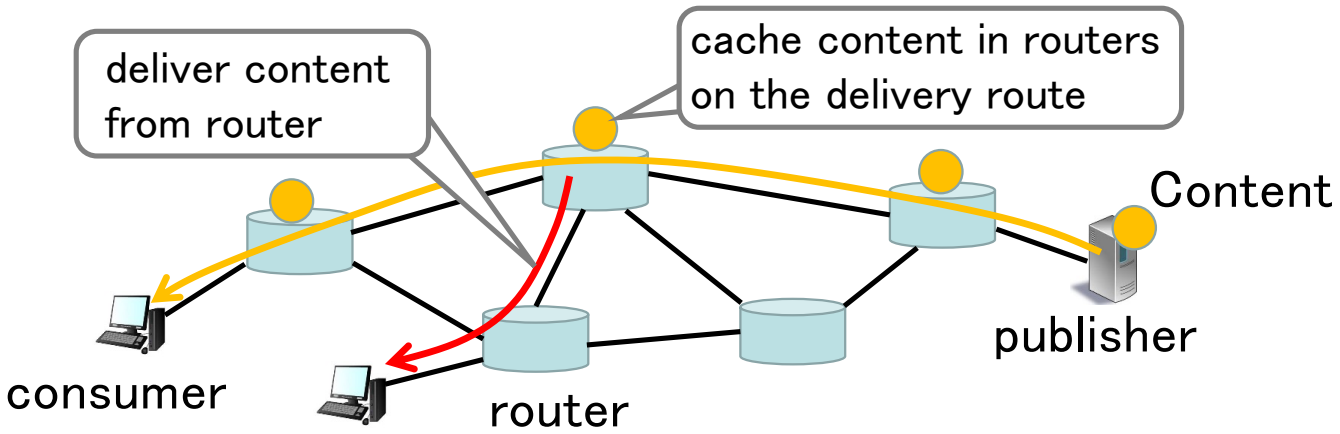
Name Management Using IOTA in ICN

Teppei Okada, Noriaki Kamiyama
Ritsumeikan University, Japan

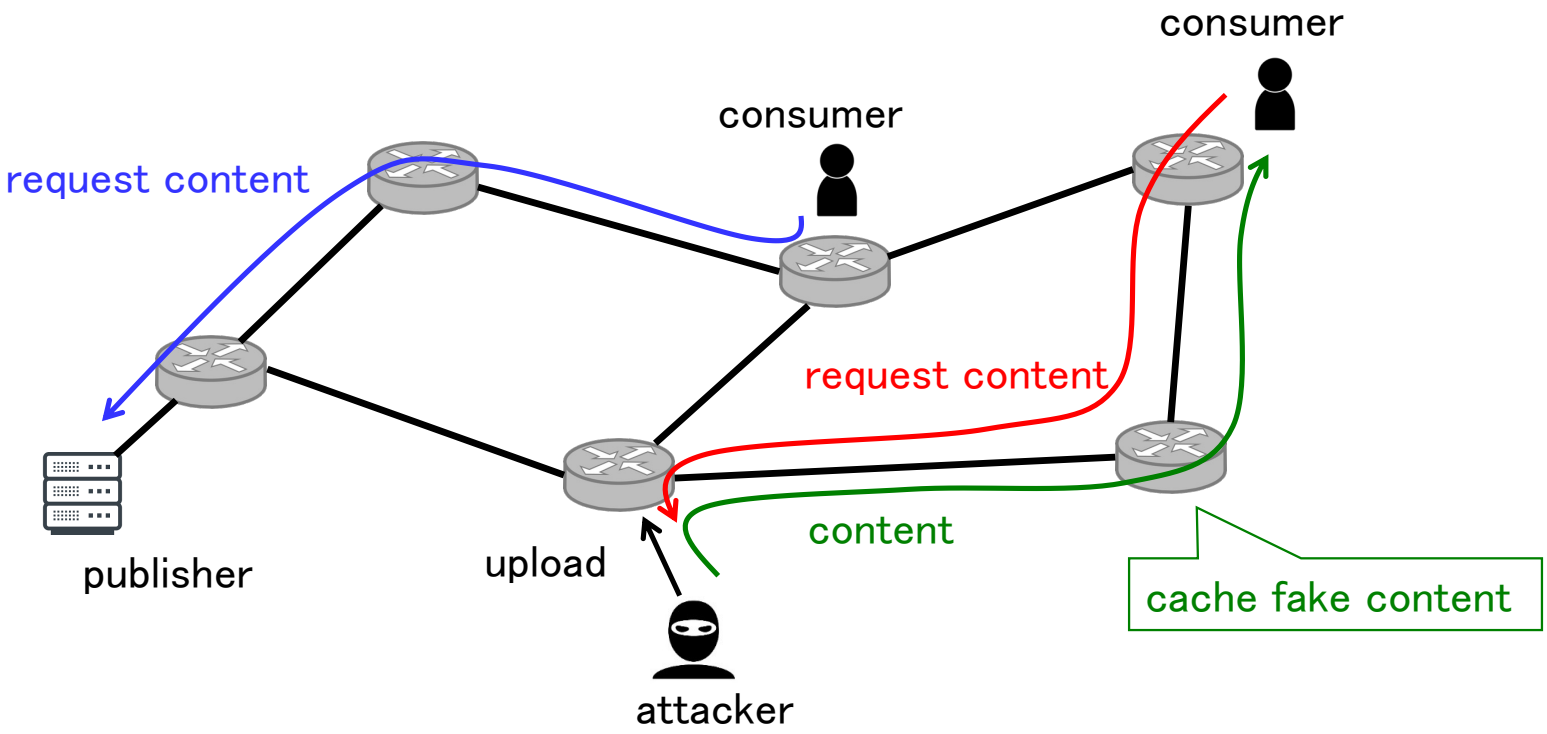
Background（1/3）

- ICN (information-centric networking)
  - Publishers forward contents to the consumers
    - based on the name of the requested content
  - Contents are cached and delivered to the router



