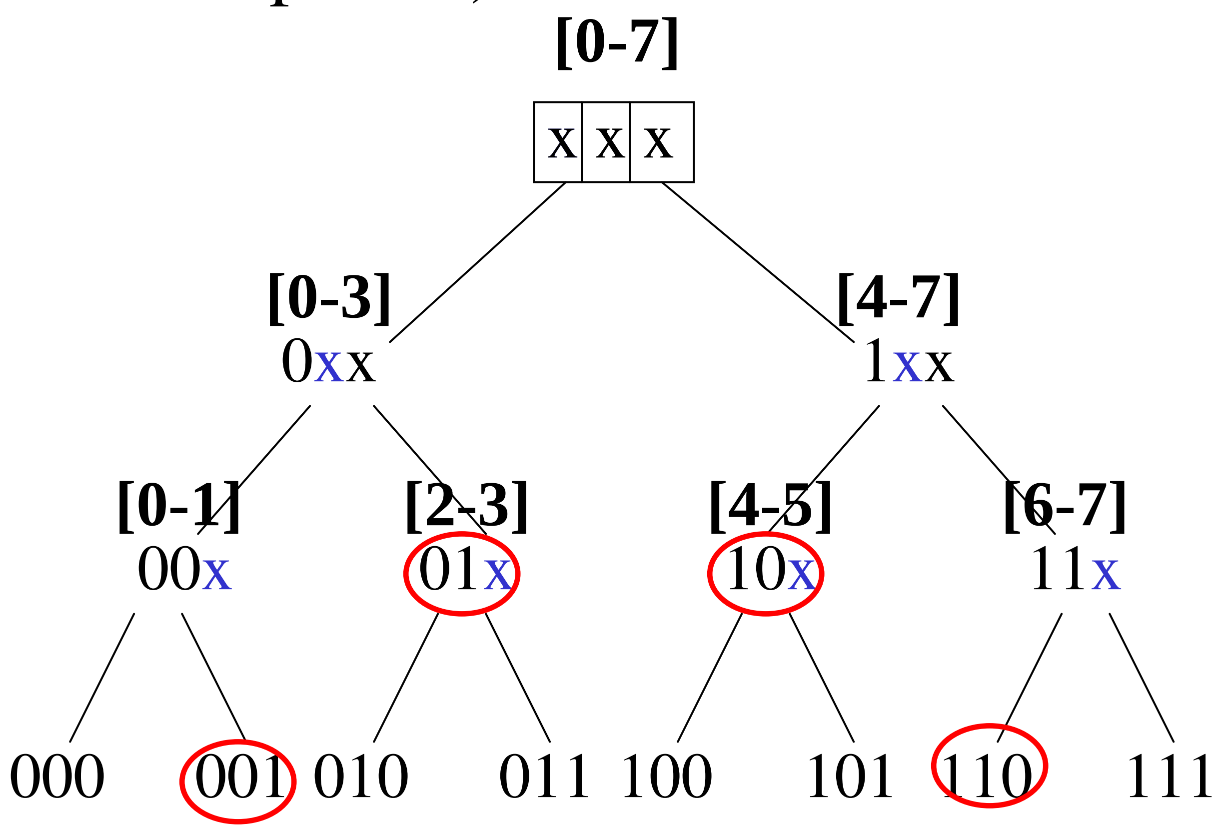
- Procedure:
  - split W-bit field into multiple *chunks*
  - encode each chunk using fence encoding
  - “combine” the chunks to form ternary entries



