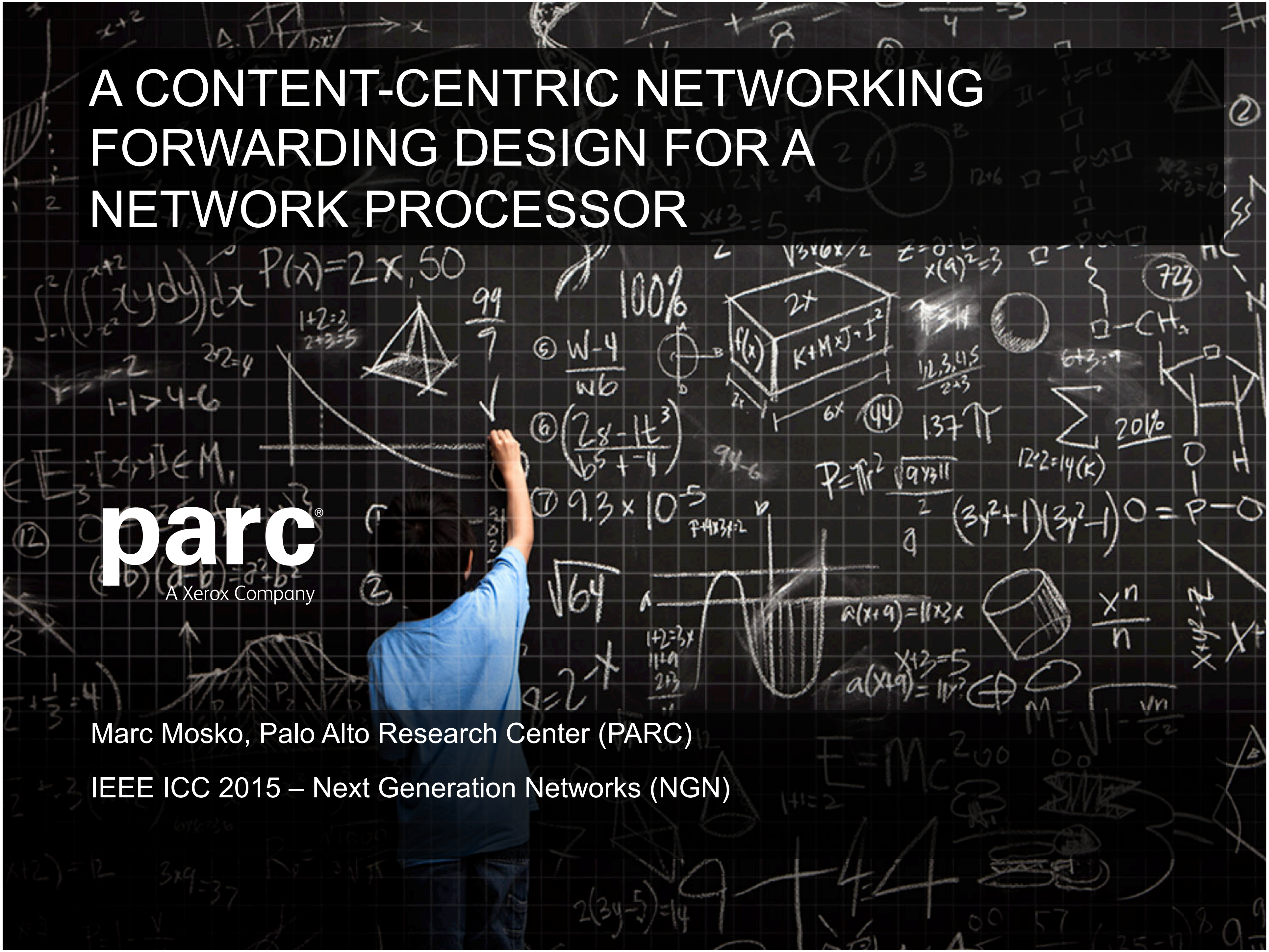
A CONTENT-CENTRIC NETWORKING FORWARDING DESIGN FOR A NETWORK PROCESSOR



parc
A Xerox Company

Marc Mosko, Palo Alto Research Center (PARC)

IEEE ICC 2015 – Next Generation Networks (NGN)

PAPER TOPIC

How to implement
token-by-token Longest Prefix Match
with variable length tokens
using hardware-assisted hash tables

HARDWARE HASH TABLE

Hardware accepts keys of fixed sizes
(e.g. up to 48 bytes)
And stores data of fixed size
(e.g. up to 96 bytes)
Performance degrades with longer sizes