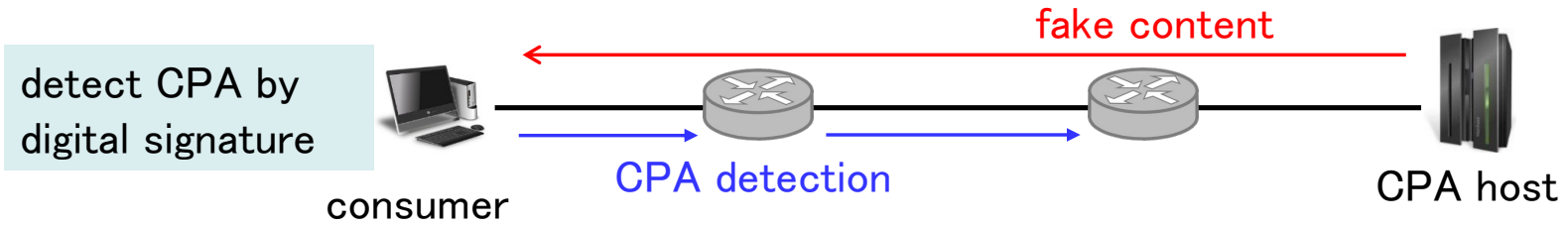
Background (2/3)

- In ICN, anyone can upload content as a publisher
- CPA (content poisoning attack)
  - Attackers degrade the cache efficiency by uploading fake content under a real name posting as a legitimate publisher



Background (3/3)

- Consumers determine the legitimacy of content using digital signatures with public keys, and alert routers of unjustified content upon detection※1



- fake-CPA
  - use public keys of fabricated content to generate digital signatures, so it's difficult to detect
- spoofed fake-CPA
  - injects fake contents that pretends to be real and popular contents into the cache
  - the staff of the certification authority (CA) that manages the public key colludes with the attacker to rewrite the legitimate publisher's public key to the attacker's
  - the impact of this attack will be significant if popular content with many accesses is spoofed

※1: W. Cui, et al., "Feedback-Based Content Poisoning Mitigation in Named Data Networking", IEEE ISCC 2018.

Purpose of this work

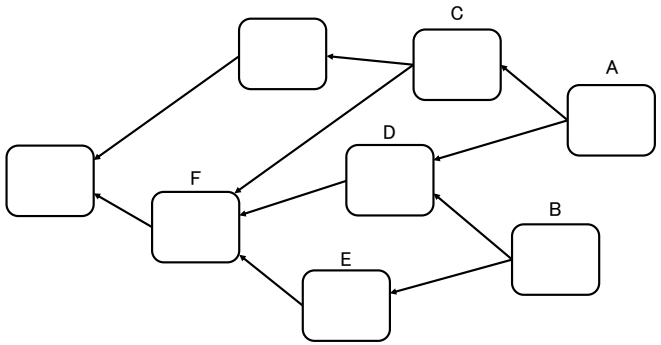
- spoofed fake-CPA
  - occurs when a public key is managed by only one authority, such as a CA

- Purpose
  - prevent it by managing content names using IOTA which is difficult to tamper with registered data

- Proposed method may increase the search time and the amount of memory requirement on IOTA
  - compare them for each of four search methods for transactions

IOTA

- One of the Distributed Ledger Technologies
  - blockchain
    - Data (transaction) are put together into blocks and managed in blocks
      - due to restriction of block size, the delay of process increases
    - needs enormous electric power for calculation (PoW: proof of work)
  - IOTA
    - manages by transaction, and can process fast
    - doesn't require as much computational power as blockchain



- New transactions select two of the unselected transactions, i.e., the tips
  - they form a DAG (directed acyclic graph)

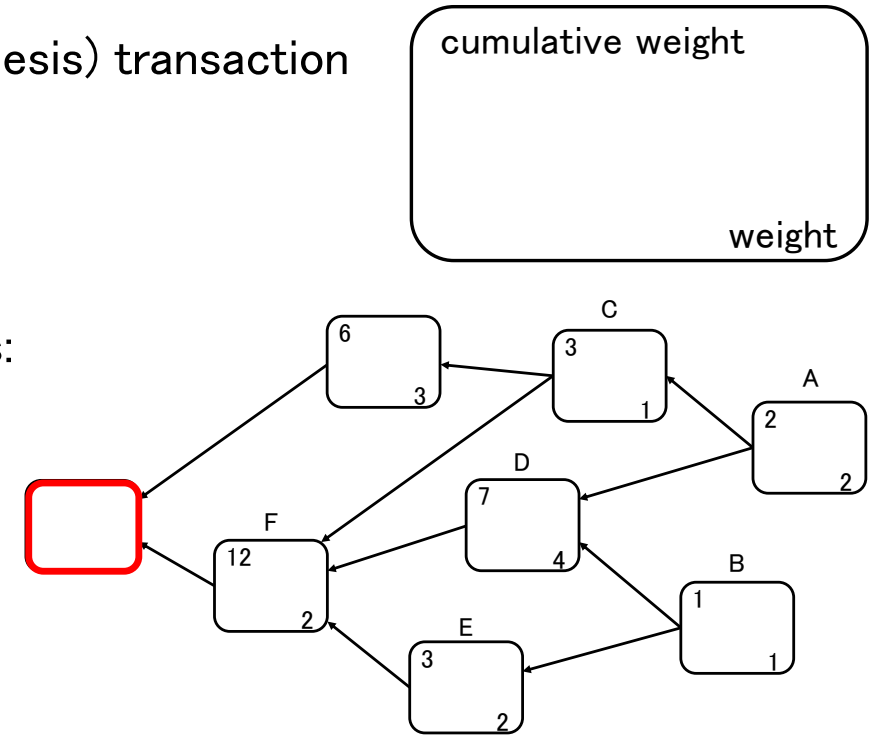
Tip selection algorithm (TSA)

- uniform random selection (URS)
  - select two transactions randomly from the existing tips
- uniform random walk (URW)
  - select tips with equal probability from the first (genesis) transaction
- weighted random walk (WRW)
  - select tips considering the weights
  - Transition probability P_{xy} from transaction y to x is:

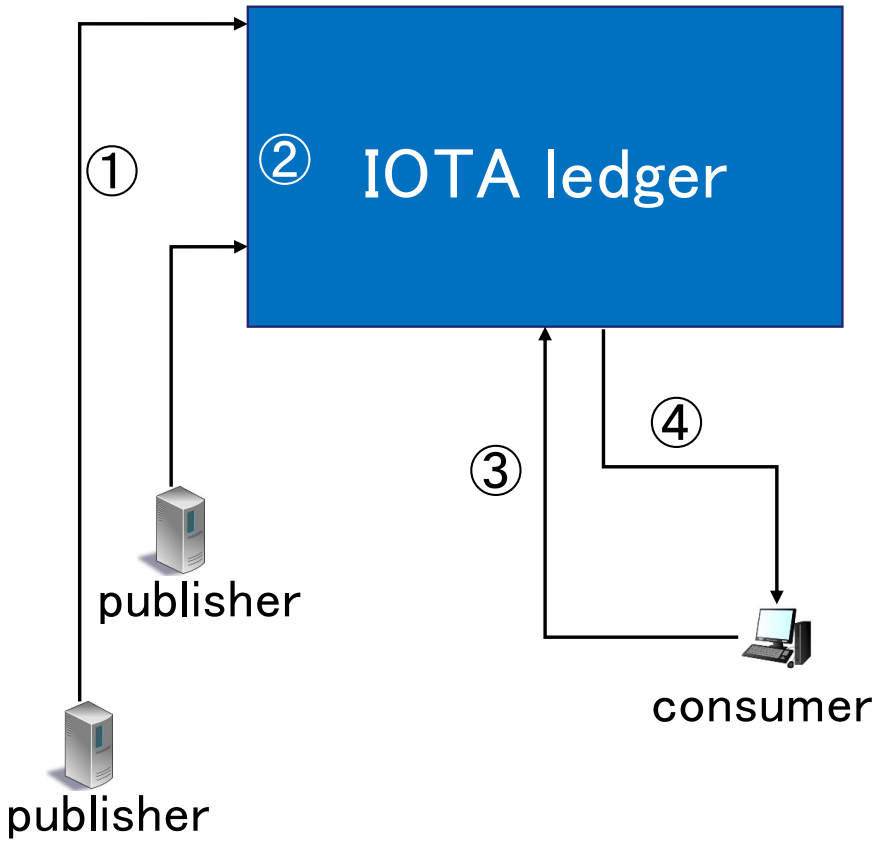
$$P_{xy} = \frac{e^{-\alpha(H_x - H_y)}}{\sum_{z:z\to x} e^{-\alpha(H_x - H_z)}}$$

H_x, H_y: cumulative weights of transaction x, y

$\alpha(\geq 0)$: parameter of the cumulative weight



Proposed Method



① When the publisher uploads the content, the transaction is registered in the IOTA ledger

- prefix of the content, ID, content name(= prefix + public key + digital signature)

② To prevent duplicate content names from being managed, search for the IOTA ledger by content prefix

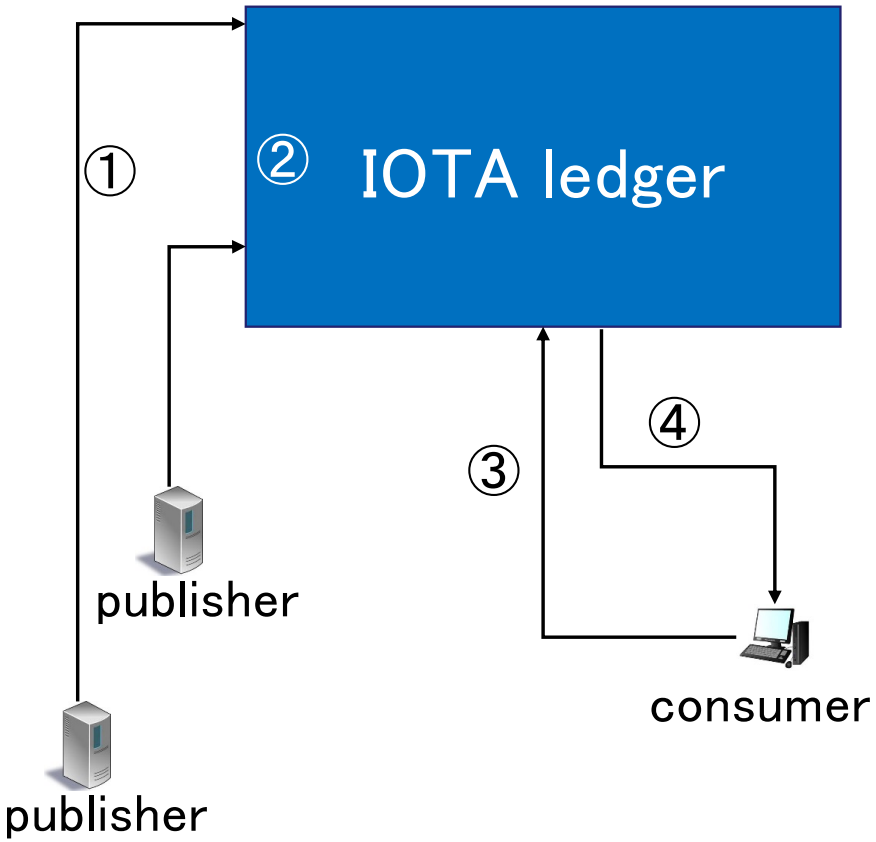
- if already exists, reject the registration; otherwise, register it

③ Consumer requests the prefix of the content and the transaction is searched in the IOTA ledger

④ Replies to the consumer with the name of the content that have hit

- Consumer sends an interest with that content name and requests the content

when to search for transactions



- Search for the corresponding transaction in the DAG at two different times

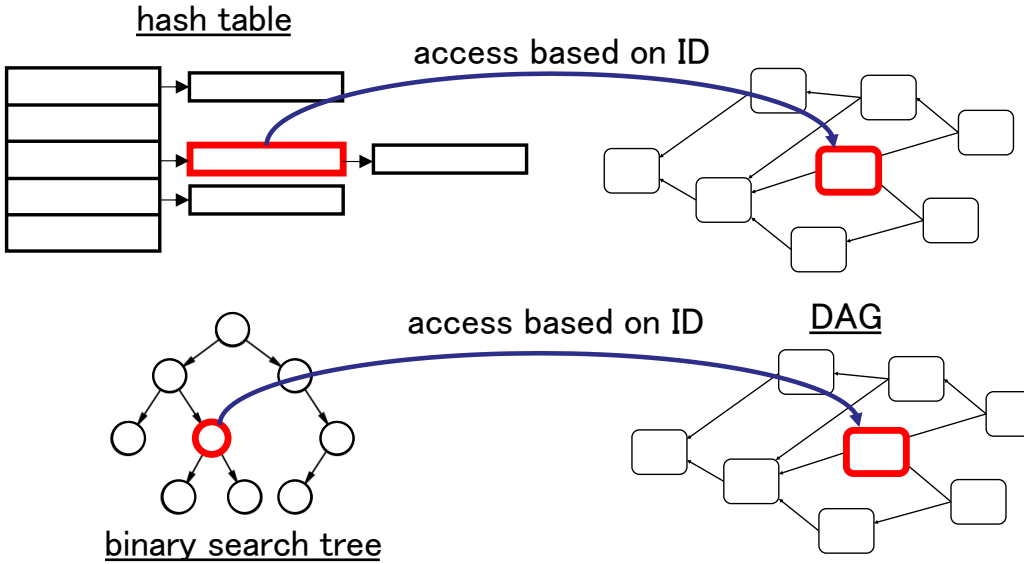
② when the content name is registered by the publisher

