A Concise Forwarding Information Base for Scalable and Fast Name Switching

Chen Qian

University of Kentucky

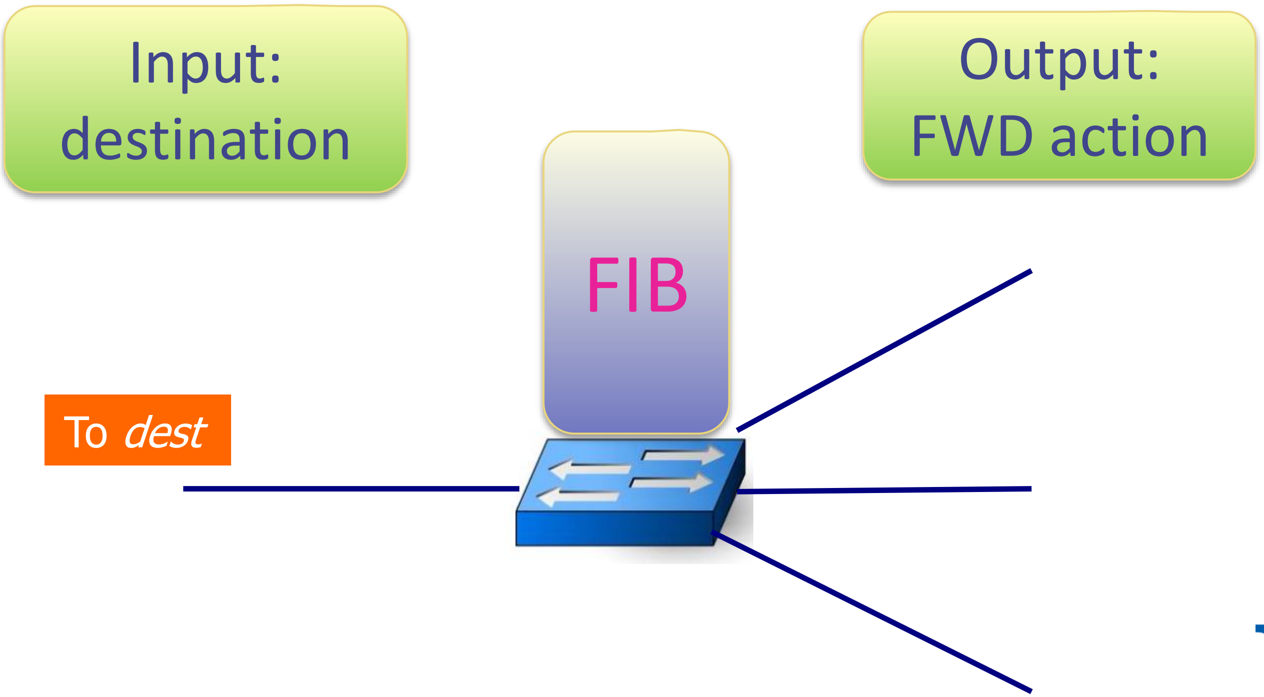
University of California Santa Cruz

with Ye Yu, Djamal Belazzougui, Qin Zhang

UNIVERSITY OF KENTUCKY

Forwarding Information Base (FIB)

A data structure (typically a table) in a network device to determine forwarding actions



Names vs. IP addresses

- (most) Names are flat, permanent, and location-independent
- Flexible network services for mobile devices and VMs: routing, firewall, VPN, etc.
- Flow IDs (Packet headers) can also be considered as names.
- Biggest problem: FIB explosion