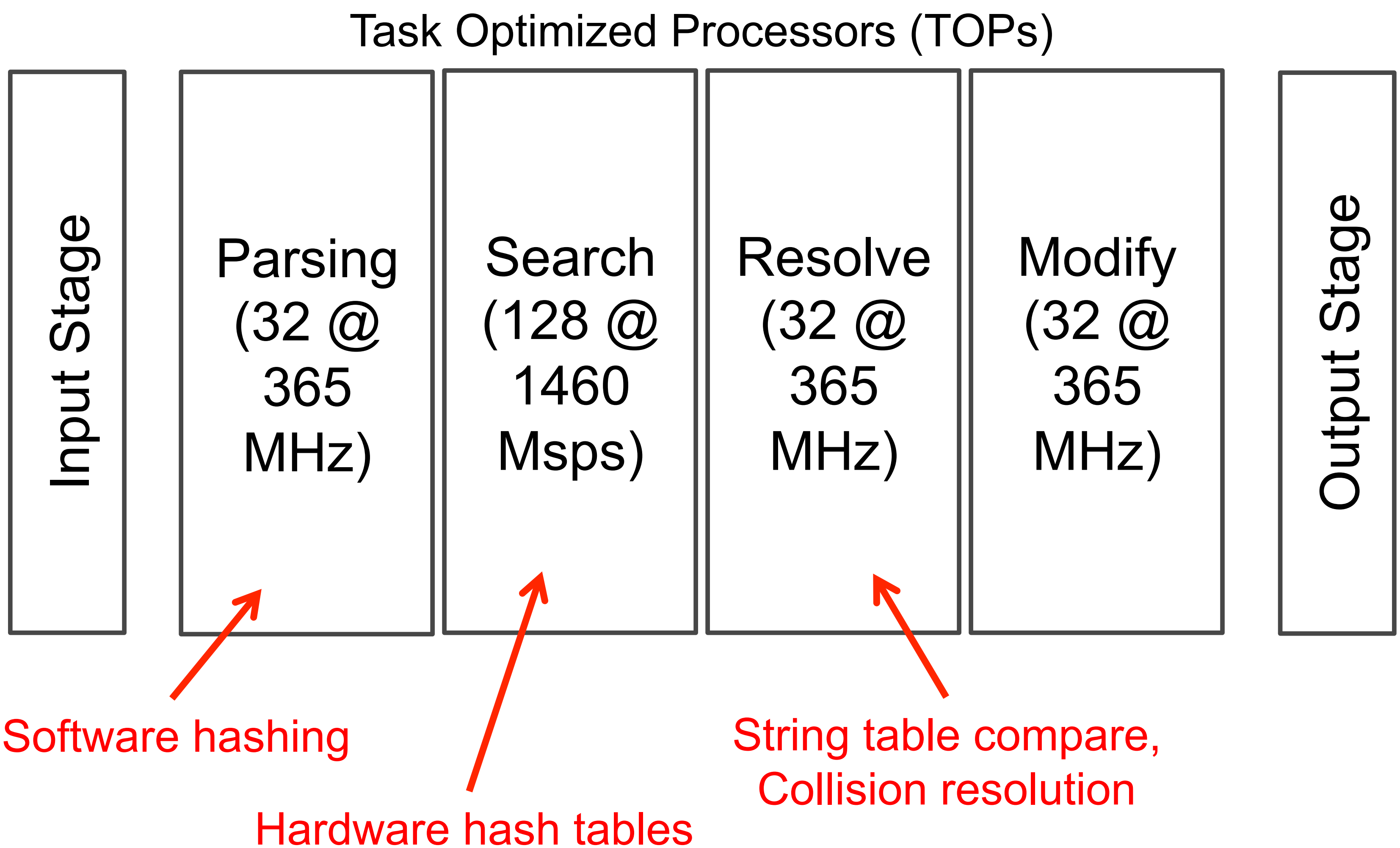
If a name component does not fit in the key, we need to compress key.

RELEVANCE

Implement Content Centric Networking
(CCNx)
On network processors
(EZchip NP4)
In multi-slot chassis switch

parc
A Xerox Company

EZCHIP NP4



Task Optimized Processors (TOPs)
Input Stage
Parsing (32 @ 365 MHz)
Search (128 @ 1460 Msps)
Resolve (32 @ 365 MHz)
Modify (32 @ 365 MHz)
Output Stage
Software hashing
Hardware hash tables
String table compare, Collision resolution