③ when the name is resolved by the consumer

Transaction search method in DAG

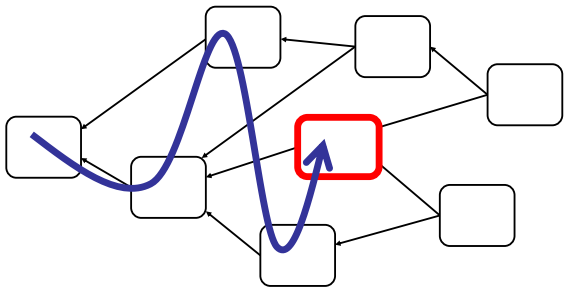
- Four search methods
  - **hash-chain method (hash)**
  - **binary search tree (bst)**

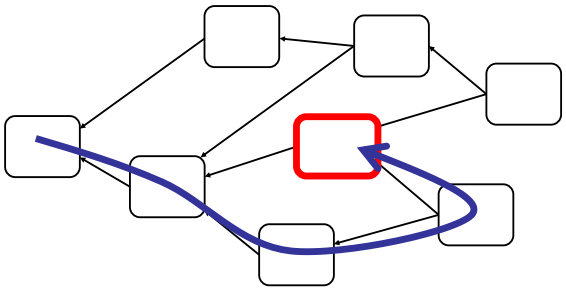
manages prefixes and ID in hash table or binary tree, and accesses DAG directly



  - **breadth-first search (bfs)**
  - **depth-first search (dfs)**

content name is recorded in transaction, so the search is complete when found


breadth-first search


depth-first search

Performance evaluation

- Compare evaluation items bellow among the four search methods by computer simulation

- Search time
  - Mean, median and the 95th percentile of the measured value
    - Content name registration
    - Name resolution

- Memory requirement
  - compare among hash-chain method, bst and DAG
    - bfs and dfs are summarized as a DAG (data is directly managed on a DAG)

Evaluation condition

URS: uniform random selection
URW: unweighted random selection
WRW: weighted random selection

- Content name registration
  - Set of the content names
    - 7,131 domains displayed web pages without errors out of the top 8,000 accessed web pages published on Alexa※1 in November 2017
  - Number of transactions N_t: 100, 1,000, 7,131 (for URS, URW), 100, 1,000 (for WRW)
    - In WRW, the process was not completed when N_t = 7, 131
  - α in the transition probability P_{xy}: 0.1
  - Number of transactions generated per second: 50
  - Number of table buckets in the hash-chain method: 100

$$P_{xy} = \frac{e^{-\alpha(H_x - H_y)}}{\sum_{z:z \to x} e^{-\alpha(H_x - H_z)}}$$

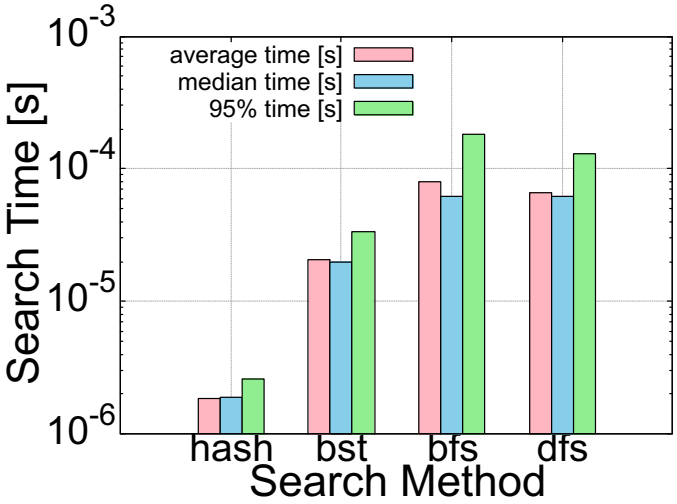
- Requests by consumers
  - Number of requests: 5,000
  - Number of requests per second: 50

※1: Alexa webpage, https://www.alexa.com/siteinfo

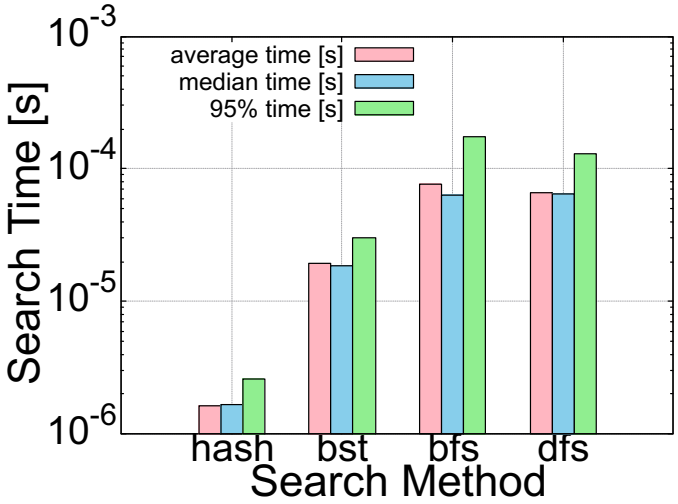
Search Time for Content Name Registration

Search time for each method (N_t=100)

