SMIC: Subflow-level Multi-path Interest Control for Information Centric Networking

Junghwan Song
Seoul National University

Munyoung Lee
Electronics and Telecommunications Research Institute

Ted "Taekyoung" Kwon
Seoul National University

Outline

- Introduction
- SMIC design
- Evaluation
- Conclusion

Introduction

Why we need multi-path Interest control?

- ICN inherently has a chance to exploit multiple paths (subflows) for a flow
  - An FIB entry can have multiple outfaces
  - ICN forwarders can choose different outfaces for consequent Interests of a consumer
  - Forwarders can also choose multiple outfaces for an Interest

- Congestion controls for multiple paths are different from those of single path
  - The number of congestion windows
  - Path selection

Subflow-level Interest control

- Most of window-based schemes have a congestion window per a flow
  - Due to difficulty of identifying each path in ICN

➔ We propose Subflow-level Multi-path Interest Control (SMIC)
  - Proposing **Interest window per subflow**
  - Introducing **path identification & forwarding scheme**
  - Providing **design consideration areas of multi-path** congestion control
  - **Evaluating SMIC** performance

SMIC design

Design criteria

- Challenging issues on designing multipath Interest control mechanism
  - i. Congestion control
    - How to control congestion of multiple subflows?
  - ii. Subflow identification & forwarding
    - How to identify multiple subflows, forward Interests through them?

- Our solution
  - Put **subflow-level congestion windows** for congestion control
  - Put **path identifier & trajectory identifier** for path identification & forwarding

Why subflow-level congestion window?

- Limits of single congestion window for multiple paths
  - i. A path experiencing frequent packet drops may decrease overall throughput
  - ii. It is hard to infer intensity of congestion using packet losses
    - Out-of-order packet delivery
    - Hard to identify a trajectory of packet
  - iii. We need to select a path when there is a packet to be sent

- Subflow-level congestion window can solve above issues

Multi-path window control algorithm

- Our subflow-level congestion control's goal
  - Provide friendliness with single-path flows
  - Get more throughput than when using single-path mechanism

- We introduce bottleneck-sharing subflow-aware window control
  - Basically based on MPTCP algorithm
  - However, MPTCP increases cwnd conservatively
    - For the worst case (all subflow shares a bottleneck link)
  - Improve performances by introducing aggressive window increase on non-bottleneck-sharing subflows

Bottleneck-sharing subflow detection

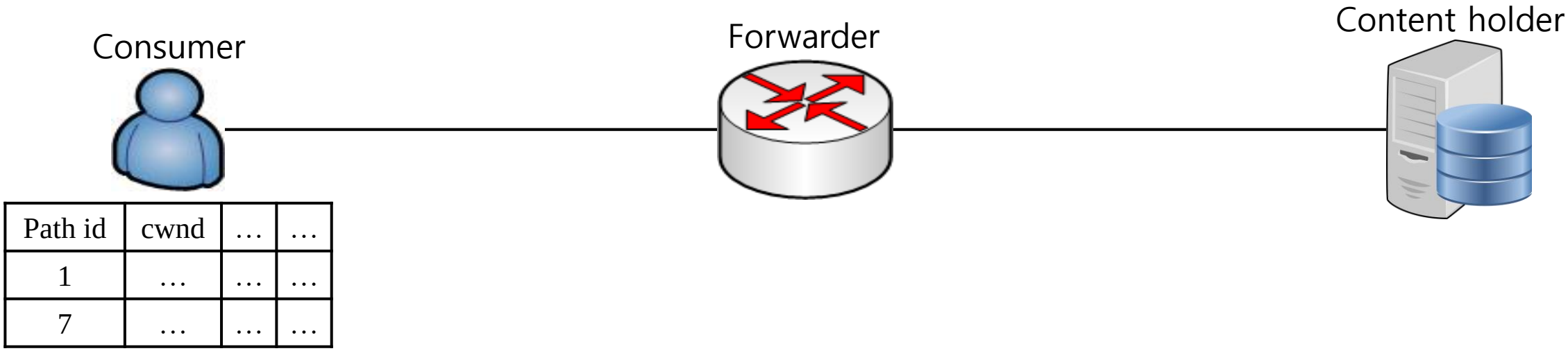
- We use two conditions
  - Timeout history
  - Estimated bottleneck bandwidth
- Consumer-side operations
  - Store a timeout history of each subflow
    - e.g. *n* timeouts times
  - Estimate bottleneck bandwidth of each subflow
    - e.g. using simple packet-pair algorithm
  - If similarities between two subflows exceed threshold *t*, regard them as bottleneck-sharing subflows