OUTLINE

ICN/CCNx Introduction
Methodology
Data structures + Algorithms
Results

parc
A Xerox Company

INFORMATION CENTRIC NETWORKS

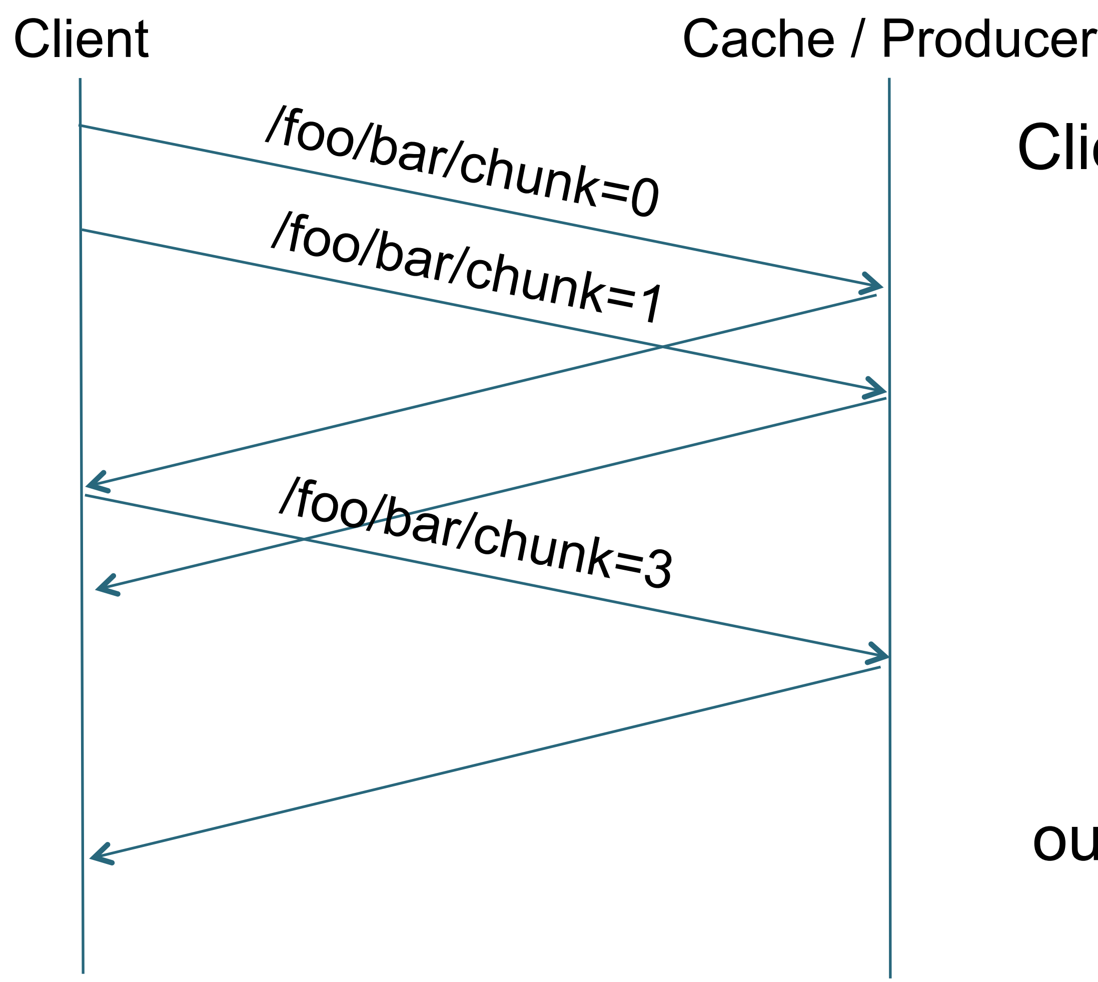
Name the data

Transfer data based on the names

Break end-to-end paradigm

Ted Nelson’s Project Xanadu (1979)

REQUEST/RESPONSE PROTOCOL

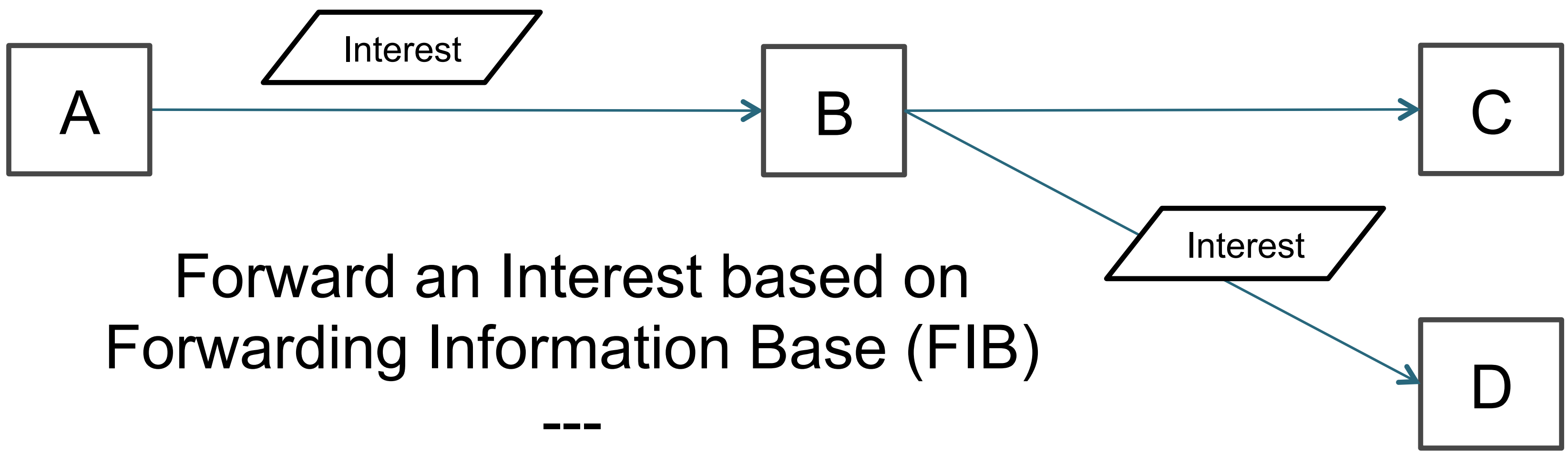


Client sends "Interest" with name

Cache/Producer sends "Content Object"

Transfer using "window" of outstanding Interests

FORWARDER'S JOB



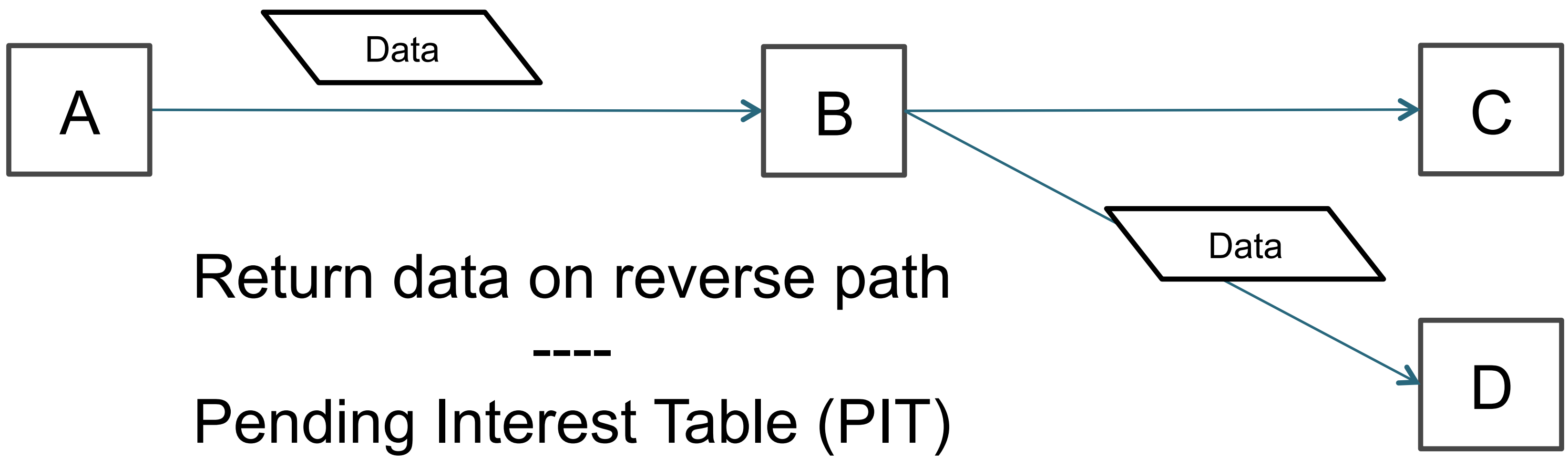
Forward an Interest based on
Forwarding Information Base (FIB)

