⟨INFOCOM 2009⟩

Distributed Arrays

A P2P Data Structure for Efficient Logical Arrays

Daisuke Fukuchi
Christian Sommer
Yuichi Sei
Shinichi Honiden

April 22, 2009

P2P data management

- Scalable
- Low cost
- DHT: (key, value) operations in $O(\log n)$

P2P data management

- Scalable
- Low cost
- DHT: (key, value) operations in $O(\log n)$

but w related items in $O(w \log n)$

e.g. large file split into segments

P2P data management

- Scalable
- Low cost
- DHT: (key, value) operations in $O(\log n)$

  but *w* related items in $O(w \log n)$

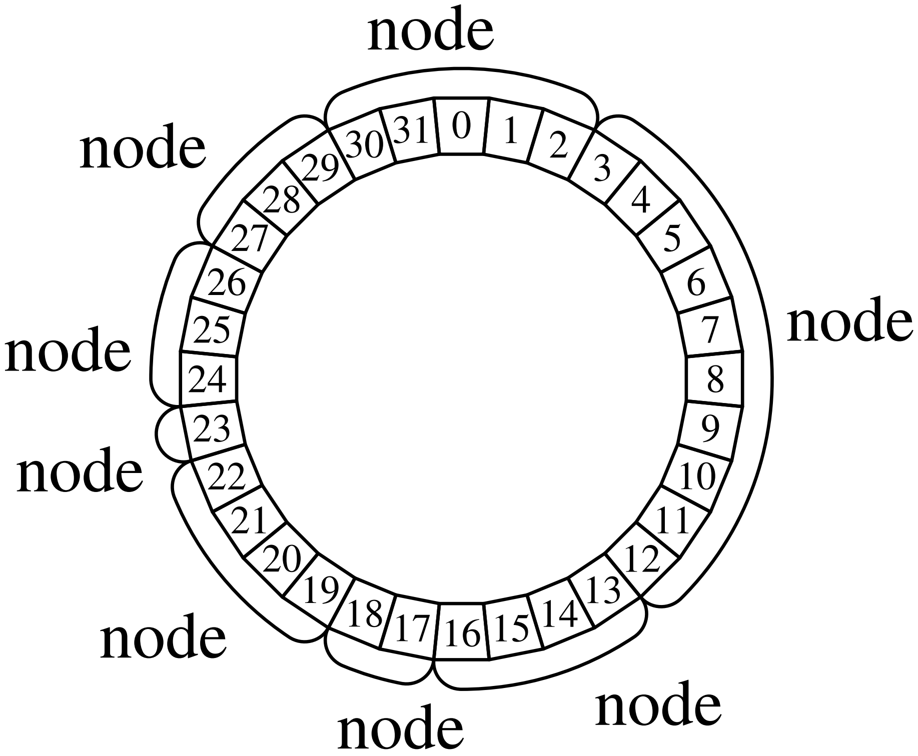
  e.g. large file split into segments

- **DA: *w* array elements in $O(w + \log n)$**

Related work

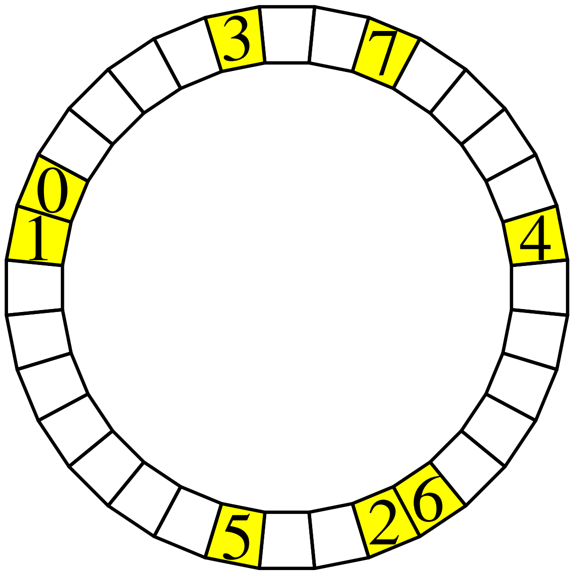
- DHT
  - Chord [Stoica et al., SIGCOMM, 2001]
  - Pastry [Rowston et al., Middleware, 2001]
  - $\cdots$
- P2P range queries
  - Skip graphs [Aspnes et al., ACM Trans. on Algo., 2007]
  - PHT [Ramabhadran et al., Technical Report, 2004]
  - $\cdots$