Path identification & forwarding

- Two requirements for realizing subflow-level congestion control
  - i. Identifying each subflow to manage subflow information
  - ii. Forwarding Interests to a specific subflow
- PathSwitching (ICN `17) satisfied requirements with path label, but
  - Path label grows up as hop count increases
  - Although encoding them, false positive or length problem still exist
- We introduce path identifier and trajectory identifier
  - Both of them are fixed-length identifiers in header
  - Path identifier in Interest header provides path identification & forwarding
  - Trajectory identifier in Data header guarantees one-on-one matching between path identifier and real path

SMIC operation

Initial phase

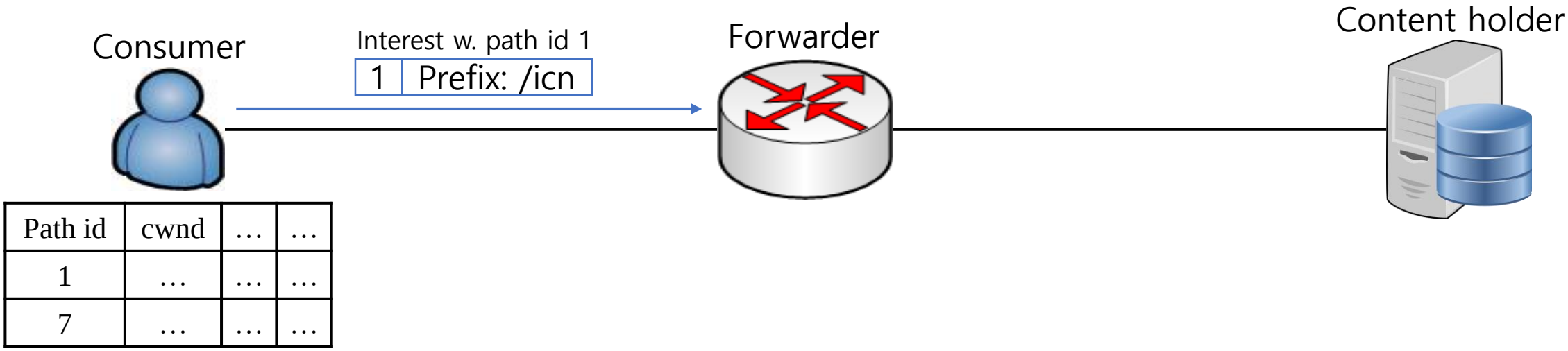


Path id	cwnd	…	…
1	…	…	…
7	…	…	…

- Consumer identifies subflows by path identifiers (path ids)
  - A path id is considered as a subflow
- Consumer sets per-subflow (per-path id) information
  - cwnd, # of on-the-fly Interests, etc.

SMIC operation

Sending Interest

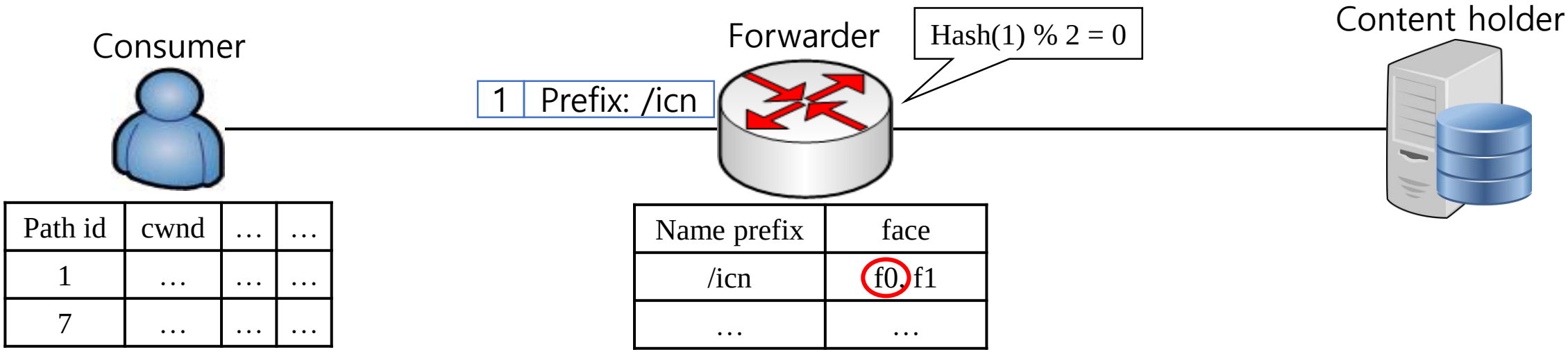


Path id	cwnd	…	…
1	…	…	…
7	…	…	…

- Consumer sends Interests with path ids
  - A new field 'path id' in Interest headers
- Interests are sent with path id *n* when
  - (cwnd of path id *n* > # of on-the-fly Interests of path id *n*)