Token-by-token Longest Prefix Match
(LPM)

lci:/**apple**/**banana**/cherry

(the topic of this talk)

FORWARDER'S JOB



Return data on reverse path

Pending Interest Table (PIT)

(not covered in this talk)

METHODOLOGY

Statistical description of CCNx Names

+

Data Structures & Algorithms

+

Hardware Performance Model

=

Expected Performance

WHAT IS A NAME?

Use the Stanford WebBase for URIs (http://ilpubs.stanford.edu:8090/652)

March 2014

64 million pages from 39,624 web sites.

Download of Links is 251GB.

Yielded 275 million unique URIs.

parc A Xerox Company

NAME ANATOMY

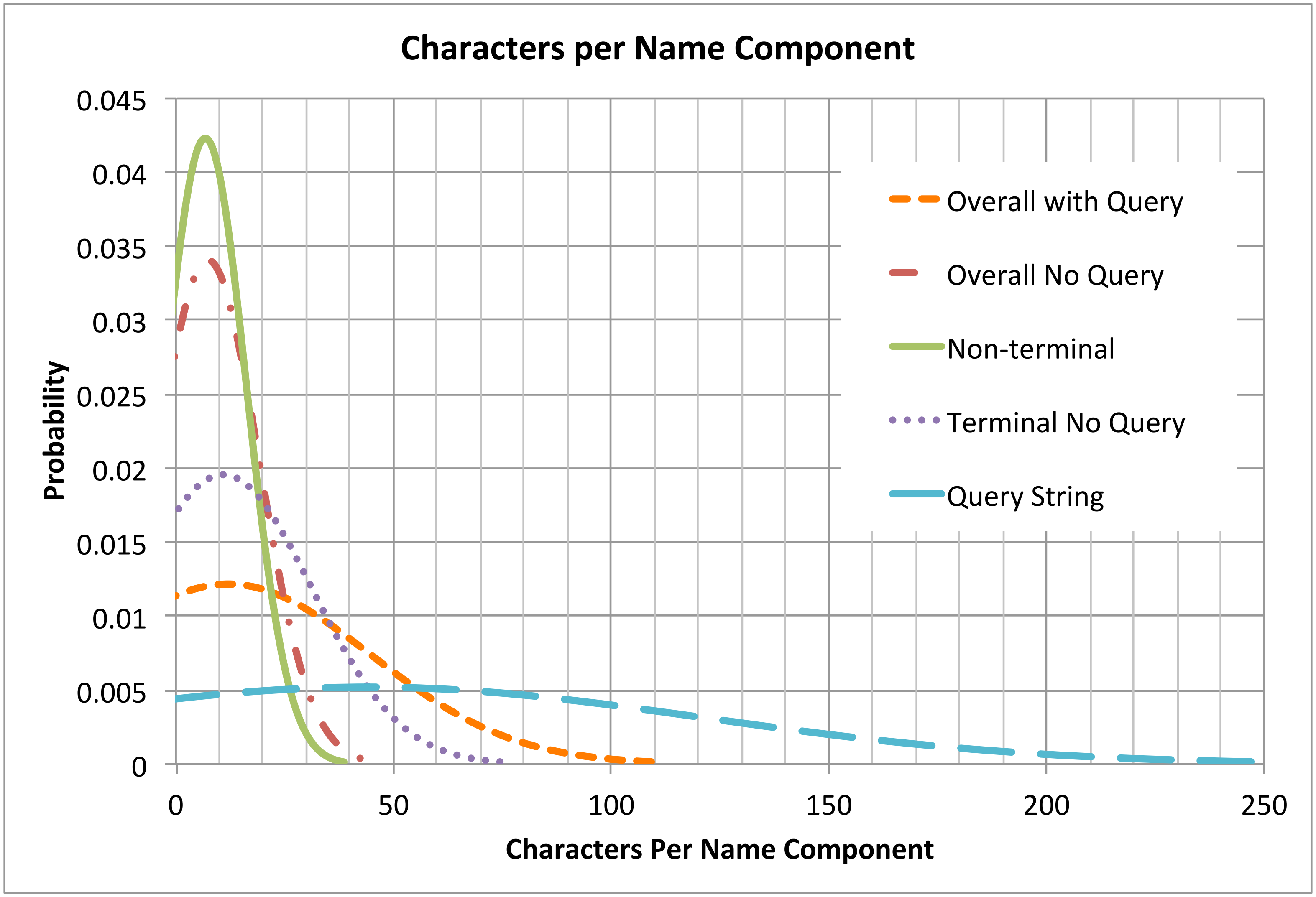
http://host.dom.tld/a/b/c/d?query_string

lci:/tld/dom/host/a/b/c/d/query_string

Non-terminal tokens

Terminal token

ANALYSIS OF URI NAMES



parc
A Xerox Company

MAIN TAKEAWAY

	mean	stdev	99% bound
Name components	7.08	1.99	12.3

	mean	stdev	99% bound
Component Length (Overall With Query)	12.03	32.8	98.0
Component Length (Overall No Query)	7.3	11.7	38.0
Component Length (Non-terminal)	**6.8**	**9.4**	**31.3**
Component Length (Query String alone)	44.1	76.7	245.1