DHT



- based on ID space
- ID access in $O(\log n)$

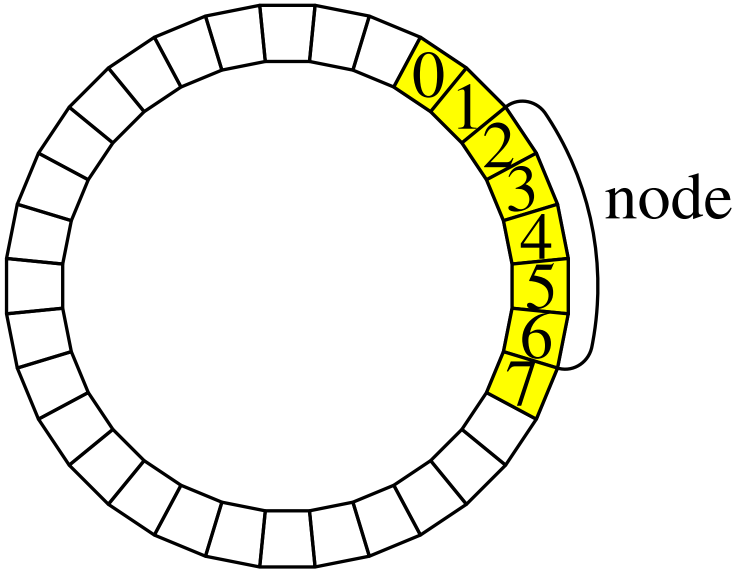
Array using DHT



- Elements one by one
- Pseudo-random placement

⇒ high message cost

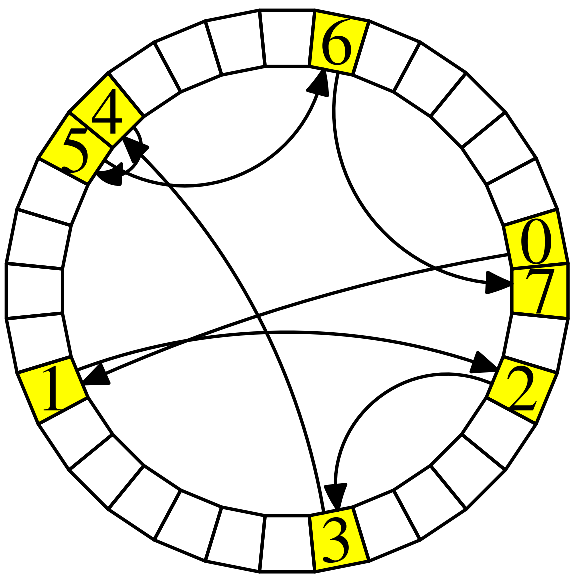
Array using range queries



- Local placement

  ⇒ Load concentration balancing problems

Array using range queries

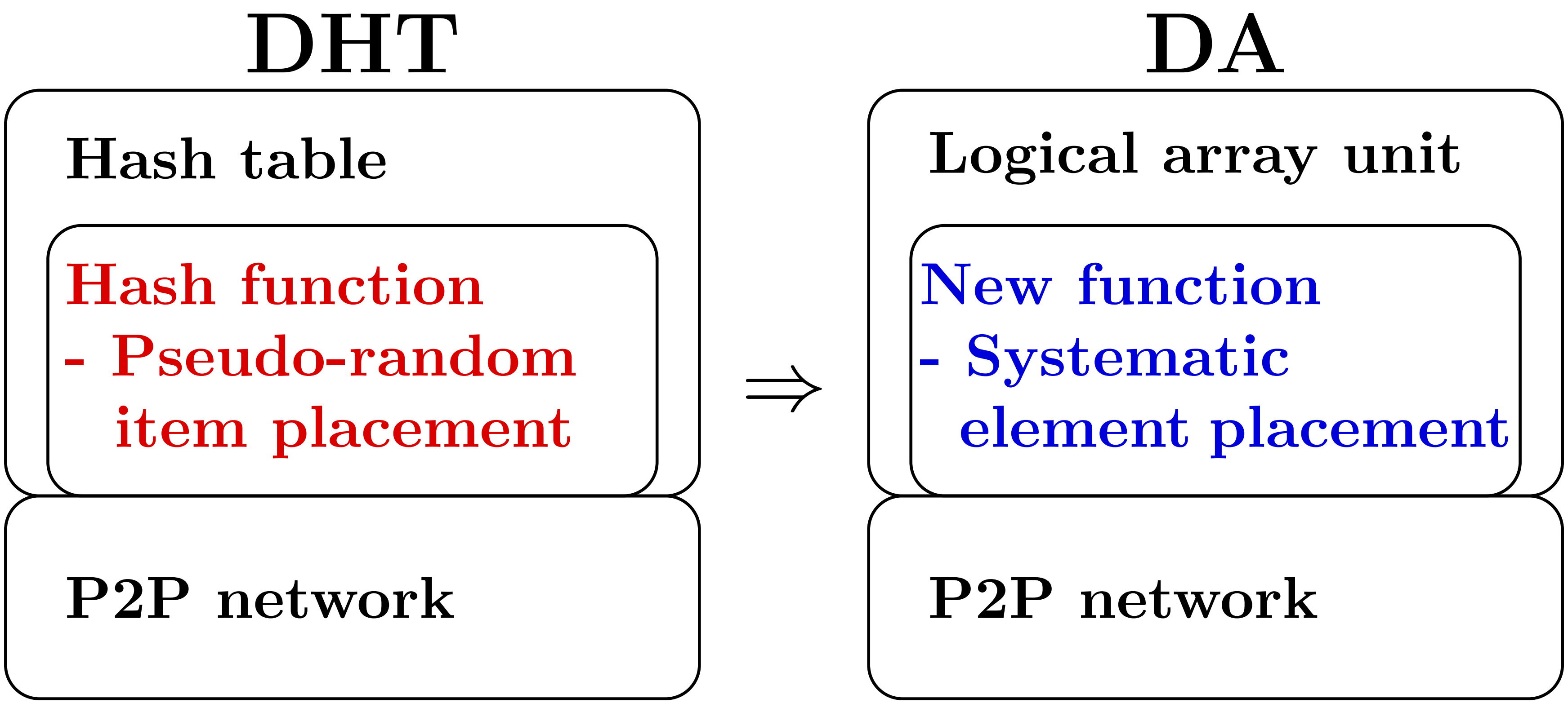


- Additional management structure (linked list, . . .)