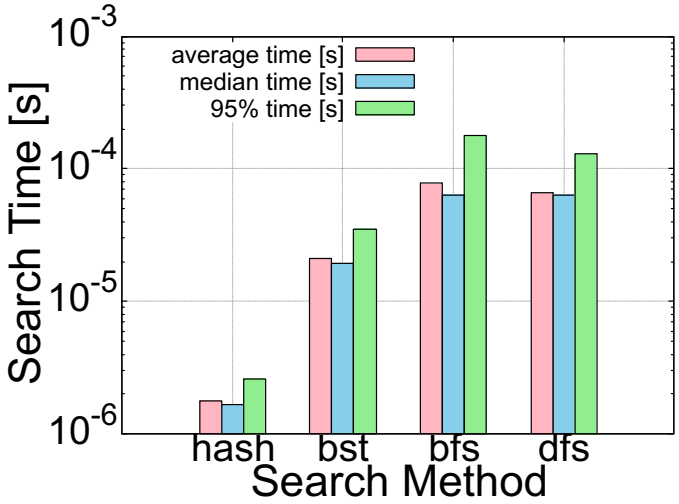
URS: uniform random selection
URW: unweighted random selection
WRW: weighted random selection



(a) URS



(b) URW



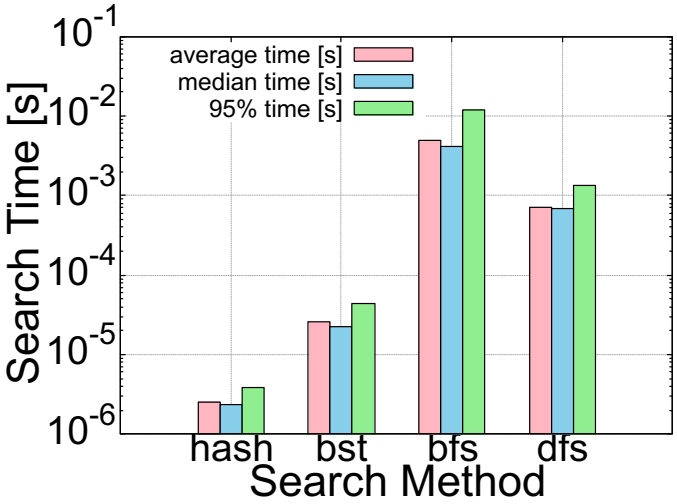
(c) WRW

- hash < bst < dfs < bfs
- Comparing TSA※, there is little difference in search time for any of the methods

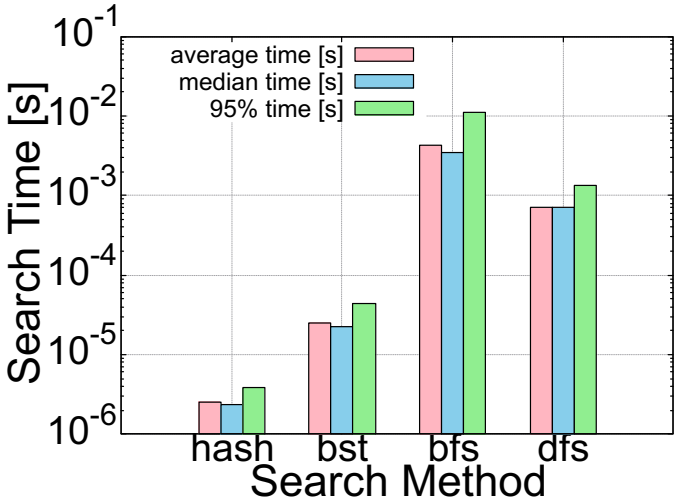
※tip selection algorithm

Search time for each method (N_t=1,000)

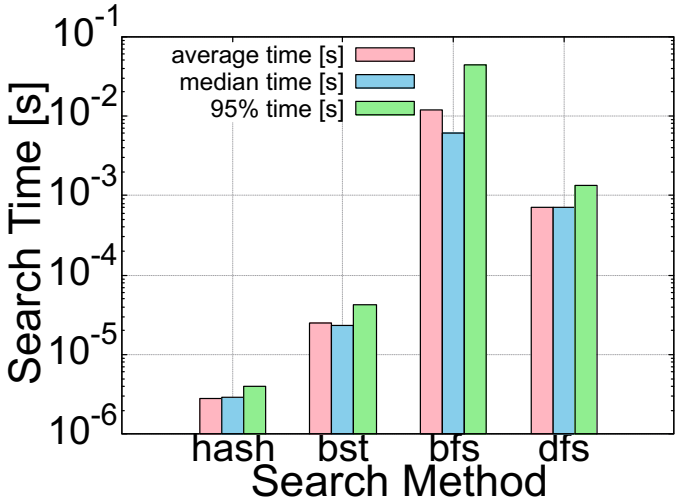
URS: uniform random selection
URW: unweighted random selection
WRW: weighted random selection



(a) URS



(b) URW



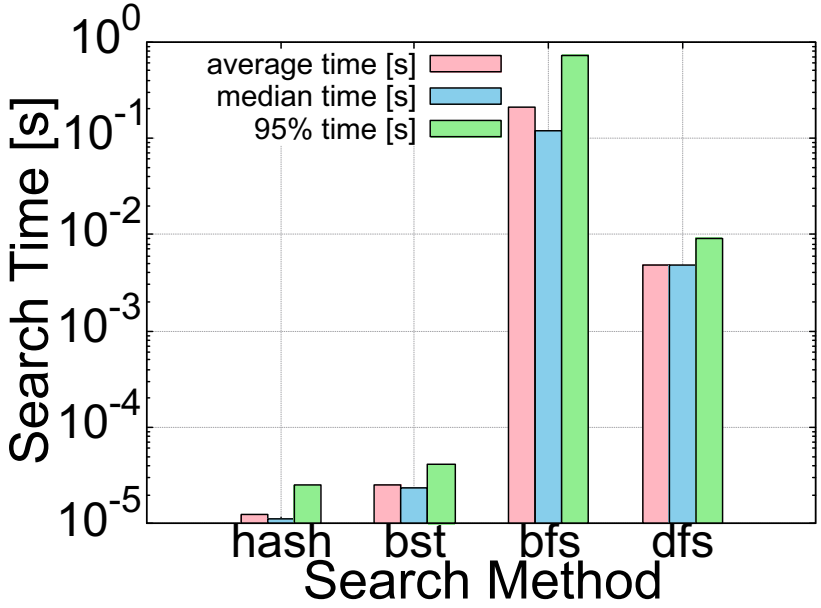
(c) WRW

- Compared to N_t=100, larger DAG size accentuates differences between hash, bst, and bfs, dfs