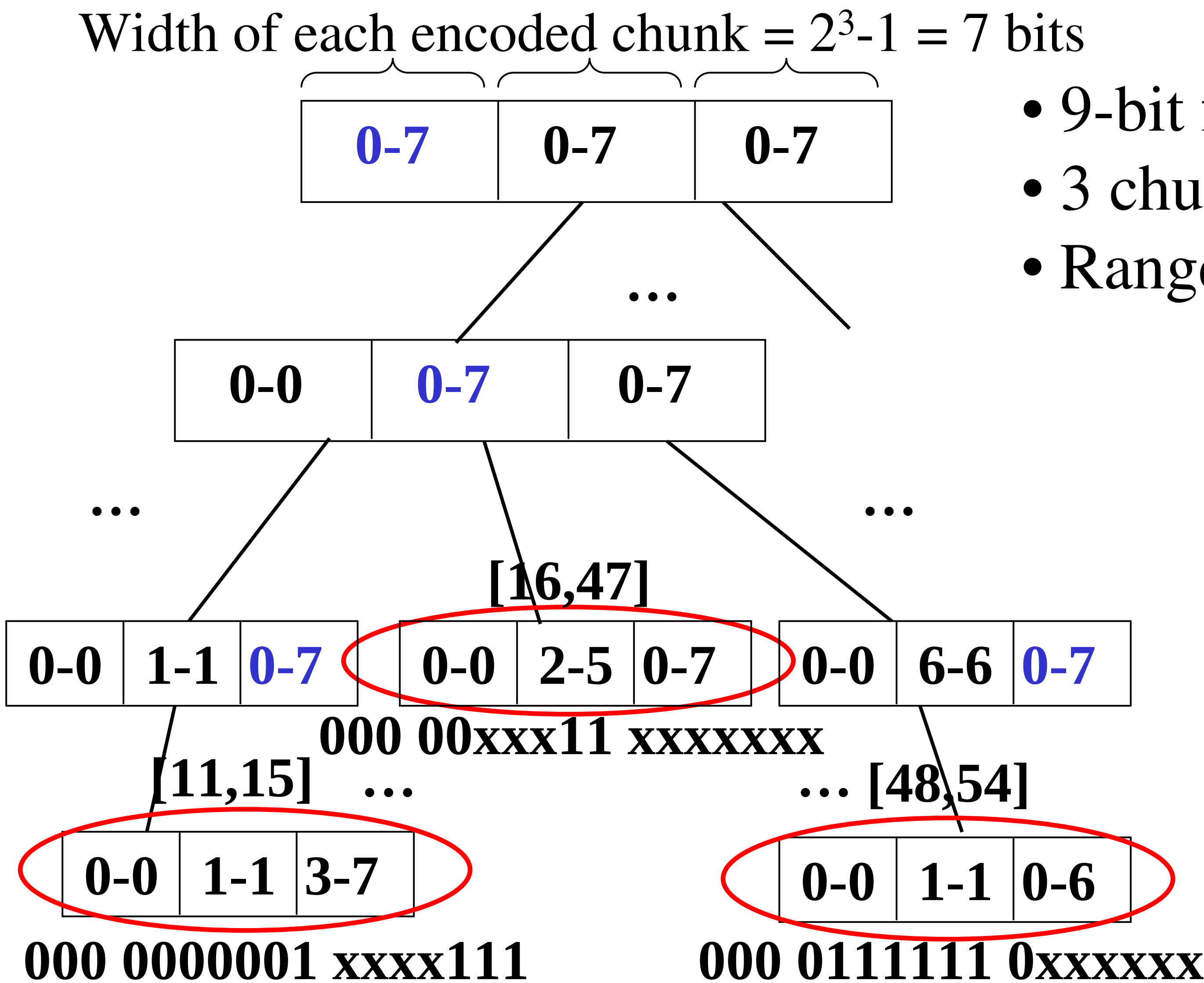
Combining chunks: analogous to multi-bit tries

Unibit view of DIRPE (Prefix expansion)

- W=3, split into three 1-bit chunks; Range=[1,6]
- Each level can contribute to at most 2 prefixes (but for the top level)



Multi-bit view of DIRPE



- 9-bit field (W=9)
- 3 chunks, 3 bits wide
- Range = [11,54]
  = [013, 066]

Worst case expansion
$= 2W/k - 1$

Number of extra bits needed
$= (2^k-1)W/k - W$

Comparison of Expansion

Extra bits	DIRPE	Region-based Range Encoding
0	30	30
8	15	30
18	11	16
27	9	14
44	7	12

Worst-case expansion

Extra bits	DIRPE	Region-based Range Encoding
0	2.69	2.69
8	2.08	2.33
18	1.79	2.17
36	1.57	1.58