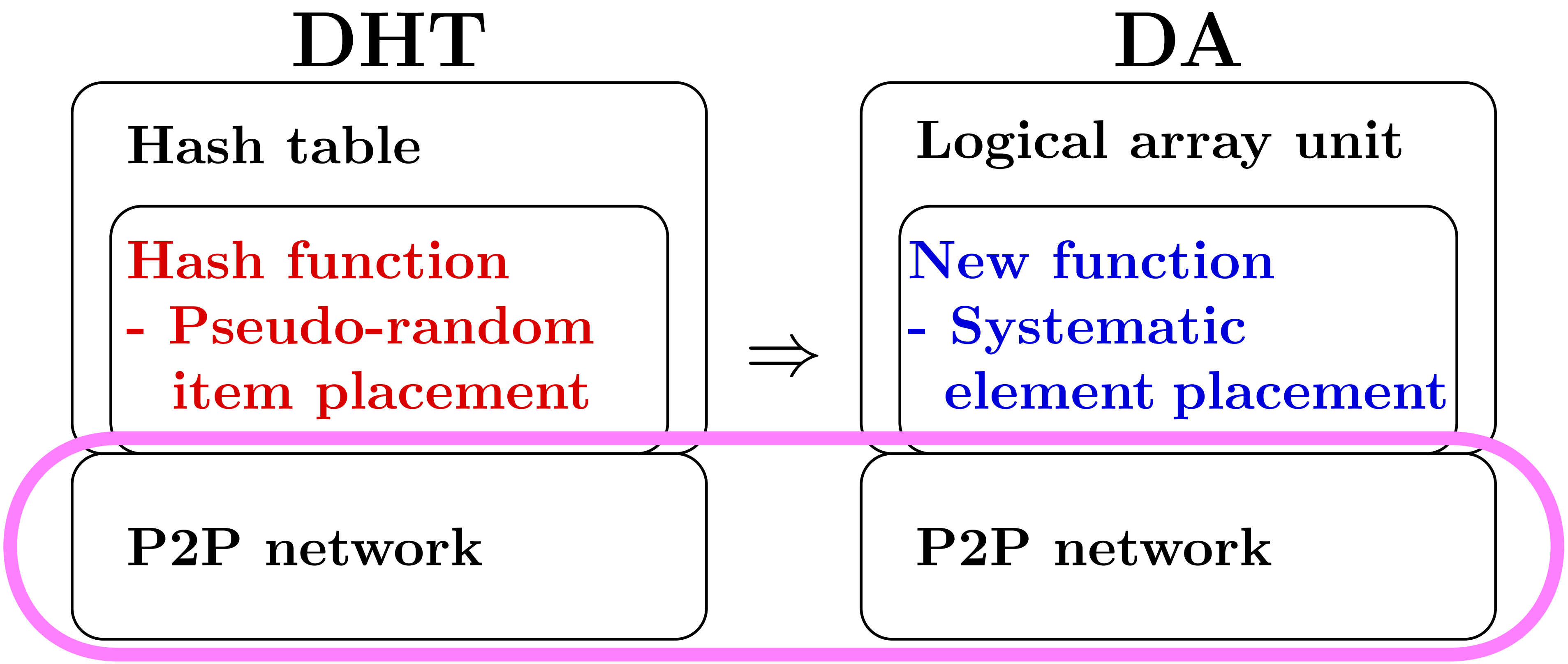
$\Rightarrow$ additional cost

binary search ?

Construction of DA

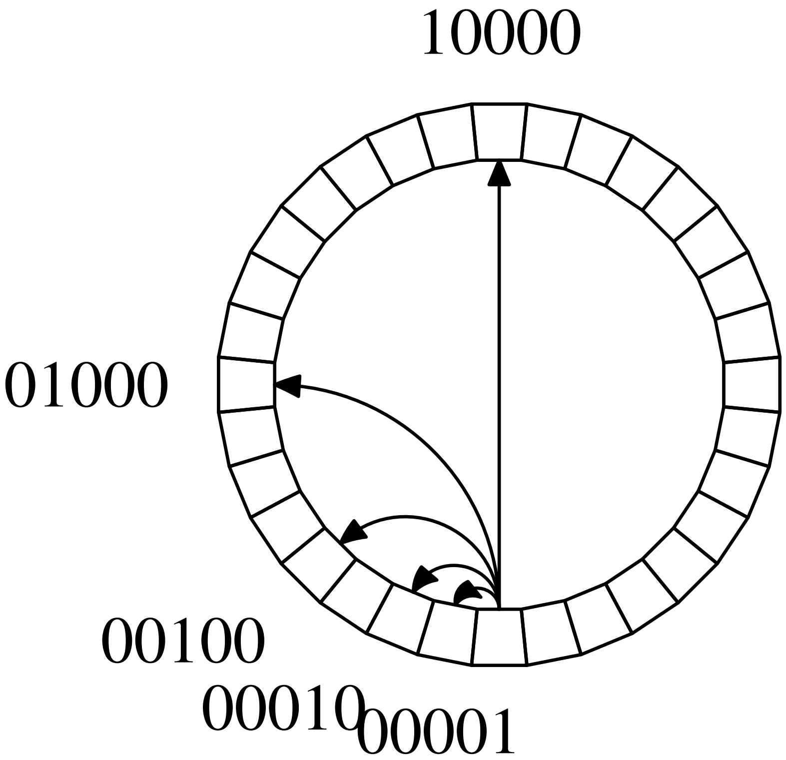


Construction of DA



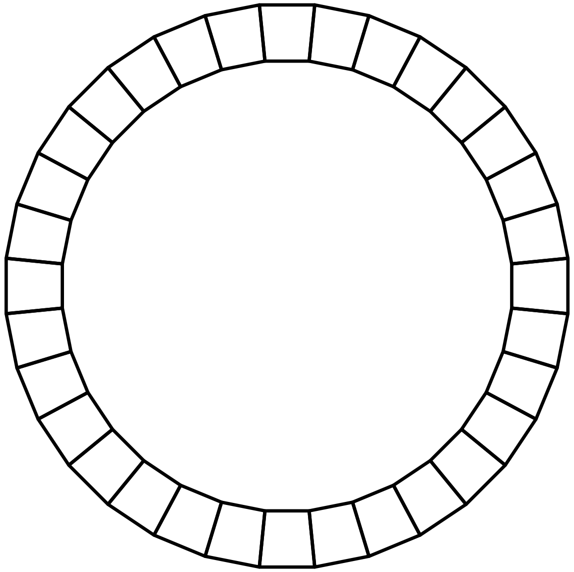
Analysis of the base P2P network (Chord)

- HERE: Ideal (for general environments, see paper)



- 2^k length node pointers

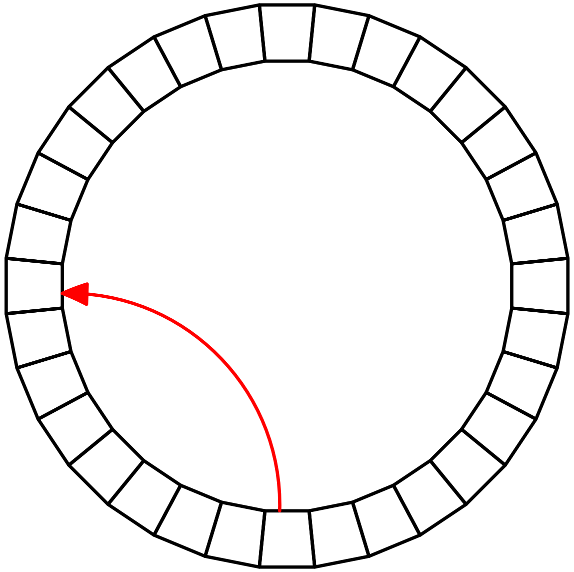
Analysis of the base P2P network (Chord)