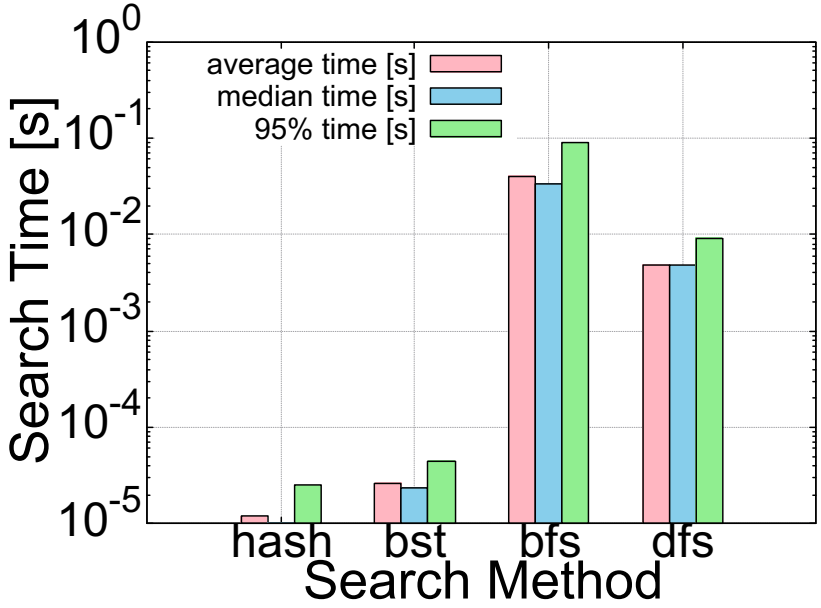
- bfs of WRW: long search time compared to URS and URW

Search time for each method (N_t=7,131)

URS: uniform random selection
URW: unweighted random selection



(a) URS



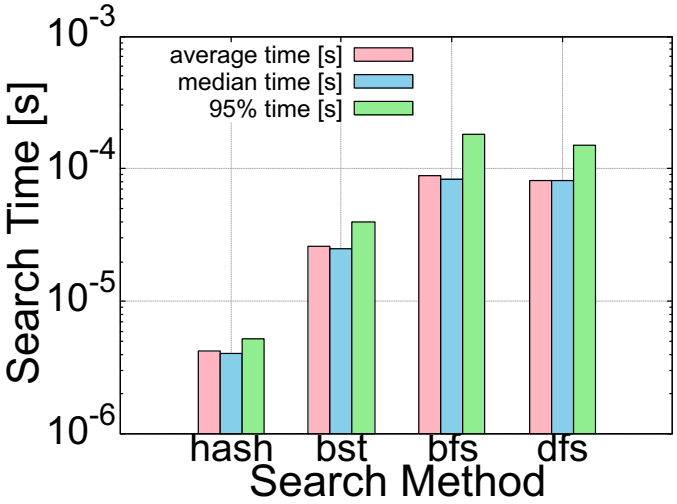
(b) URW

- URS spent more time in search than URW in bfs

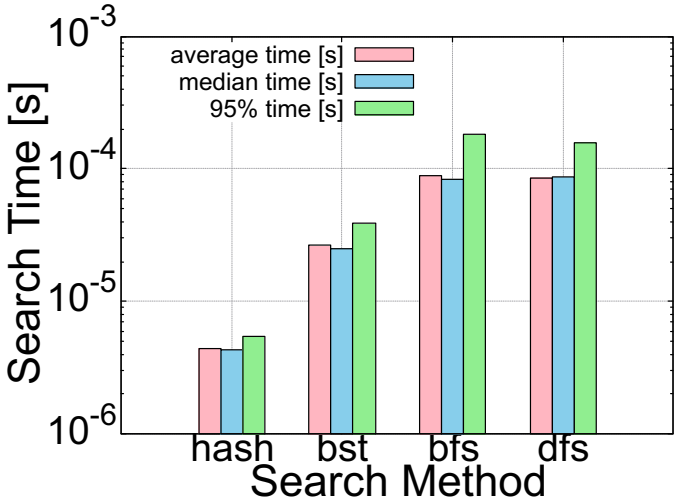
Search Time for Name Resolution

Search time for each method (N_t=100)

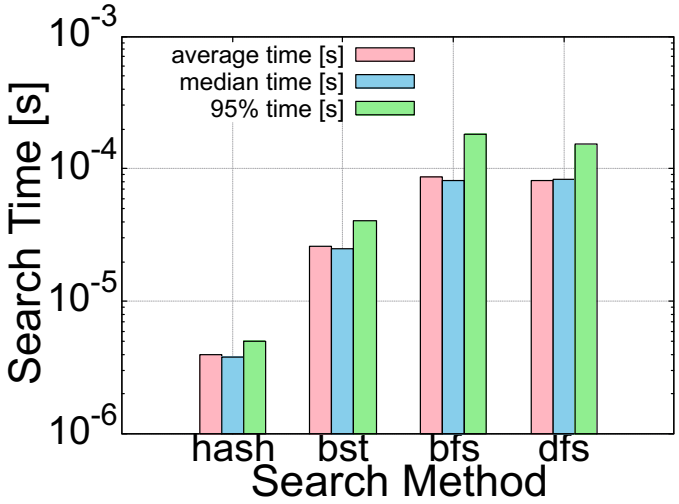
URS: uniform random selection
URW: unweighted random selection
WRW: weighted random selection