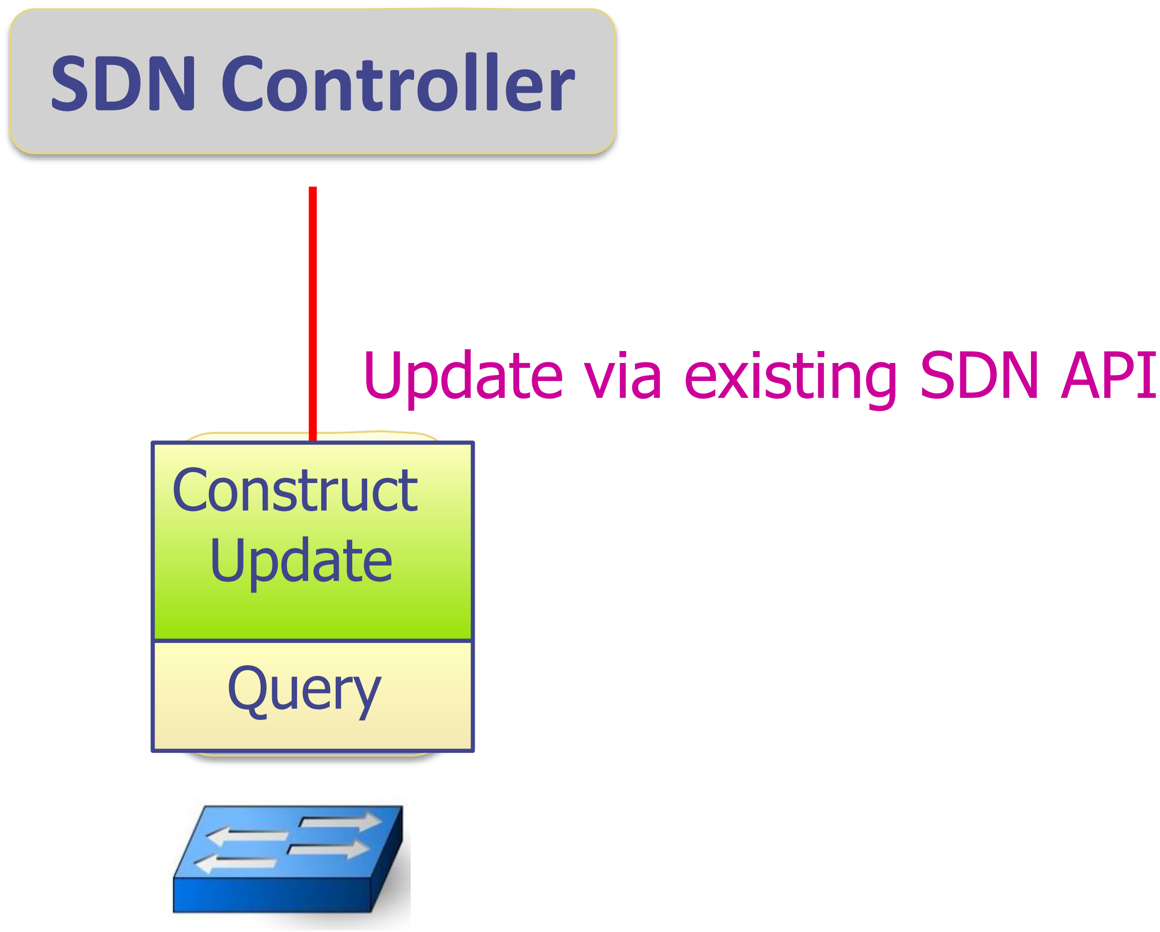
Examples of network names

- Enterprise and data center networks:
  - SEATTLE [SIGCOMM'08], VIRO [Infocom'11], ROME [ICNP'12]
- Future Internet Architecture (FIA)
  - Layered Naming Architecture [SIGCOMM'04], AIP [SIGCOMM'08]
  - NSF FIA projects: NDN, XIA, MobilityFirst
- LTE access network [SIGCOMM'15]

New FIB design: Concise

- 1. Use the least memory ever
  - Fast memory is expensive and power-hungry.
  - Only 10% - 30% of Cuckoo [CoNEXT'13, SIGCOMM'15]
- 2. Fast query speed ever
  - 2x to 5x advantage
- 3. Update speed slower than some FIBs
  - Still support millions of updates per second.

Idea of Concise



Optimize memory and query cost

Concise functions

- Classify *n* names into *d* different sets.
- Each set is a forwarding action

- Relying on a data structure Othello

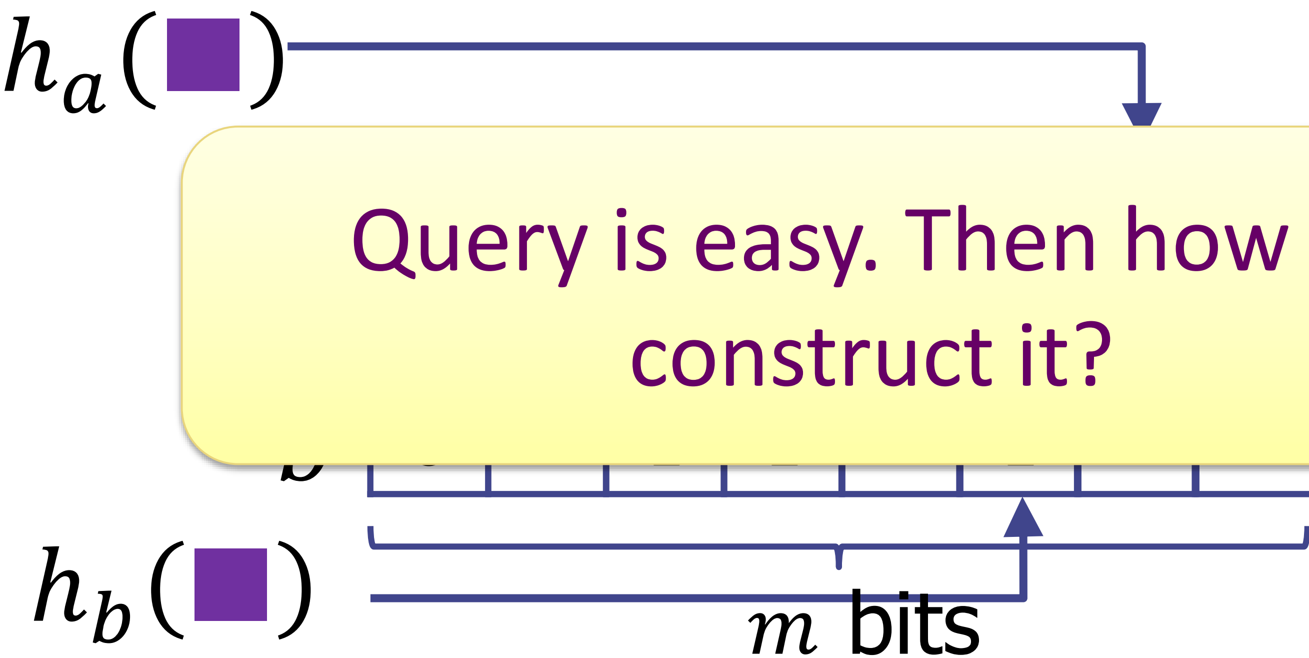
A new data structure Othello

- Classifies names to two sets X and Y
  - Based on MWHC perfect hashing, which is static

- Query result
  - $\tau(k) = 0 \Leftrightarrow k \in X$
  - $\tau(k) = 1 \Leftrightarrow k \in Y$