Some of these are too big for a hardware hash table

parc A Xerox Company

MAIN TAKEAWAY

	mean	stdev	99% bound
Name components	7.08	1.99	12.3

	mean	stdev	99% bound
Component Length (Overall With Query)	12.03	32.8	98.0
Component Length (Overall No Query)	7.3	11.7	38.0
Component Length (Non-terminal)	**6.8**	**9.4**	**31.3**
Component Length (Query String alone)	44.1	76.7	245.1

The QueryString (app data) more than triples the storage requirements of the Name

parc A Xerox Company

ALGORITHM: INCREMENTAL HYBRID TABLES

Key = ParentKey + NameComponent

Or

Key = ParentKey + SWHash(NameComponent)

EXAMPLE

T	L		T	L		T	L	
1	5	apple	1	3	pie	2	47	Abcd…

K1 = 0x00000000 + “0x00010005apple” + pad

➤ EID = 0x00000011 + FIB entry

K2 = 0x00000011 + “0x000010003pie” + pad

➤ EID = 0x00002200 + FIB entry

K3 = 0x00002200 + Hash(“0x0002002FAbcd…”)

➤ EID = 0x00770000 + FIB entry + String ID

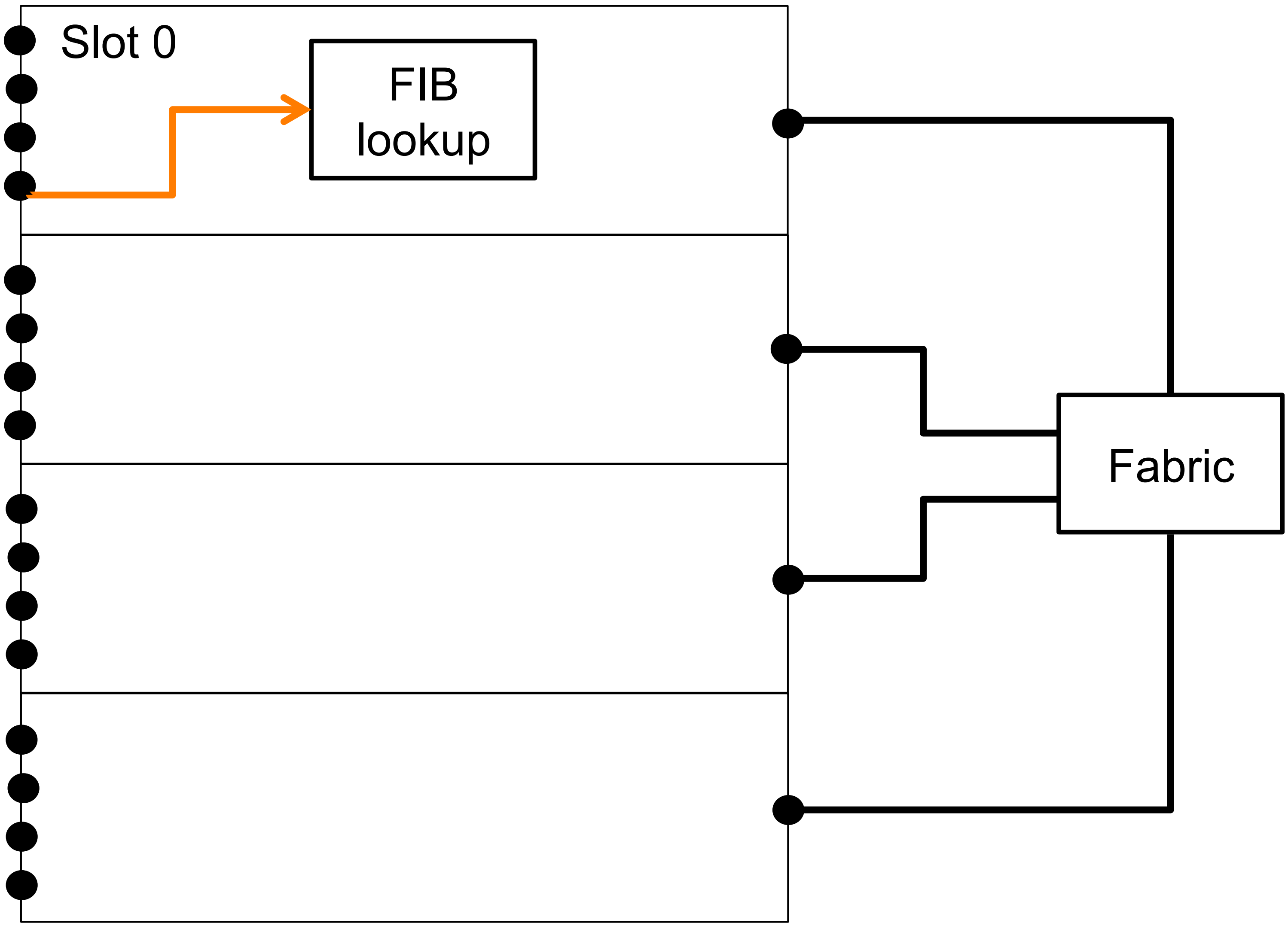
parc A Xerox Company

DATA STRUCTURES

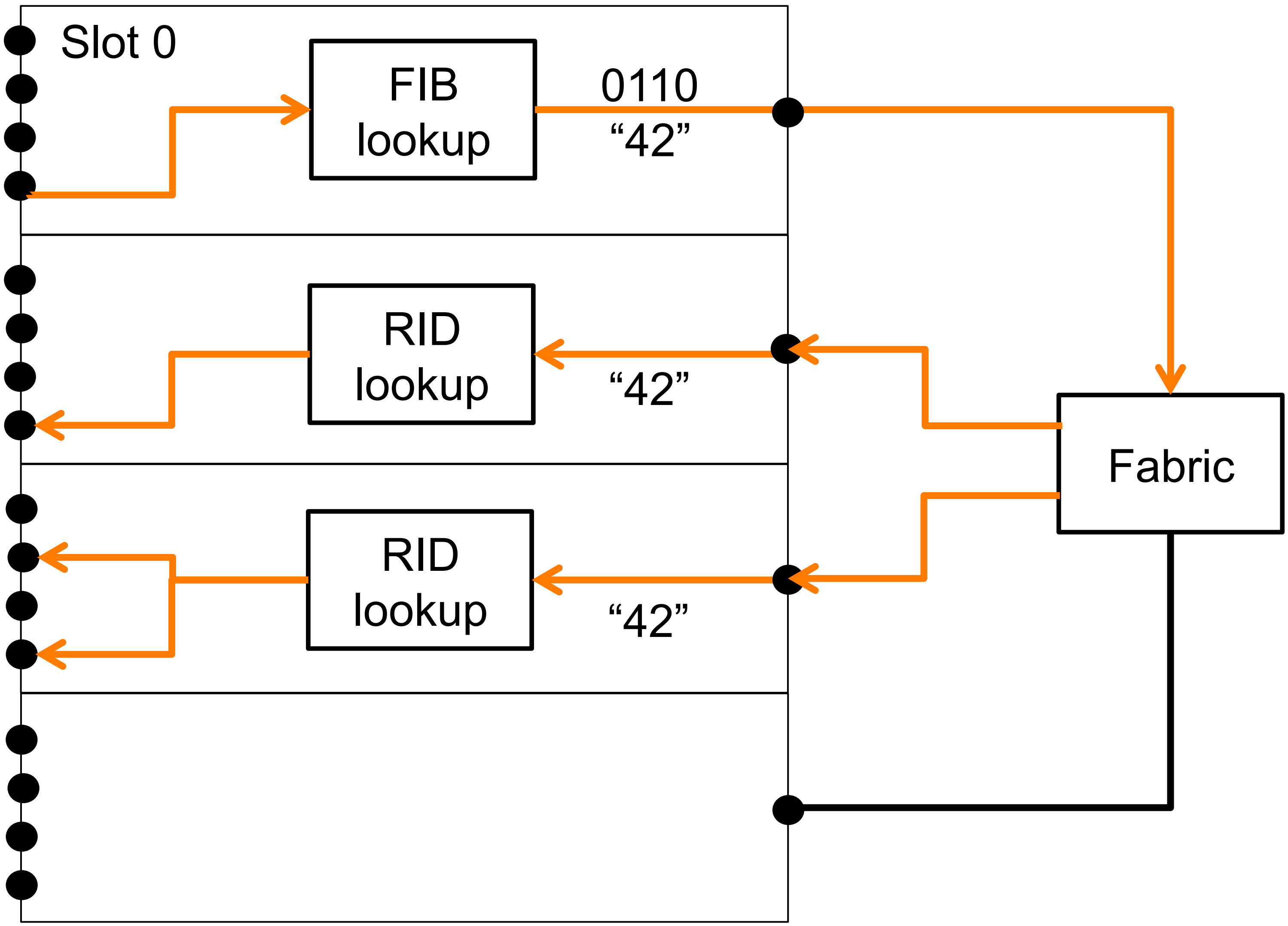
- Control plane knows names N[i] with components N[i,j]
  – It inserts FIB entries on each line card and uses a bitmap for egress card slots. It also contains a Route ID (RID) programmed on each egress slot to resolve specific media ports.
  – Fabric switching is done on the egress bitmap and carries the RID.
  – On each egress card, the RID resolves to the specific egress media ports on that card.
  – For name components that are too large for a hardware hash table key, the control plane also inserts the full name component in a string table identified by a String ID (SID).
    - If two or more name components (plus parent id) collide, there is also a Collision ID (CID), but for the hash size used this is very rare.