- Greedy access

e.g. $01110_{(2)}$ (distance to target ID)

Analysis of the base P2P network (Chord)



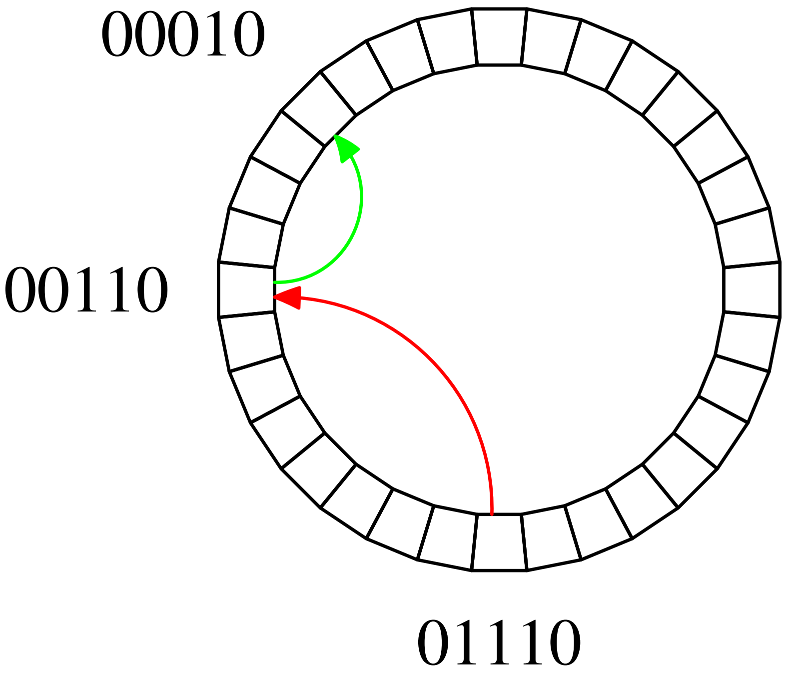
- Greedy access

e.g. $01110_{(2)}$ (distance to target ID)

↓ use $01000_{(2)}$ length pointer

$00110_{(2)}$

Analysis of the base P2P network (Chord)



- Greedy access

e.g. $01110_{(2)}$ (distance to target ID)

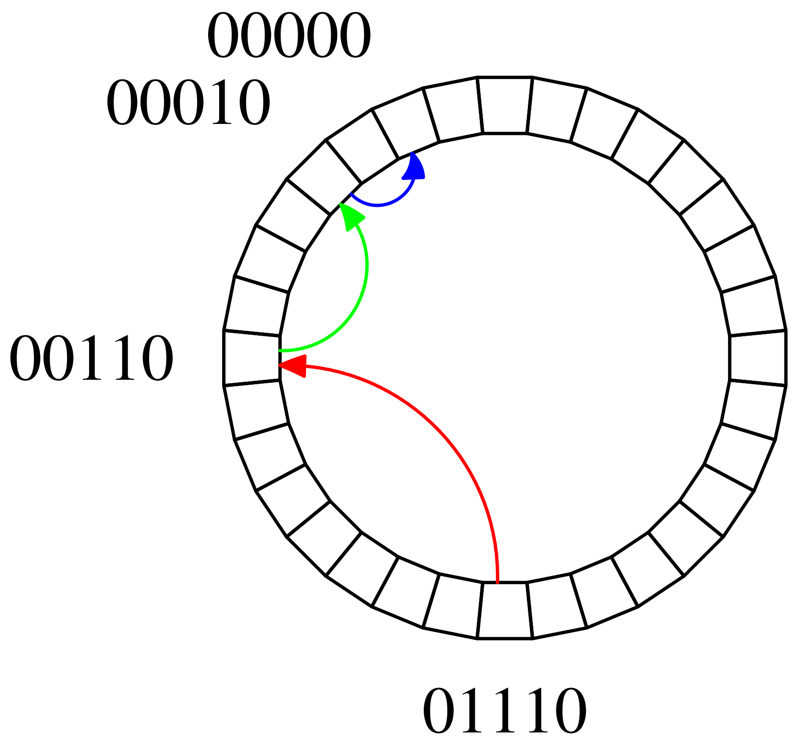
↓ use $01000_{(2)}$ length pointer

$00110_{(2)}$

↓ use $00100_{(2)}$ length pointer

$00010_{(2)}$

Analysis of the base P2P network (Chord)



- Greedy access

e.g. $01110_{(2)}$ (distance to target ID)

↓ use $01000_{(2)}$ length pointer

$00110_{(2)}$

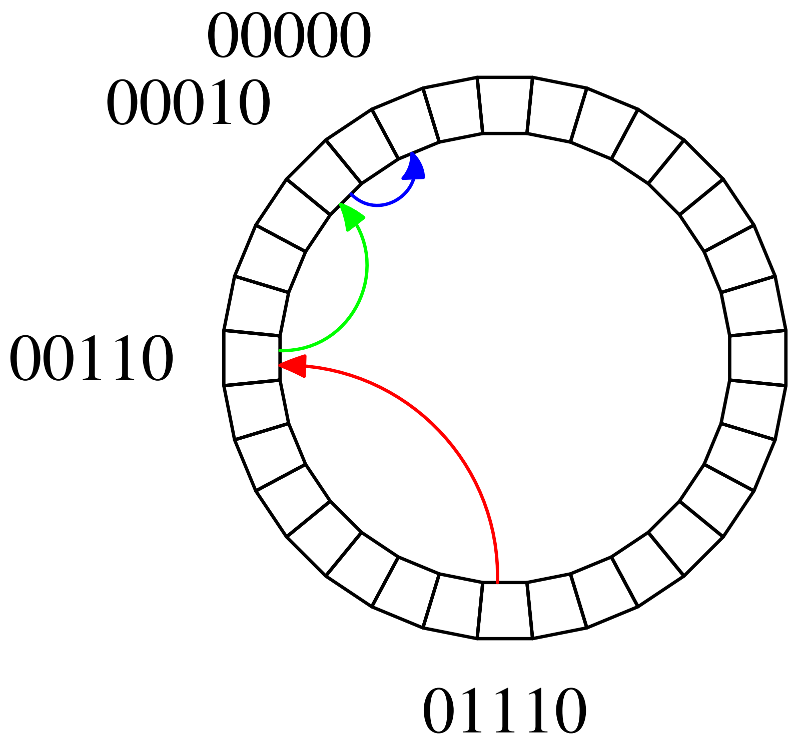
↓ use $00100_{(2)}$ length pointer

$00010_{(2)}$

↓ use $00010_{(2)}$ length pointer

$00000_{(2)}$

Analysis of the base P2P network (Chord)



- Greedy access

e.g. $01110_{(2)}$ (distance to target ID)

↓ use $01000_{(2)}$ length pointer

$00110_{(2)}$

↓ use $00100_{(2)}$ length pointer

$00010_{(2)}$

↓ use $00010_{(2)}$ length pointer

$00000_{(2)}$

No. of messages	**=**	**No. of 1's in distance**

Introducing systematic element placement

- **Efficiency:**

ID | | **same**

e.g.