Real-life expansion

Metric	Prefix Expansion	Region-based Encoding (with *r* regions)	DIRPE (with *k*-bit chunks)	DIRPE + Region-based
Extra bits	0	$F(\log_2 r + \frac{2n-1}{r})$	$F(\frac{W(2^k-1)}{k} - W)$	$F(\frac{(2^k-1)\log_2 r}{k} + \frac{2n-1}{r})$
Worst-case capacity degradation	$(2W-2)^F$	$(2\log_2 r)^F$	$(\frac{2W}{k} - 1)^F$	$(\frac{2\log_2 r}{k})^F$
Cost of an incremental update	$O(W^F)$	$O(N)$	$O((\frac{W}{k})^F)$	$O(N)$
Overhead on the packet processor	None	Pre-computed table of size: $O((\log_2 r + \frac{2n-1}{r}) F.2^W)$ (*or*) $O(nF)$ comparators of width W bits	$O(\frac{W.2^k}{k})$ logic gates	Both pieces of logic from previous two columns

DIRPE: Summary

↑ Database independent

↑ Scales well for large databases

↑ Good incremental update properties

↓ Additional bits needed

↓ Small logic needed for modifying search key

  ↑ Does not affect throughput

Problems

- Range Expansion Problem
- Multimatch Classification Problem

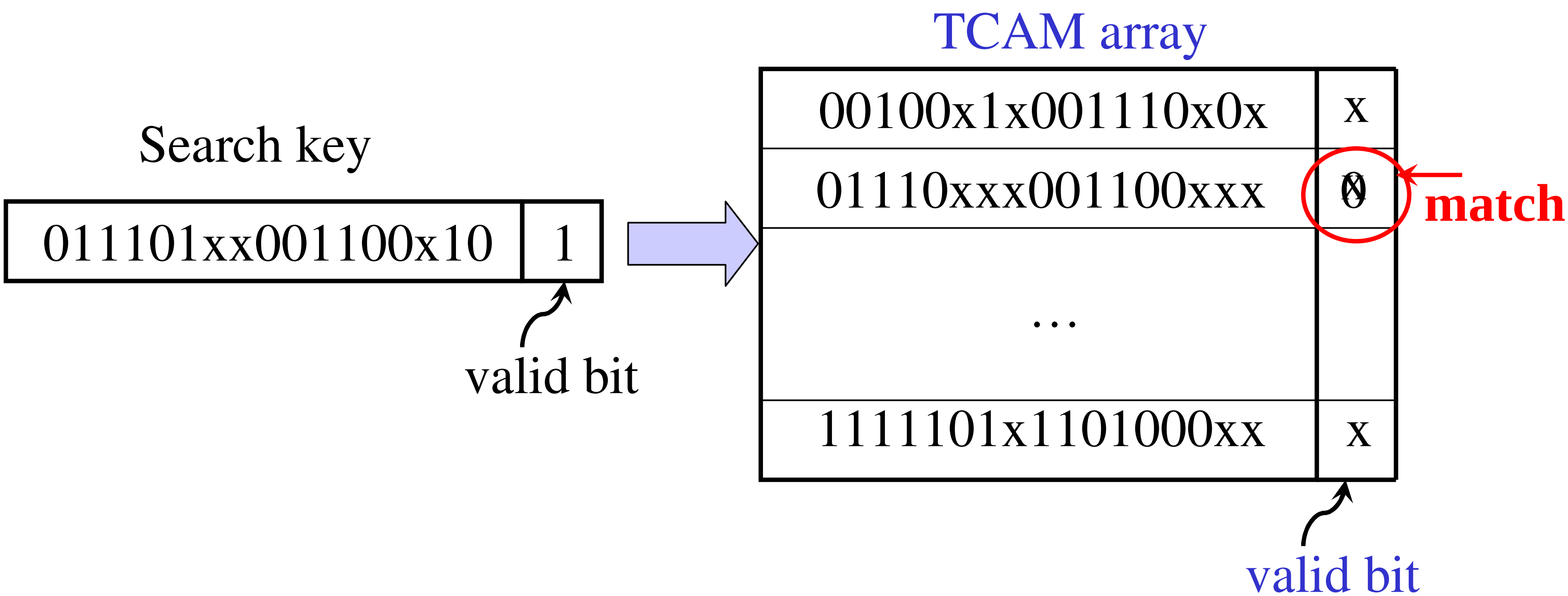
Multimatch Classification Problem

- TCAM search primitive: return first matching entry for a key
- Multimatch requirement: return *k* matches (or all matches) for a key
  - security applications where all signatures that match this packet need to be found
  - accounting applications where counters have to be updated for all matching entries

Earlier Approaches

Entry Invalidation scheme:

- maintain state of multimatch using an additional bit in TCAM called “valid” bit



Earlier Approaches

Entry Invalidation scheme:

– maintain state of multimatch using an additional bit in TCAM called “valid” bit

- Disadvantage:
  – ill-suited for multi-threaded environments

Earlier Approaches

Geometric intersection scheme:

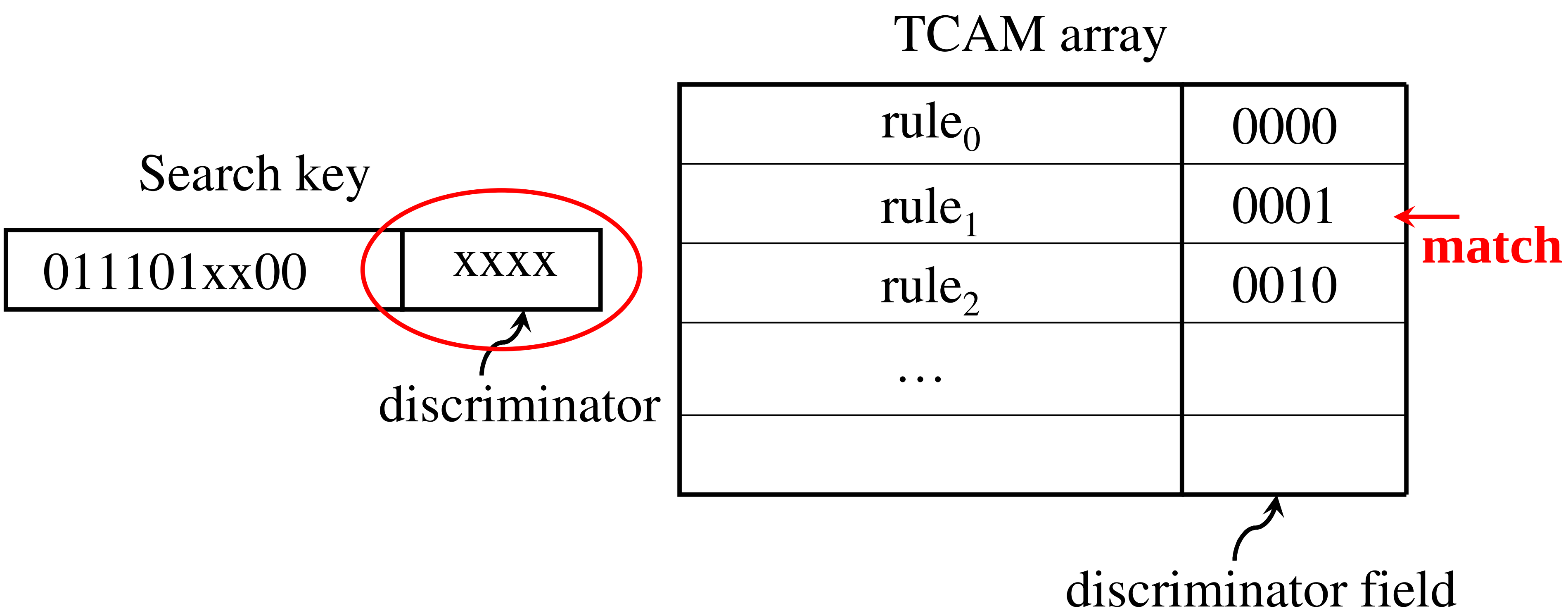
- construct geometric intersection (cross-products) of the fields and place in TCAM
- pre-processing step is expensive
- search is fast

- Disadvantage:
  - does not scale well in capacity
  - for router dataset: expansion of 25—100

Multimatch Using Discriminators (MUD)