parc
A Xerox Company

EXAMPLE



EXAMPLE



HYBRID TABLES

Row Length	Hash Key				Result				
	3B	2B	2B	9B	2B	3B	2B	3B	
16+10 26 ≤ 32	PEID	Type	Len	Value	Flags	EID	Bitmap	RID	
	3B	2B	2B	21B	2B	3B	2B	3B	
28+10	PEID	Type	Len	Value	Flags	EID	Bitmap	RID	
	3B	2B	2B	41B	2B	3B	2B	3B	
48+10 58 ≤ 64	PEID	Type	Len	Value	Flags	EID	Bitmap	RID	
	3B			16B	2B	3B	2B	3B	3B
19+13 32 ≤ 32	PEID			Hash	Flags	EID	Bitmap	RID	SID/ CID

parc
A Xerox Company

FIB TABLE RESULT

Flags	EID	Bitmap	RID	SID/ CID

Flags	NP4 flags
EID	Entry ID (used as Parent ID in next lookup)
Bitmap	Indicates egress slots
RID	Route ID (index to table on each egress card)
SID/CID	String ID or Collision ID (software hash lookups)

parc
A Xerox Company

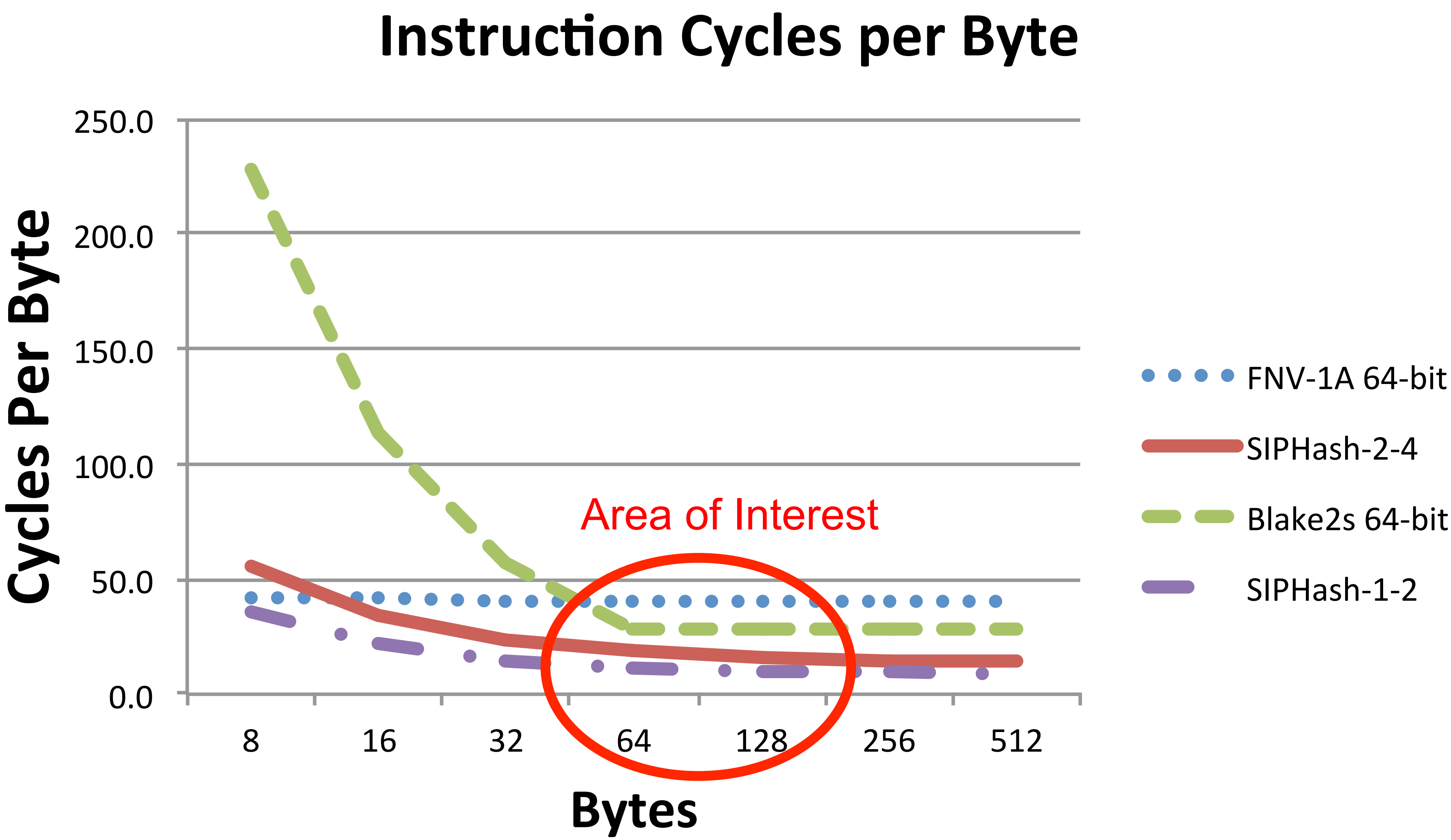
WHAT CAN GO WRONG?

Software Hash will result in key collisions

If routing process inserts two names with same hash, it detects and indicates a “Collision ID” for second lookup

A lookup on a name may not be an actual match (hash bucket collision) so still need to do a string comparison via “String ID” lookup