ID 410 = 0 0 0 1 1 0 0 1 1 0 1 $0_{(2)}$

ID 1434 = 0 1 0 1 1 0 0 1 1 0 1 $0_{(2)}$

ID 3738 = 1 1 1 0 1 0 0 1 1 0 1 $0_{(2)}$

No. of messages = **No. of 1's in distance**

Introducing systematic element placement

- **Efficiency:**

ID		**same**

e.g.

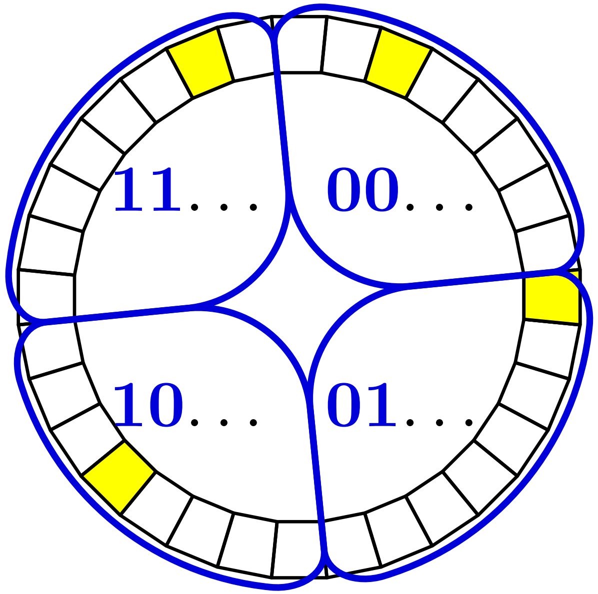
$$\begin{aligned}
\text{ID } 410 &= 0\,0\,0\,1\;\;1\,0\,0\,1\,1\,0\,1\,0_{(2)} \\
\downarrow 1024 &= 0\,1\,0\,0\;\;\mathbf{0\,0\,0\,0\,0\,0\,0\,0}_{(2)} \\
\text{ID } 1434 &= 0\,1\,0\,1\;\;1\,0\,0\,1\,1\,0\,1\,0_{(2)} \\
\downarrow 2304 &= 1\,0\,0\,1\;\;\mathbf{0\,0\,0\,0\,0\,0\,0\,0}_{(2)} \\
\text{ID } 3738 &= 1\,1\,1\,0\;\;1\,0\,0\,1\,1\,0\,1\,0_{(2)}
\end{aligned}$$

No. of messages = **No. of 1's in distance**

Introducing systematic element placement

- Efficiency: ID | | **same** |
- **Load balance**: ID | **different** | |

e.g.



Introducing systematic element placement

- Efficiency:

ID		**same**

- Load balance:

ID	**different**	

- **Array property**:

Index	**same**	**different**

e.g. sequential access

$$6 = 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0_{(2)}$$
$$7 = 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1_{(2)}$$
$$8 = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0_{(2)}$$

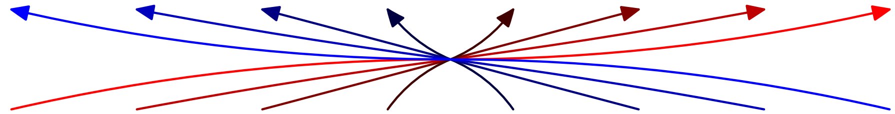
Introducing systematic element placement

• Efficiency:	ID		**same**
• Load balance:	ID	**different**	
• Array property:	Index	**same**	**different**

Introducing systematic element placement

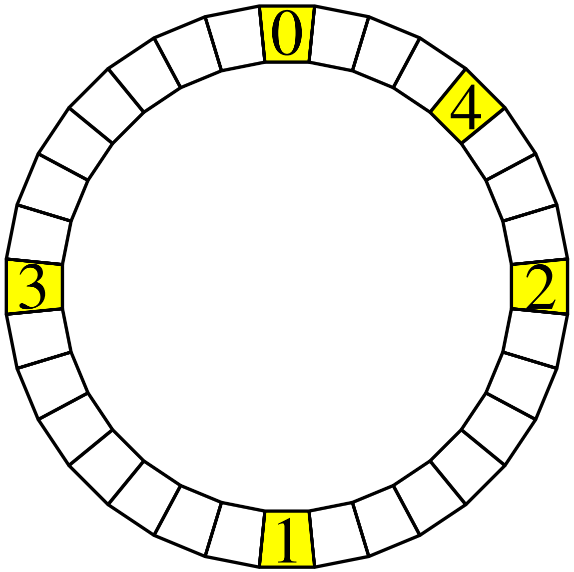
- Efficiency: ID | | **same** |
- Load balance: ID | **different** | |

Reverse bit order



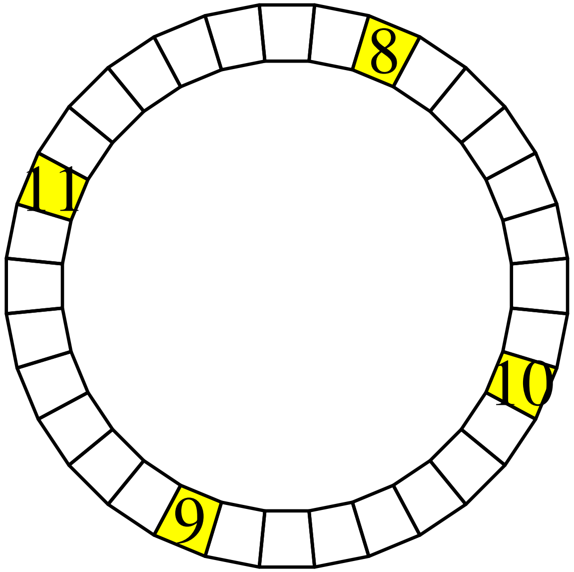
- Array property: Index | **same** | **different** |