Othello Query Structure

- Two bitmaps a, b with size m (m in $(1.42n, 2.86n)$)

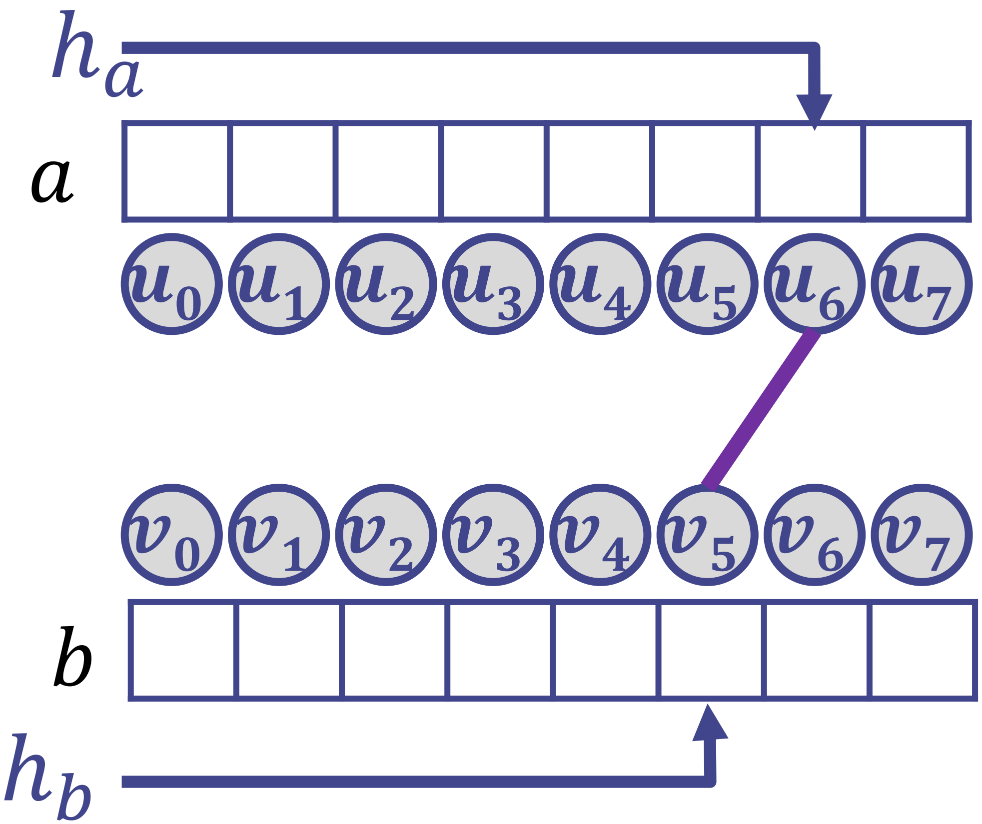


Query is easy. Then how to construct it?