(a) URS



(b) URW

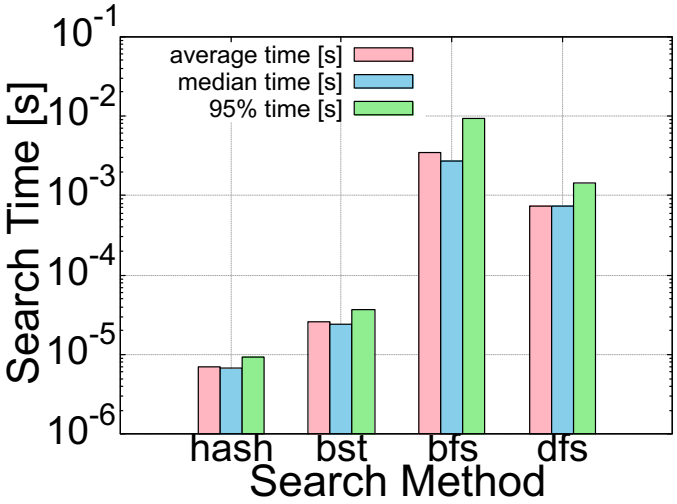


(c) WRW

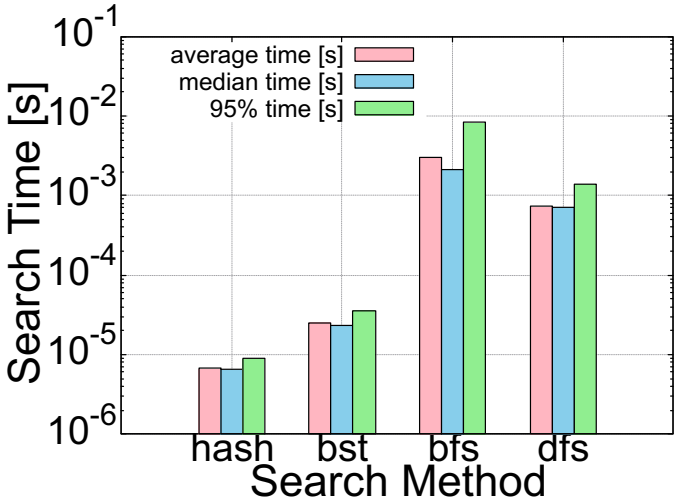
- hash < bst < dfs < bfs
- Due to the small number of transactions, the search time difference based on TSA is small

Search time for each method (N_t=1,000)

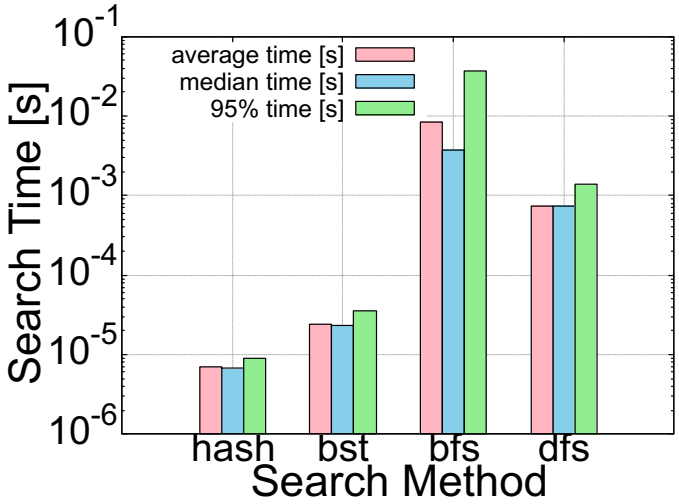
URS: uniform random selection
URW: unweighted random selection
WRW: weighted random selection



(a) URS



(b) URW

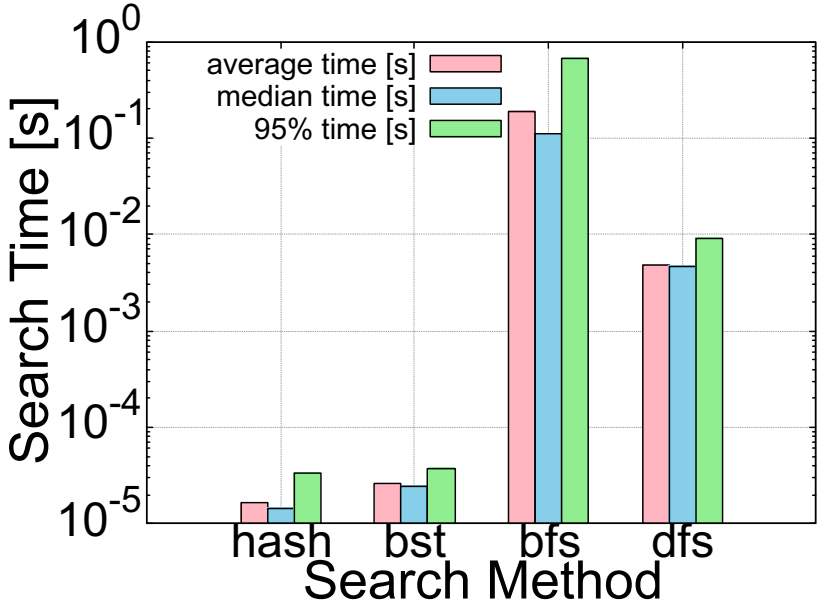


(c) WRW

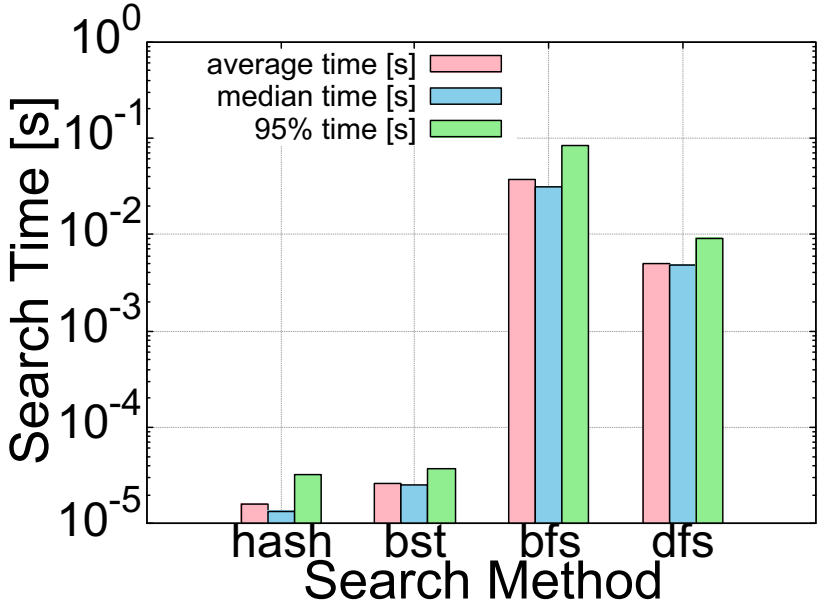
- Large difference between hash, bst, managed by tables, and bfs, dfs, directly managed by the DAG

Search time for each method (N_t=7,131)