Reverse bit order placement



Index		ID
$0 = 00000_{(2)}$	rev.	$00000_{(2)} = 0$
$1 = 00001_{(2)}$	$\mapsto$	$10000_{(2)} = 16$
$2 = 00010_{(2)}$		$01000_{(2)} = 8$
$3 = 00011_{(2)}$		$11000_{(2)} = 24$
$4 = 00100_{(2)}$		$00100_{(2)} = 4$
$\vdots$		$\vdots$

Load balance

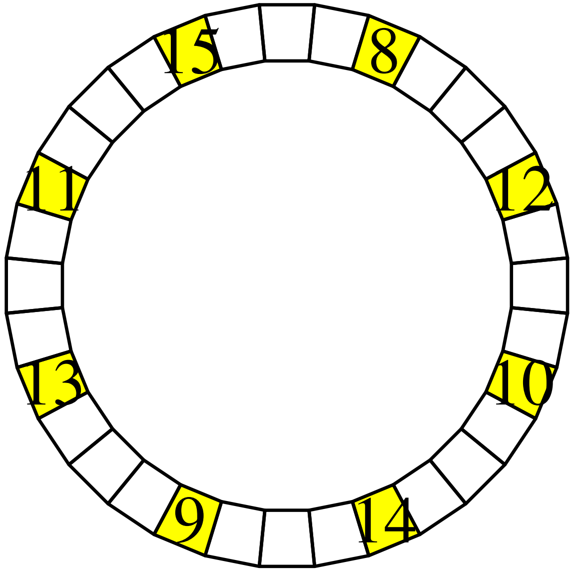


e.g. $[8, 12)$

- **Sequential elements ⇒ different nodes**

  $\because$ Indices $[a2^k, (a+1)2^k) \Rightarrow$ equal interval IDs.

Load balance



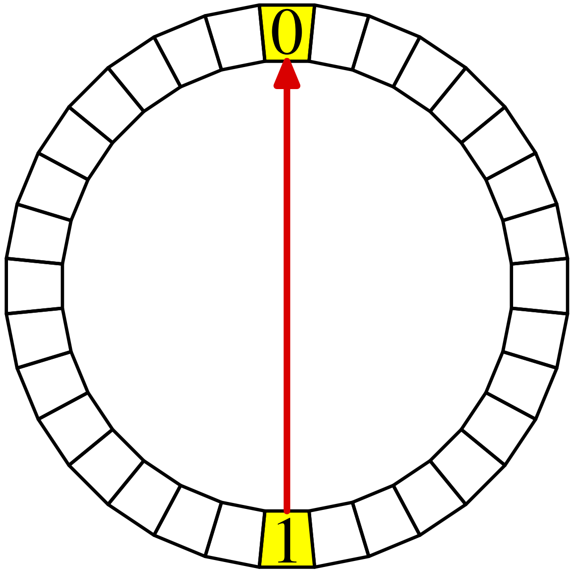
e.g. [8, 12)

[8, 16)

- **Sequential elements ⇒ different nodes**

  $\because$ Indices $[a2^k, (a+1)2^k) \Rightarrow$ equal interval IDs.

Efficiency of one by one element access



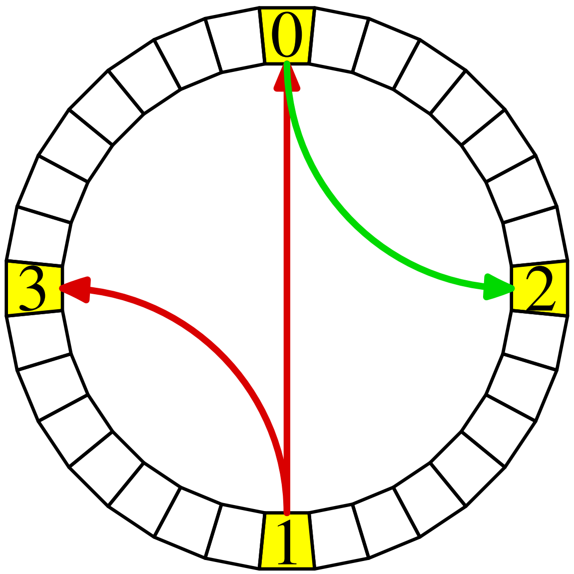
e.g.

Indices	Max hop
[0,2)	1

- **Independent of No. of nodes**
- **Closer indices ⇒ less messages**

$\because$ in $[a2^k, (a+1)2^k) \Rightarrow$ max k hop

Efficiency of one by one element access



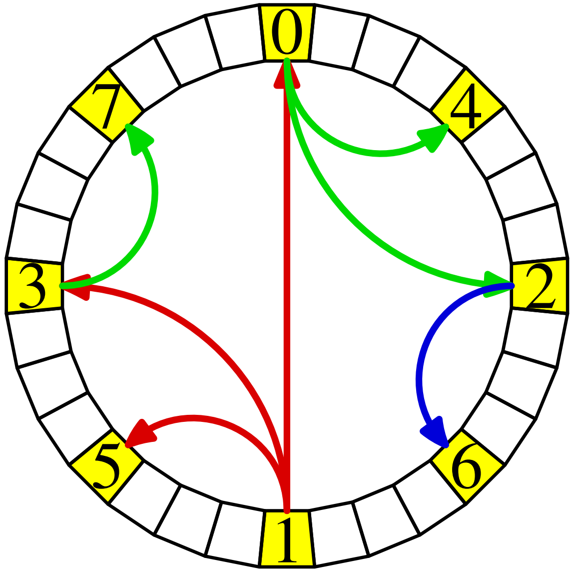
e.g.

Indices	Max hop
[0,2)	1
[0,4)	2

- **Independent of No. of nodes**
- **Closer indices ⇒ less messages**

$\because$ in $[a2^k, (a+1)2^k) \Rightarrow$ max k hop

Efficiency of one by one element access



e.g.