■ is in set Y

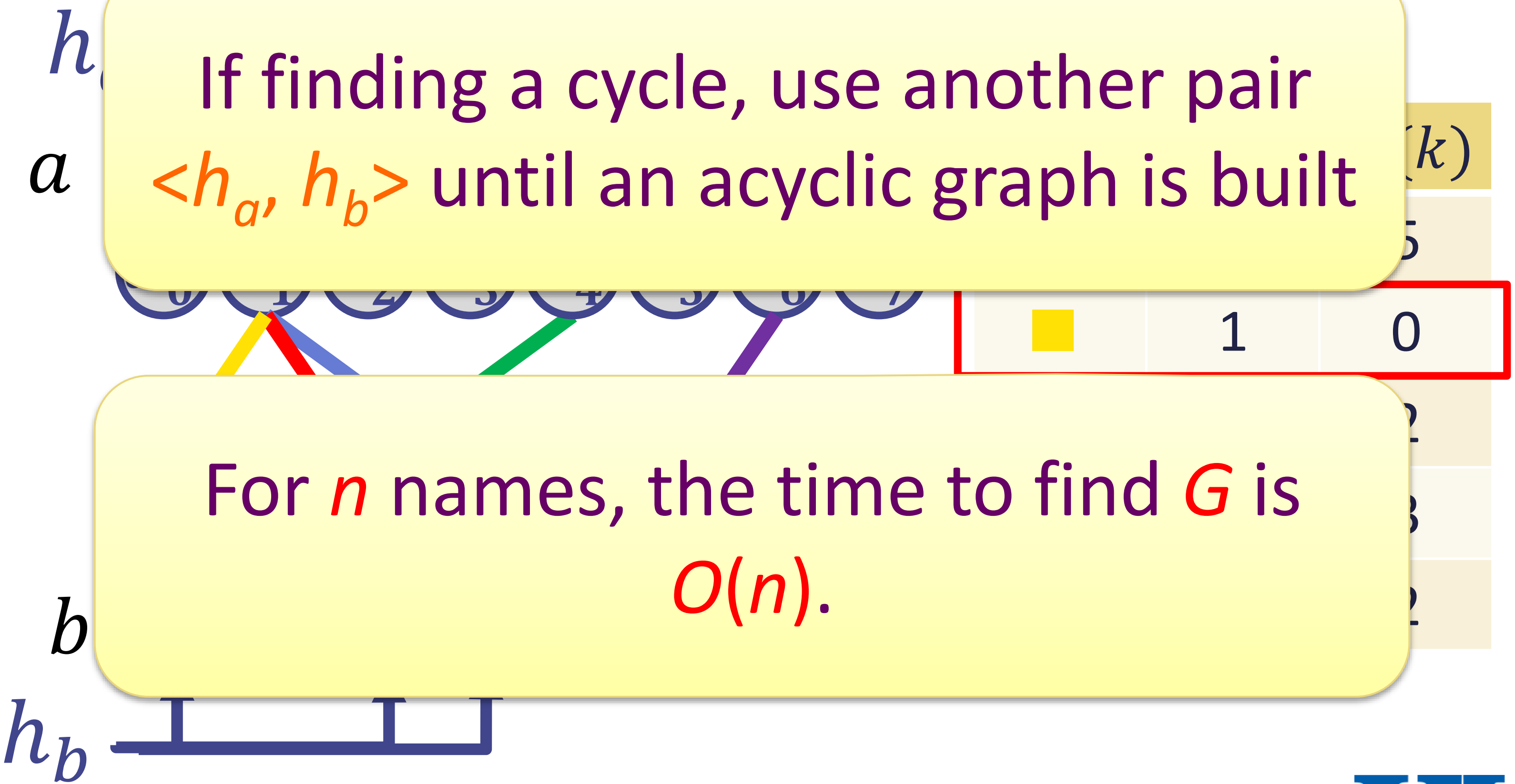
Othello Control Structure: Construct

- G: acyclic bipartite graph



k	$h_a(k)$	$h_b(k)$
■	6	5

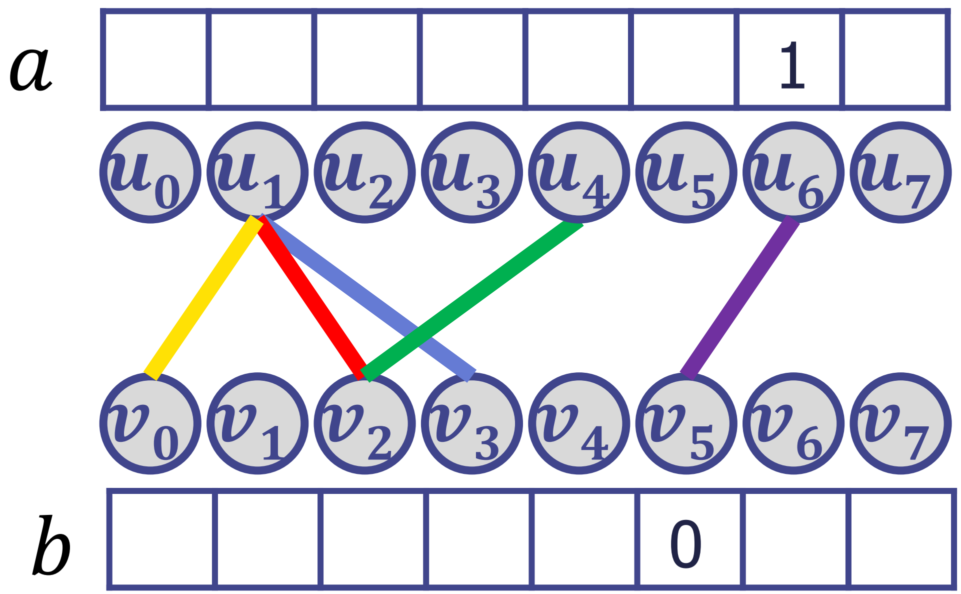
Othello Construct



If finding a cycle, use another pair <h_a, h_b> until an acyclic graph is built

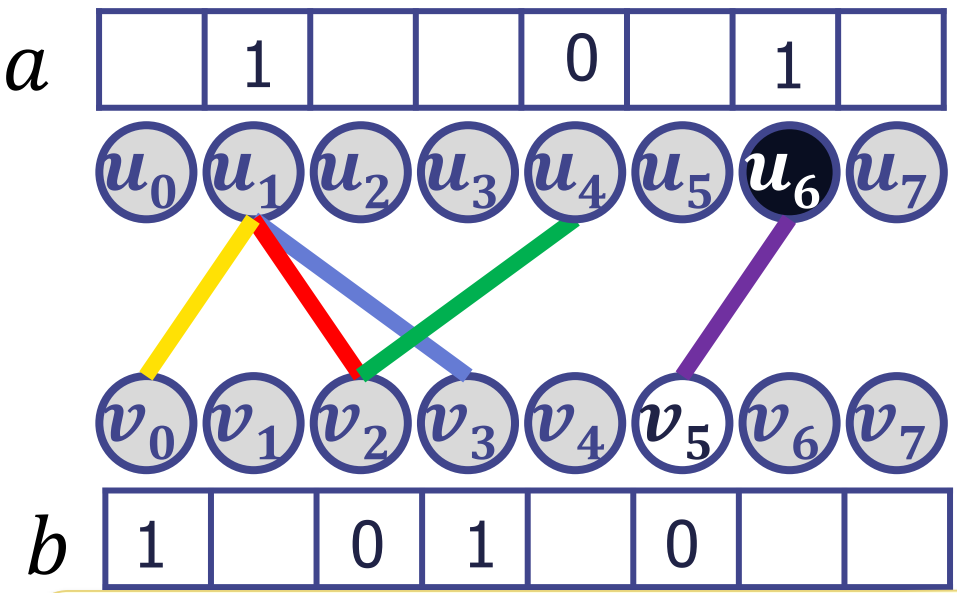
For n names, the time to find G is $O(n)$.