URS: uniform random selection
URW: unweighted random selection



(a) URS

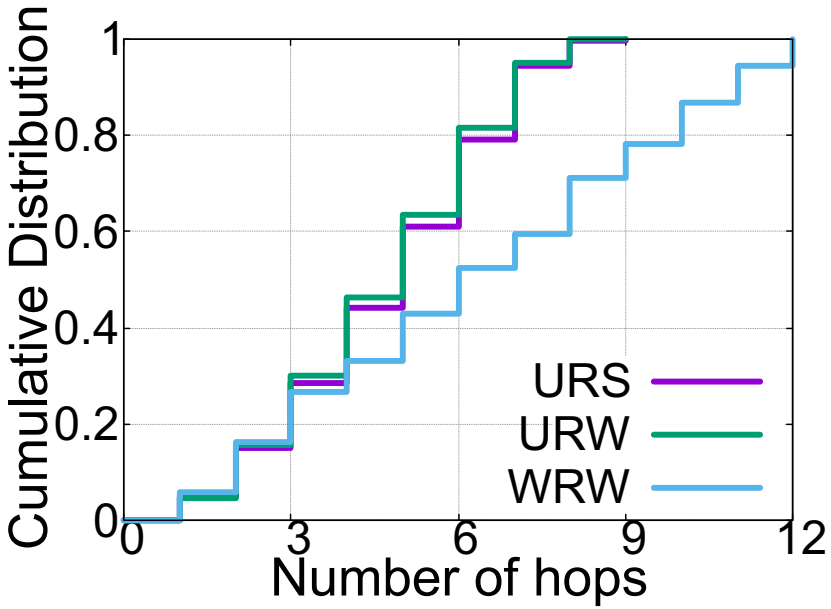


(b) URW

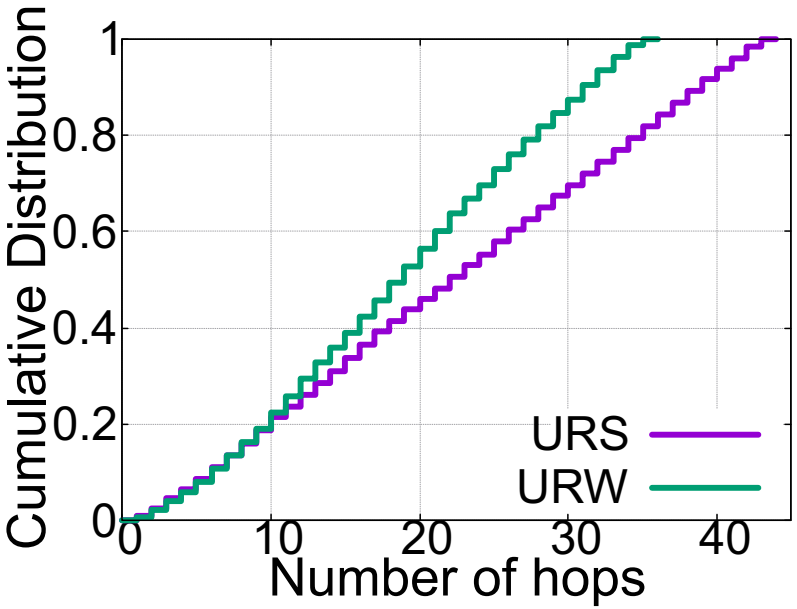
- URS > URW in bfs

Hop counts from genesis transaction on the DAG

URS: uniform random selection, URW: unweighted random selection, WRW: weighted random selection



(a) N_t=1,000

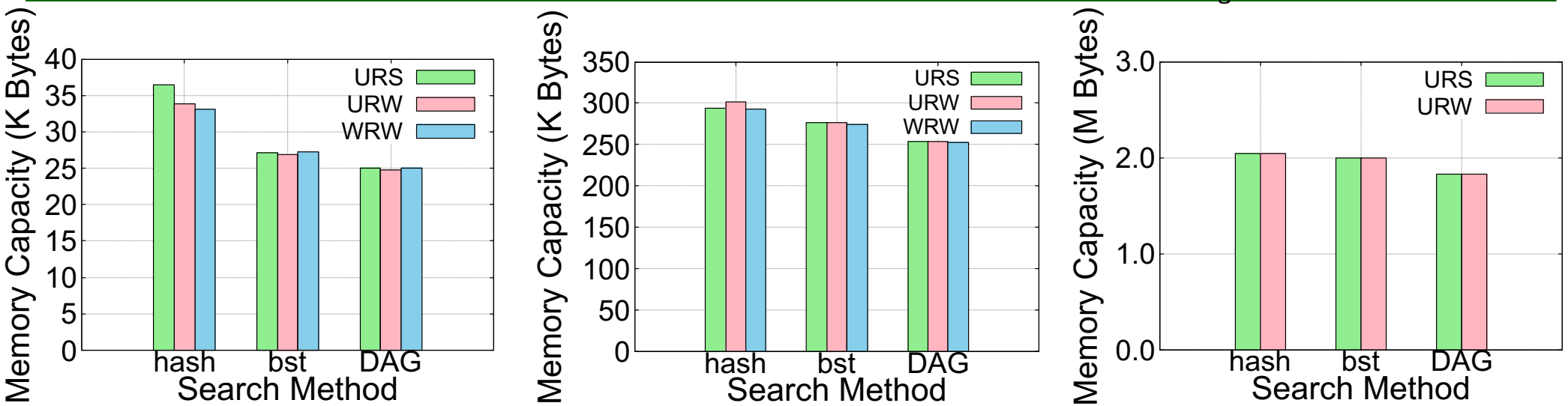


(b) N_t=7,131

- For N_t=1,000, WRW had more transactions with many hops than others
- For N_t=7,131, URS had more transactions with more hops than URW

Amount of Memory for each method

URS: uniform random selection
URW: unweighted random selection
WRW: weighted random selection



(a) N_t=100 (b) N_t=1,000 (c) N_t=7,131

- hash > bst > DAG
- hash: the largest capacity bucket in the hash table is allocated for all
- bst: requires a node capacity equal to the number of transactions

- These results confirmed the trade-off between search time and memory for each method

Conclusion

- We proposed a method to manage content names in IOTA to prevent spoofed fake-CPA

- Comparison of search time and amount of memory requirement among four methods
  - search time: hash < bst < dfs < bfs
  - memory requirement: hash > bst > DAG

- Trade-off between search time and memory
  - use **hash** or **bst** which requires short search time when the search time is important
  - use **dfs** or **bfs** which requires less memory when the amount of memory is important