Indices	Max hop
[0,2)	1
[0,4)	2
[0,8)	3

- **Independent of No. of nodes**
- **Closer indices ⇒ less messages**

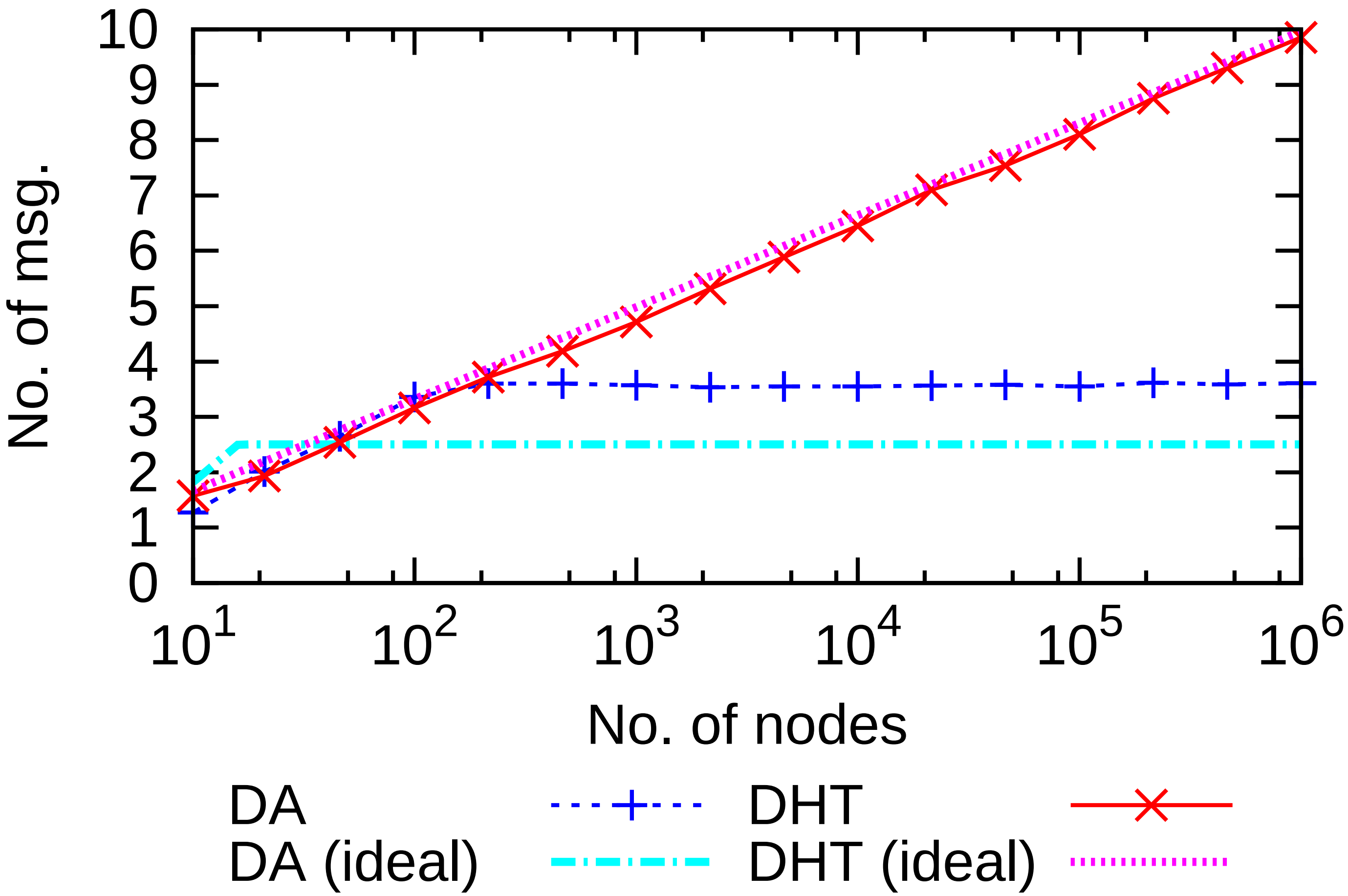
$\because$ in $[a2^k, (a+1)2^k) \Rightarrow$ max k hop

Improvements

- n nodes and w target elements

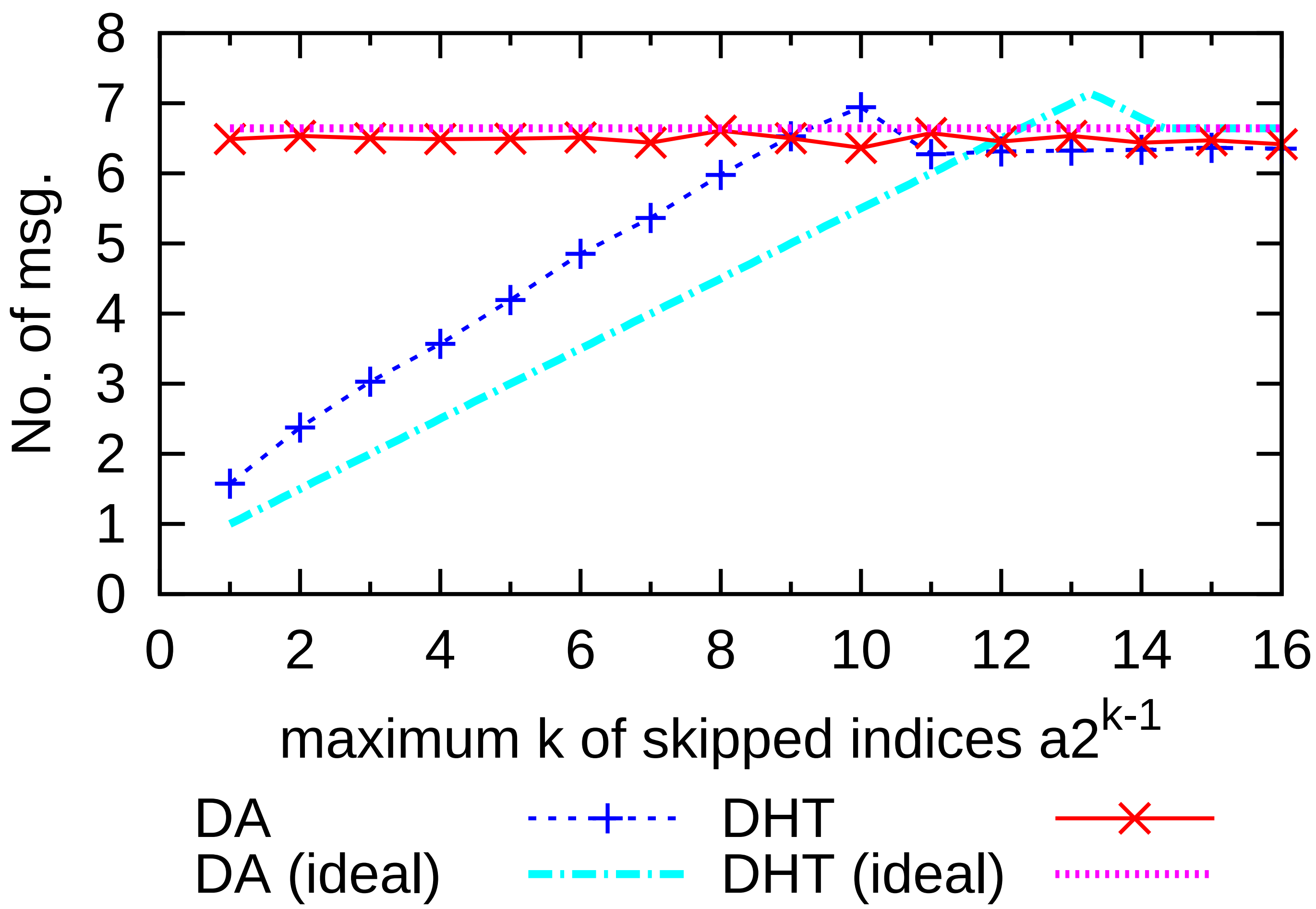
	DHT	DA
Sequential access	$O(w \log n)$	$O(w + \log n)$
Search on a sorted array	$O(\log^2 n)$	$O(\log n)$