Compute Bitmap



k	$h_a(k)$	$h_b(k)$	set
	6	5	Y
	1	0	
	1	2	
	1	3	
	4	2	

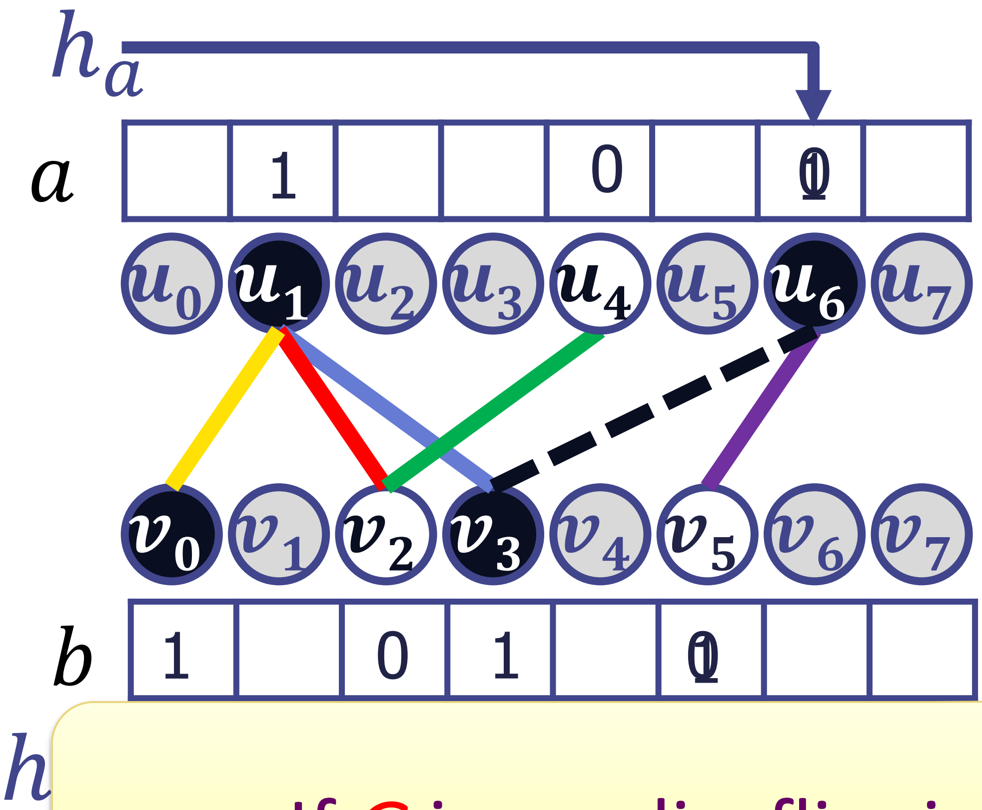
Compute Bitmap



k	$h_a(k)$	$h_b(k)$	set
	6	5	Y
	1	0	X
	1	2	Y
	1	3	X
	4	2	X

If *G* is acyclic, easy to find a coloring plan

Name Addition – color flip

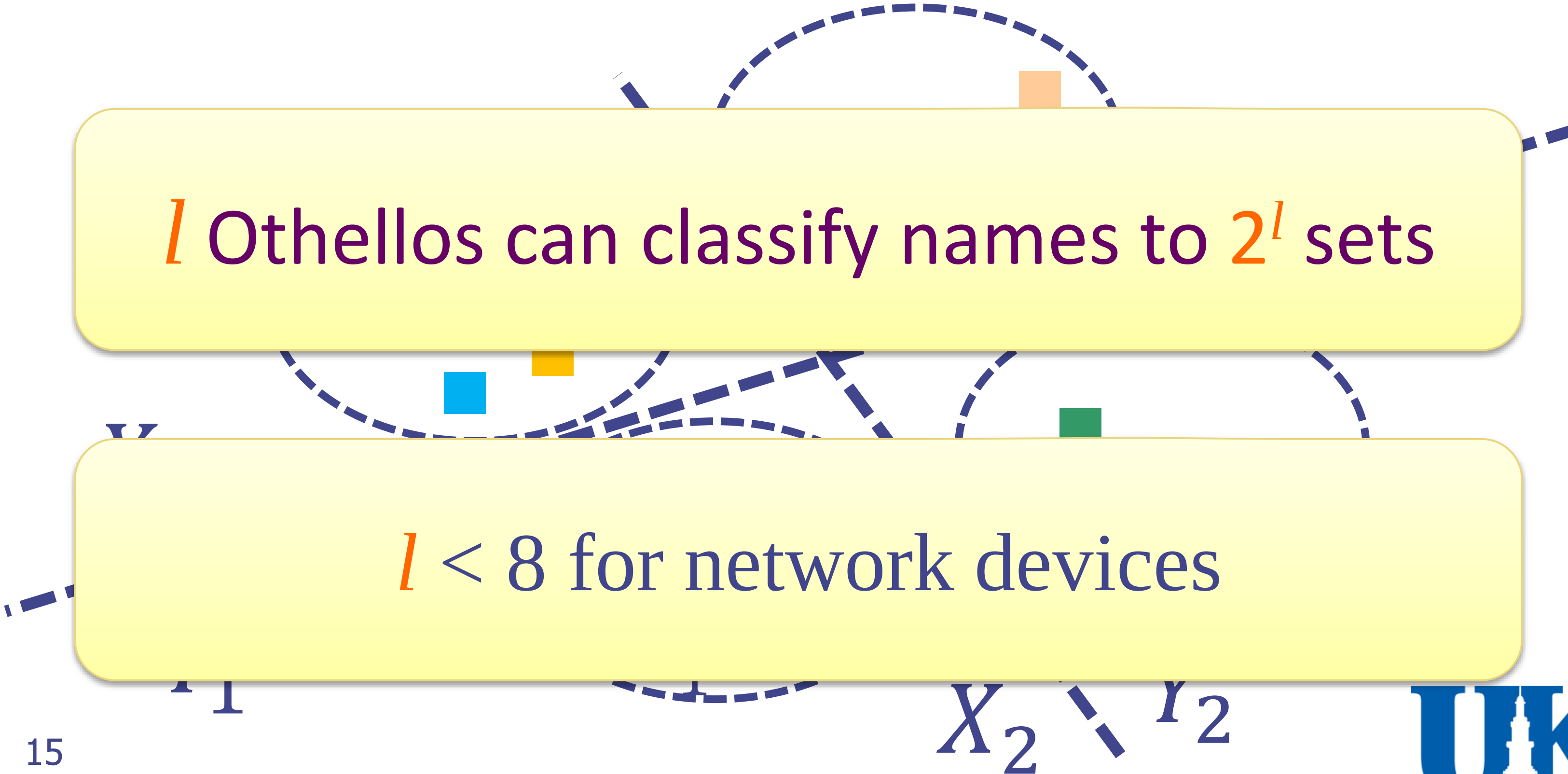


k	$h_a(k)$	$h_b(k)$	set
	6	5	Y
	1	0	X
	1	2	Y
	1	3	X