SOFTWARE HASHING



parc
A Xerox Company

ARCHITECTURE SUMMARY

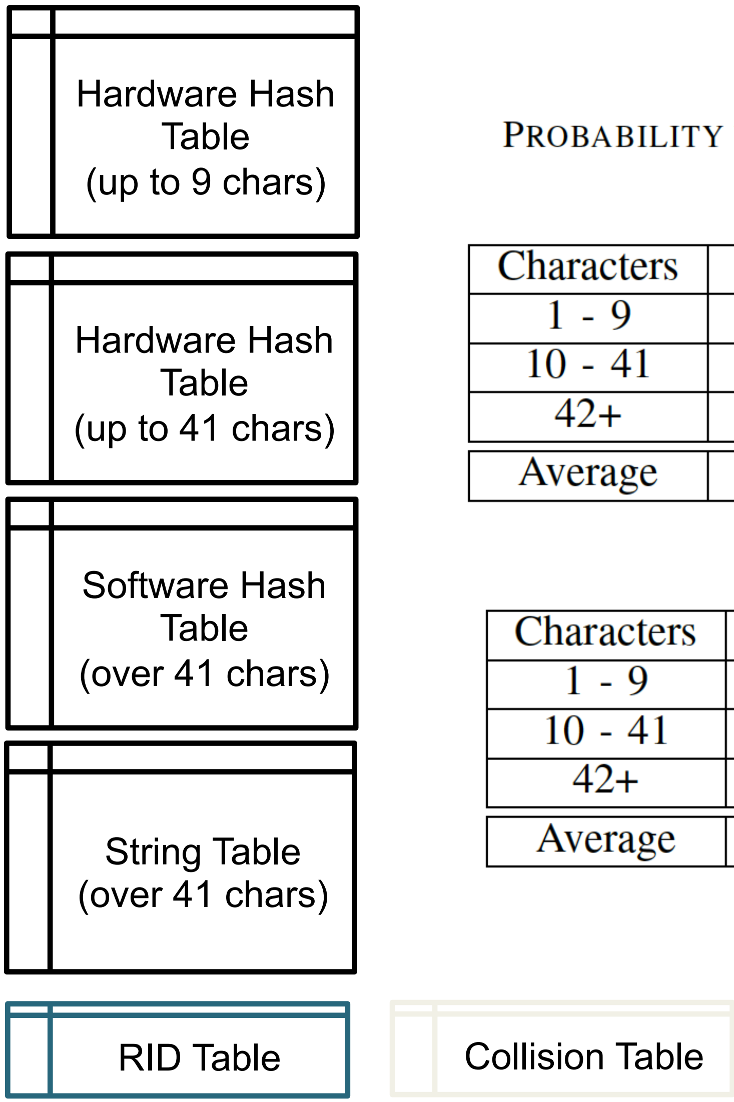


TABLE I
PROBABILITY WEIGHTED BYTES AND CORE CYCLES PER ENTRY

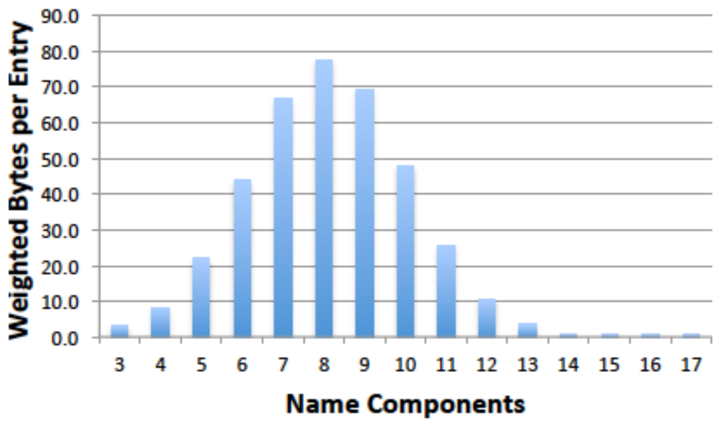
Characters	Probability	Memory Bytes	Core Cycles
1 - 9	59.42%	21.39	76.06
10 - 41	40.57%	27.59	77.89
42+	0.01%	0.03	0.03
Average		49.00	153.97

(a) Non-terminal Components

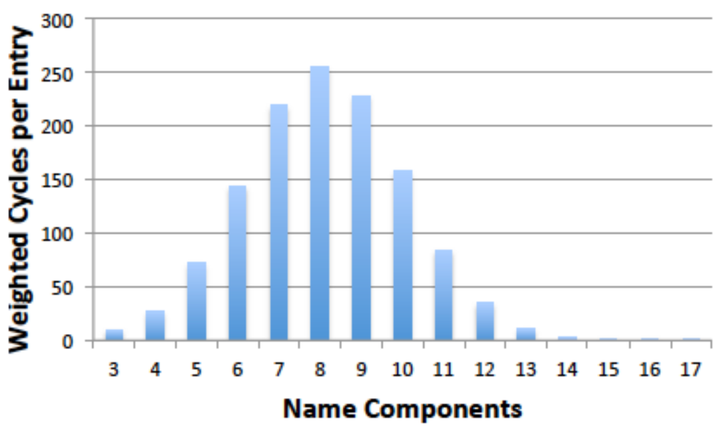
Characters	Probability	Memory Bytes	Core Cycles
1 - 9	46.34%	16.68	59.31
10 - 41	46.67%	31.74	89.61
42+	6.99%	15.94	17.90
Average		64.36	166.82

(b) Terminal Components

FINAL RESULTS



(a) Weighted Bytes



(b) Weighted Cycles

Fig. 5. Probability weighted name distributions

Summing 3 … 17

Average FIB Entry is 378.3 bytes

Average Lookup is 1246.2 core cycles

EXPECTED PERFORMANCE

2GB DRAM stores 5.6M FIB entries

1246 core cycles over 128 lookup engines service 37 Mpps

Computing SIPHash 2-4 for the 0.01% of long tokens is done in the Parse TOP takes 1178 cycles, done in parallel with search

CONCLUSION

- Analysis of today's URIs shows that most names will fit in hardware hash tables.
- Implemented several hash functions on EZchip NP4.
- Propose a multi-stage incremental hash lookup to allow mixing plain keys and software compressed keys in hardware hash tables.
- Using performance model from EZchip, estimate 37Mpps.
- Software compressed keys are calculated in separate processor from hash lookups.

parc
A Xerox Company