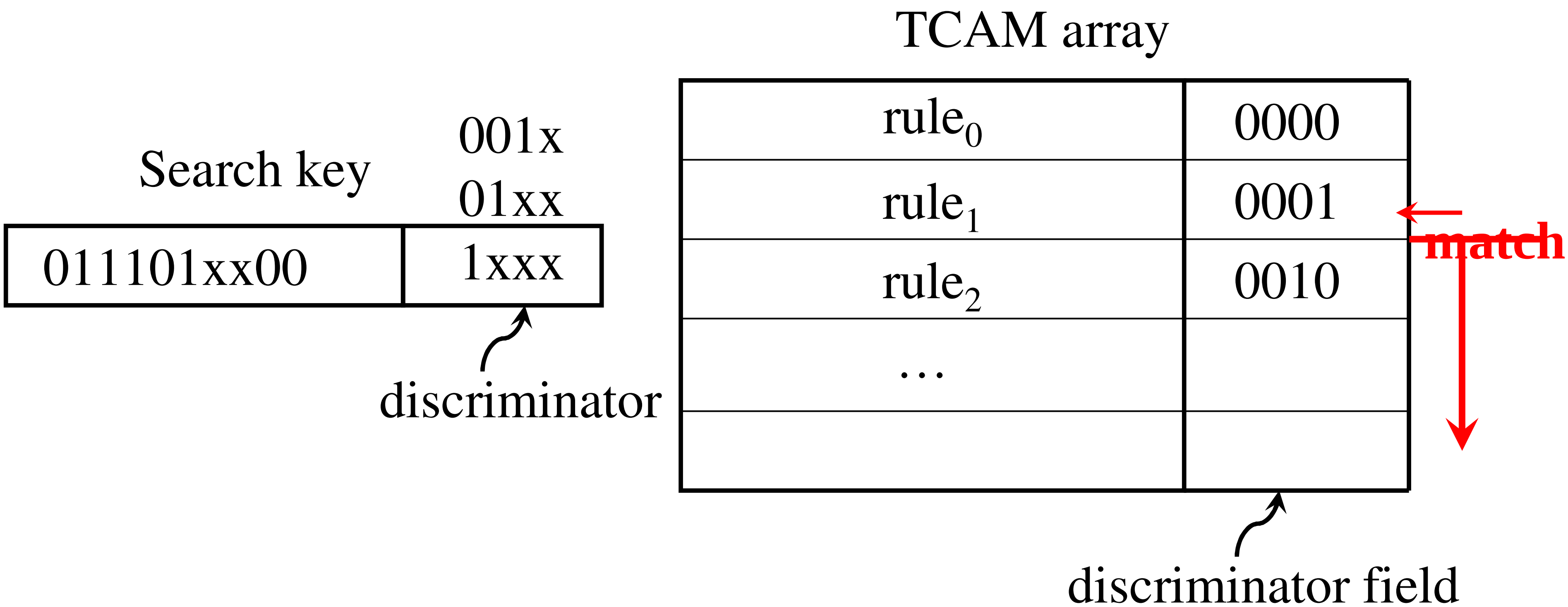
- Observation: after index *j* is matched, the ACL has to be searched for all indices >*j*
- Basic idea:
  - store a discriminator field with each row that encodes the index of the row
  - to search rows with index >*j*, the search key is expanded to prefixes that correspond to >*j*
  - multiple searches are then issued

MUD: Example



TCAM array
Search key
011101xx00
xxxx
discriminator
rule0
rule1
rule2
…
0000
0001
0010
match
discriminator field

MUD: Example


TCAM array
Search key
001x
01xx
011101xx00
1xxx
discriminator
rule0
0000
rule1
0001
rule2
0010
…
match
discriminator field

Metric	Entry Invalidation	Geometric Intersection-based	MUD
Multi-threading support	No	Yes	Yes
Worst-case TCAM entries for N rules	N	$O(N^F)$	N
Update cost	O(N)	$O(N^F)$	O(N)
Cycles for k multi-matches	7k	k	1 + d + (d-1)(k-2) with DIRPE: $1 + \frac{d(k-1)}{r}$
Extra bits	0	0	without DIRPE: d with DIRPE: $\log_2(d/r) + (d-r) + (2^r-1)$
Overhead on the packet processor	Small state machine logic; can be implemented using a few hundred gates or a few microcode instructions	None	Small state machine logic; can be implemented using a few hundred gates or a few microcode instructions

MUD: Summary

- ↑ No per-search state in TCAM — suitable for multi-threaded environments
- ↑ Incremental updates fast
- ↑ Scales well to large databases

- ↓ Additional bits needed
- ↓ Extra search cycles
  - ↑ Can still support Gbps speeds

Conclusion

- Range expansion problem: DIRPE, a database independent range encoding
  - scales to large number of ranges
  - good incremental update properties
- Multimatch classification problem: MUD
  - suitable for multithreaded environments
  - scales to large databases
- No change to TCAM hardware and simple

  → easy to deploy