	4	2	X
	6	3	Y

If G is acyclic, flipping is trivial

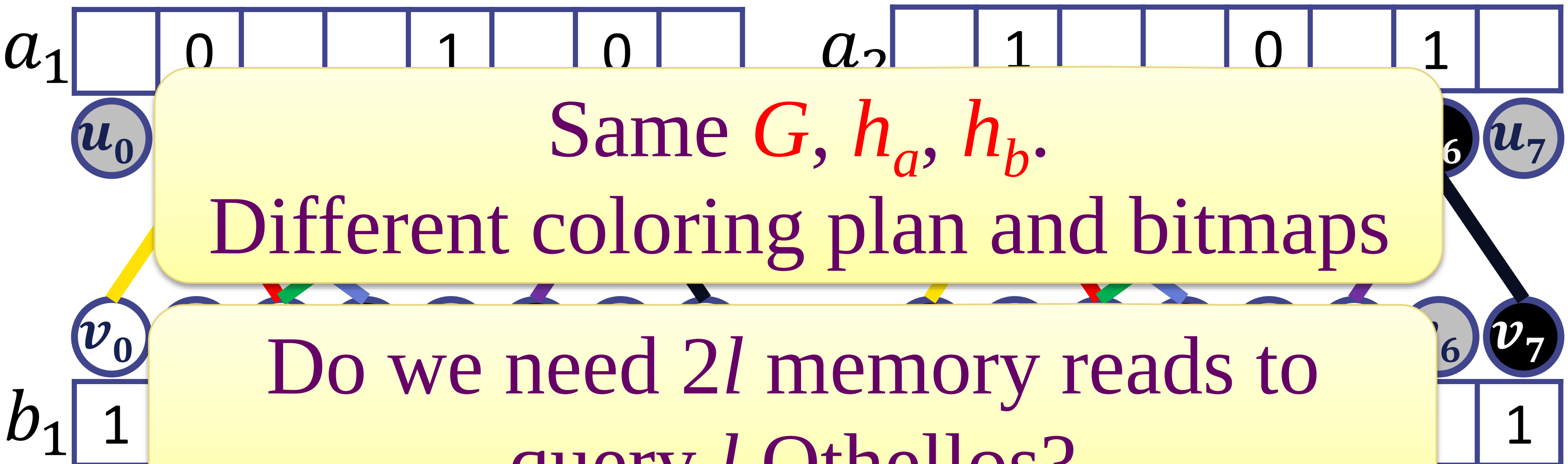
Concise functionality

- Classifies names into 2^l sets: $Z_0, Z_1, \cdots, Z_{2^l-1}$



l Othellos can classify names to 2^l sets

$l < 8$ for network devices

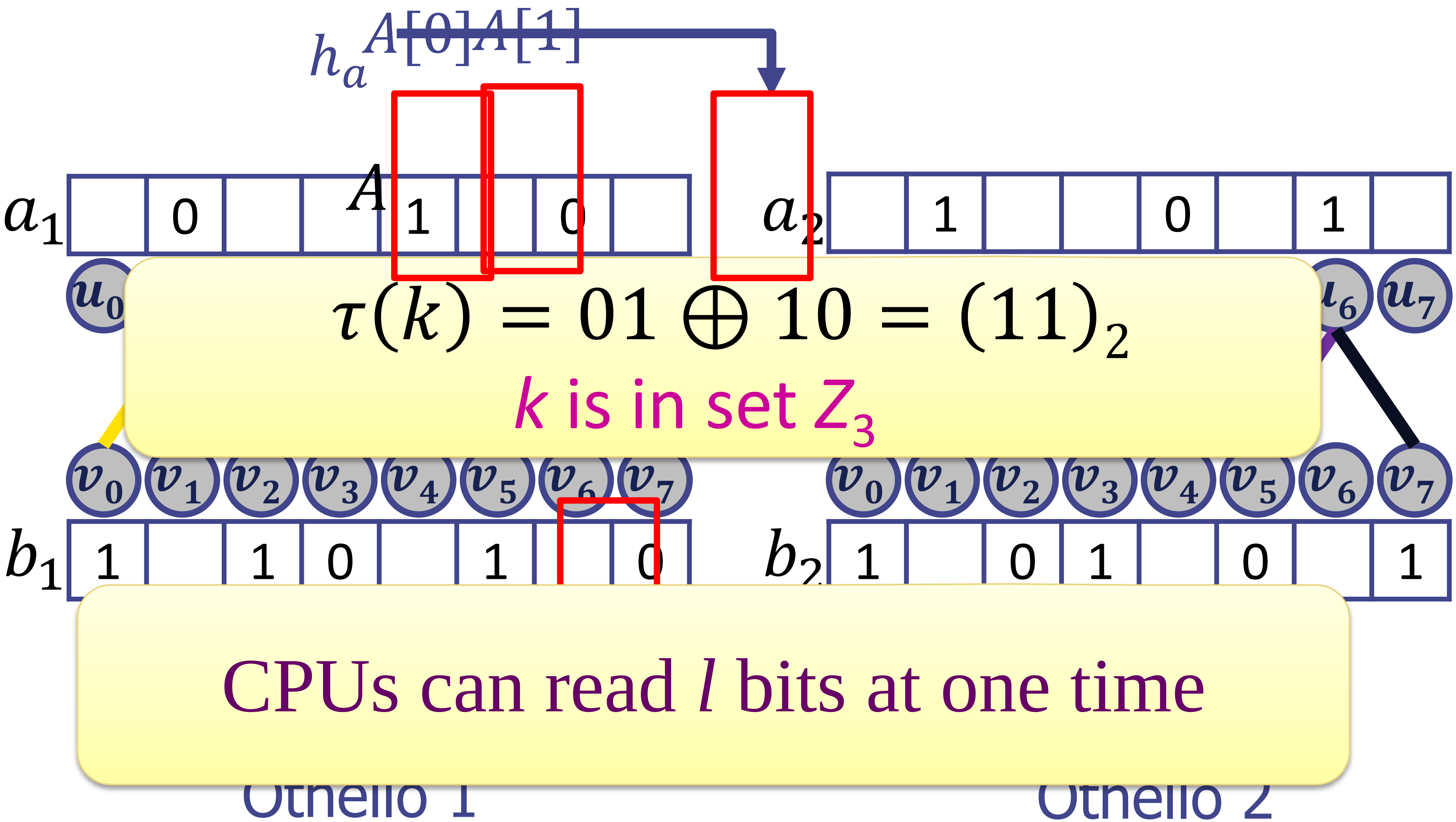


Same G, h_a, h_b.

Different coloring plan and bitmaps

Do we need $2l$ memory reads to query l Othellos?

Othello 1 Same $X \cup Y$ Othello 2



$$\tau(k) = 01 \oplus 10 = (11)_2$$

k is in set Z_3

CPUs can read *l* bits at one time

Implementation of three prototypes

- 1. Memory mode
  - Query and control structures running on different threads.
- 2. CLICK modular router
- 3. Intel Data Plane Development Kit (DPDK)