Simulation of non-ideal env. (No. of nodes)



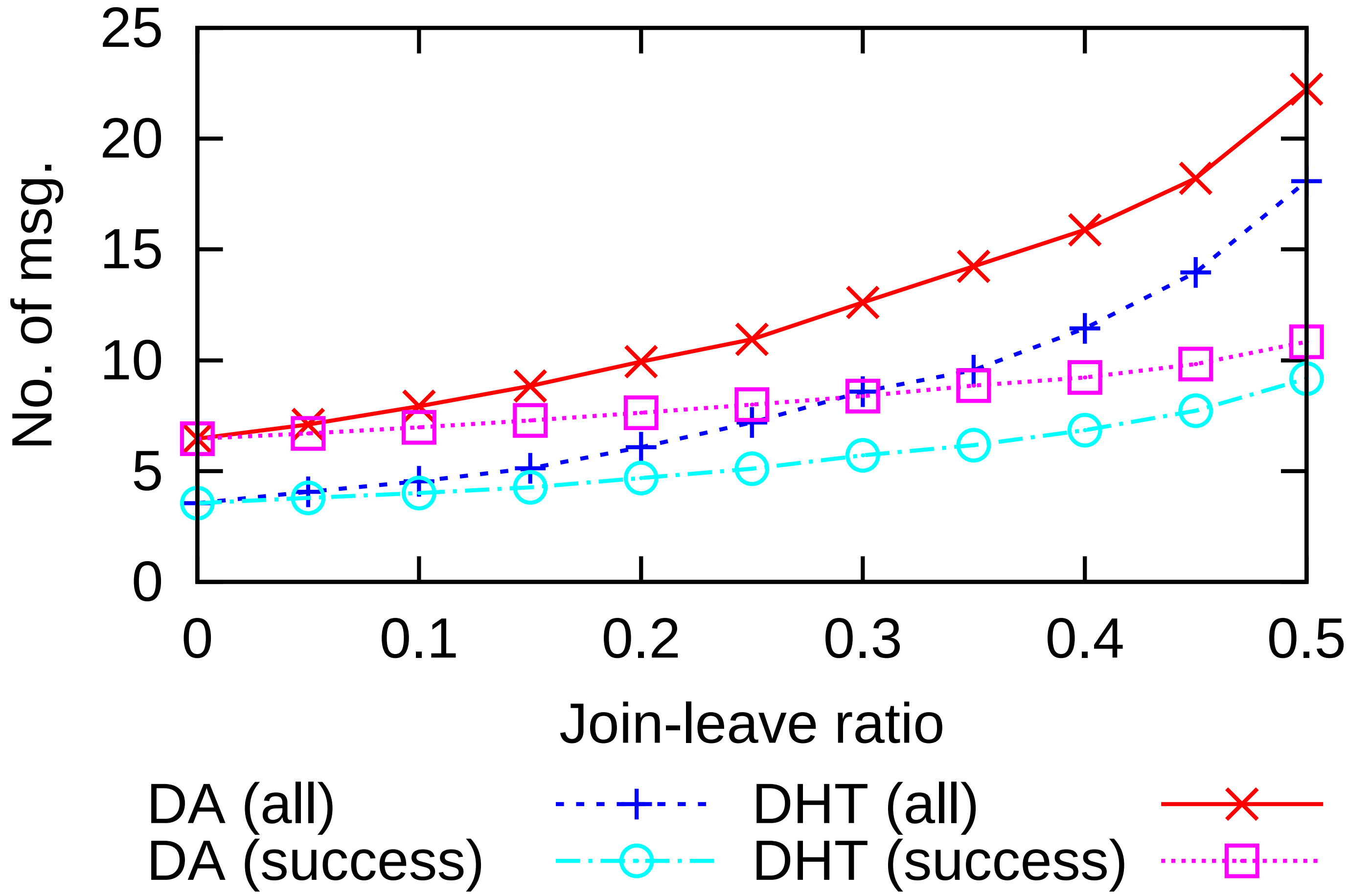
- One by one element access

Simulation of non-ideal env. (index distance)



- log (average index distance) $\propto x$-axis

Simulation of non-ideal env. (dynamic)



Conclusion

- **Efficient arrays in P2P environments**

Future work

- DA for other P2P networks
- Other P2P data structures