Comparison: best solutions in the literature

- Buffalo

in CoNEXT'09

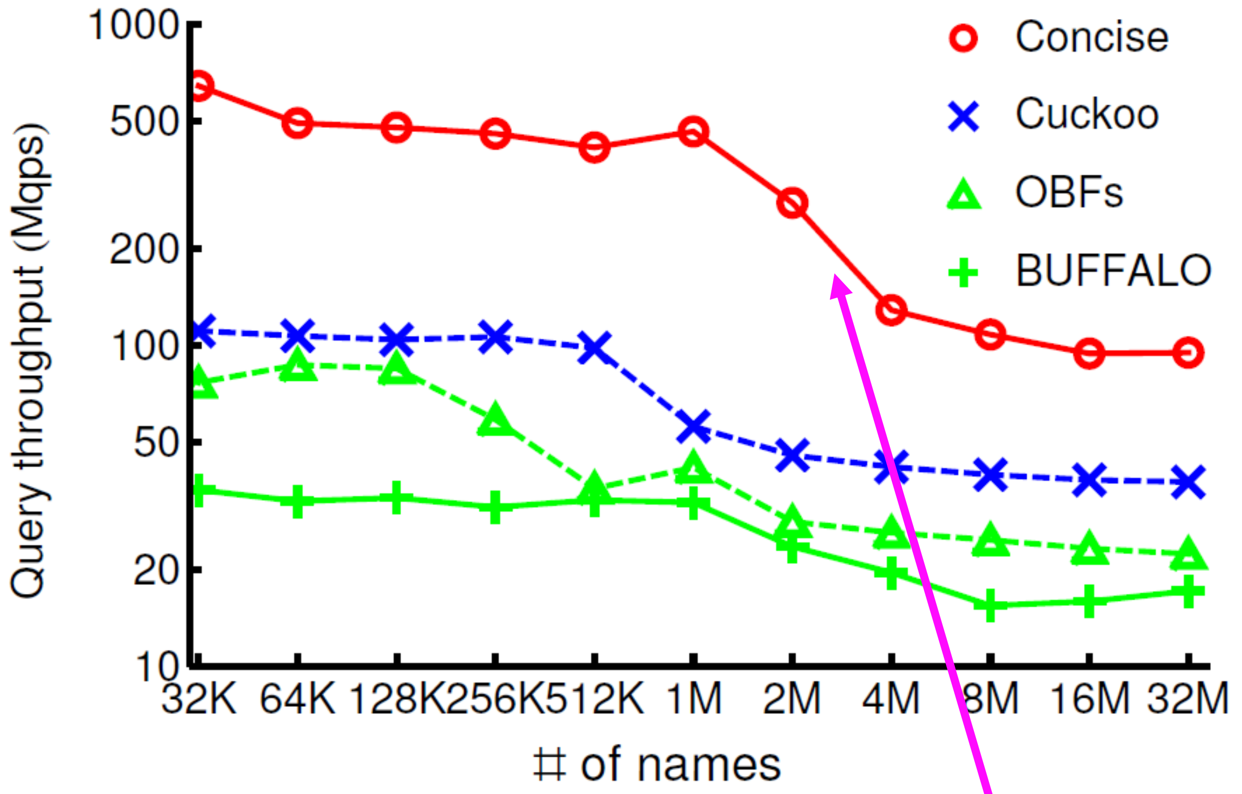
- Cuckoo hashing

in CoNEXT'13 and SIGCOMM'15

Comparison: Memory size

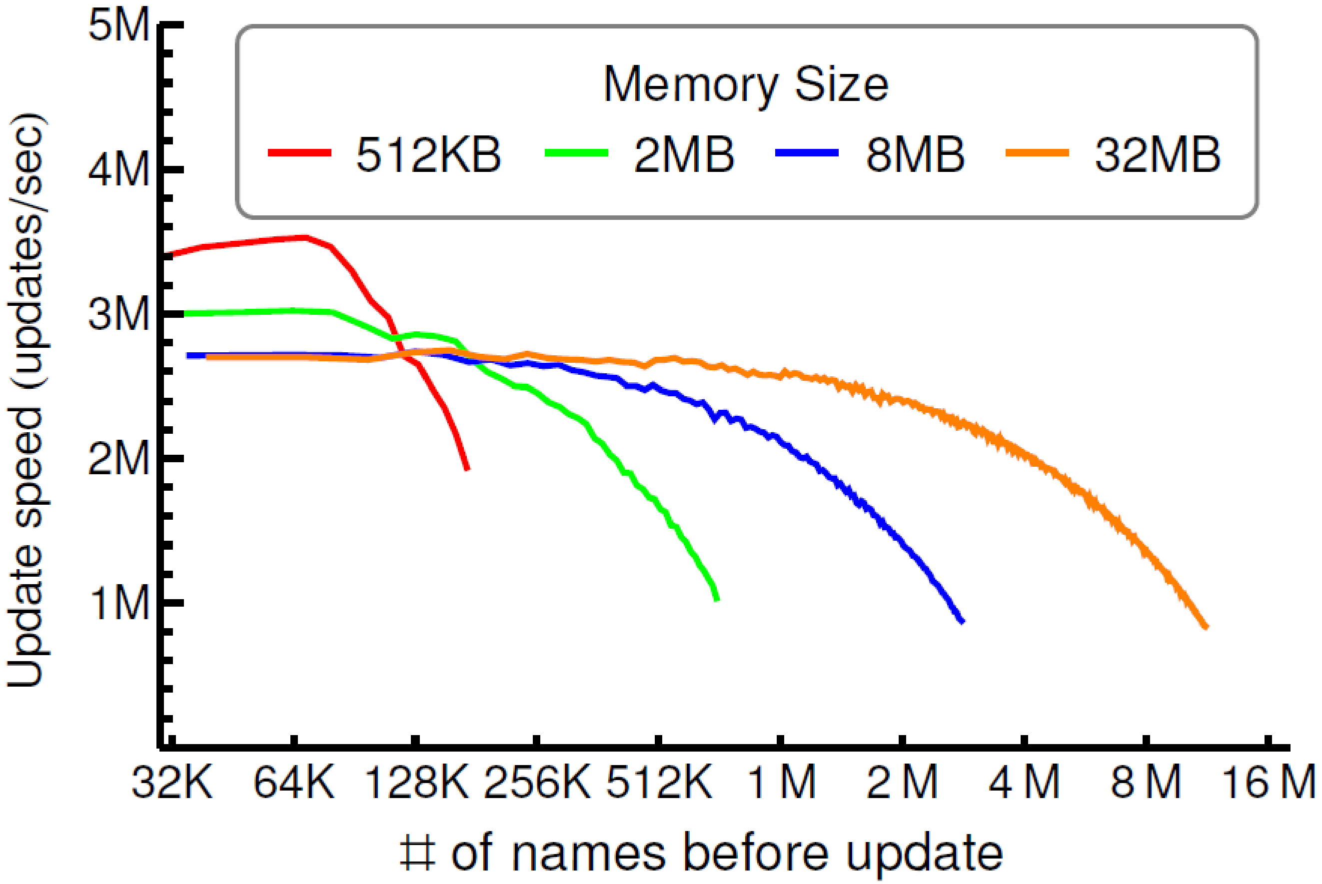
FIB Example			Memory Size		
Name Type	# Names	# Actions	Concise	Cuckoo	Buffalo
MAC (48 bits)	$7*10^5$	16	1M	5.62M	2.64M
MAC (48 bits)	$5*10^6$	256	16M	40.15M	27.70M
MAC (48 bits)	$3*10^7$	256	128M	321.23M	166.23M
IPv4 (32 bits)	$1*10^6$	16	2M	4.27M	3.77M
IPv6 (128 bits)	$2*10^6$	256	8M	34.13M	11.08M
OpenFlow (356 bits)	$3*10^5$	256	1M	14.46M	1.67M
OpenFlow (356 bits)	$1.4*10^6$	65536	8M	67.46M	18.21M
File name (varied)	359194	16	512K	19.32M	1.35M

Query speed



2x to 4x speed advantage

Update



Each update is a network-wide update

More possible applications of Concise

- Essentially a key-value mapping
- 1. Memory cache
- 2. Support query to distributed content storage
- 3. Sparse vector data processing

Thank You

Questions?