SMIC operation

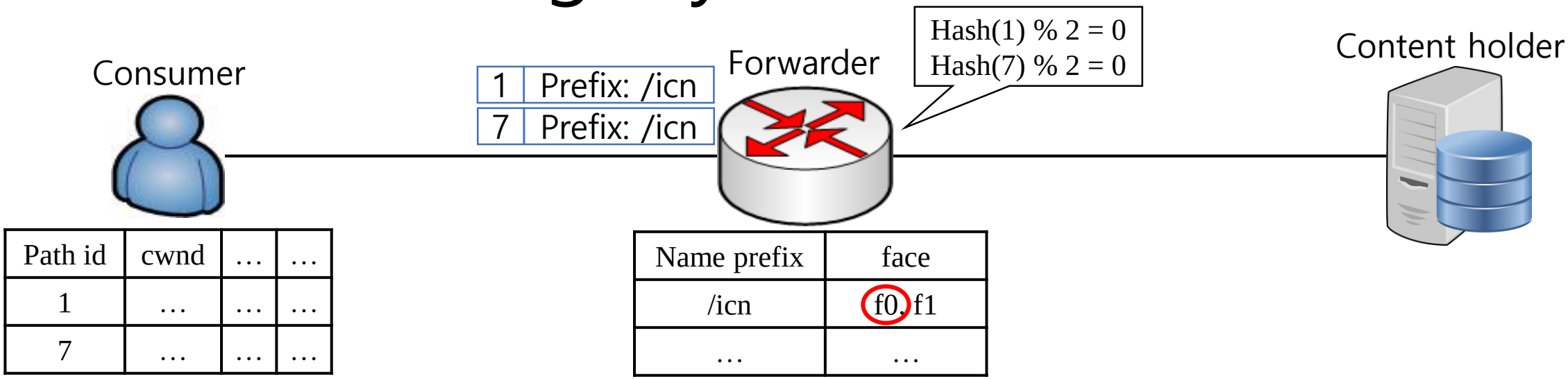
Forwarding Interest



- Forwarder forwards Interests according to their path id
  - e.g. if hash(*n*) modulo *m* = *k*, then forward to k^{th} outface
  - *n* : path id
  - *m* : # of outfaces on matched FIB entry
- Interests with same path id are forwarded to same path
  - Unless FIB entry or *m* do not change

SMIC operation

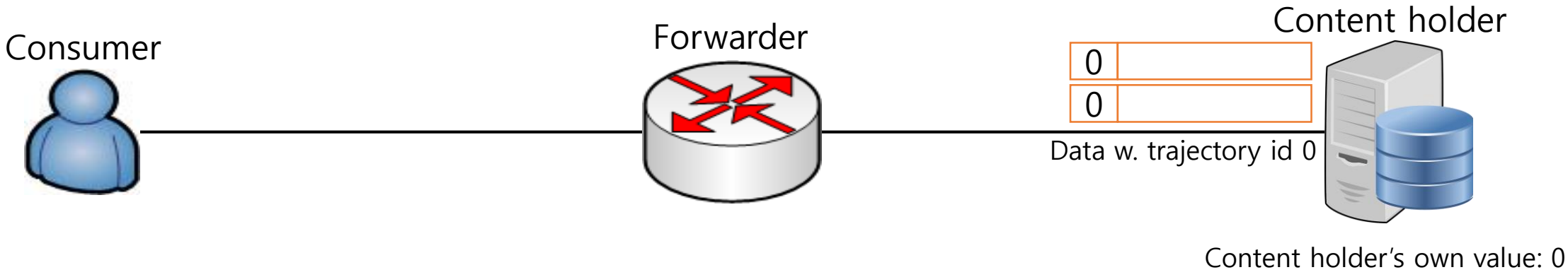
Path id ambiguity



- Interests with different path ids might be forwarded to same path
  - Due to hash collision or modulo *m*
- *N* path ids on same path result in
  - *N* times more aggressive path utilization

SMIC operation

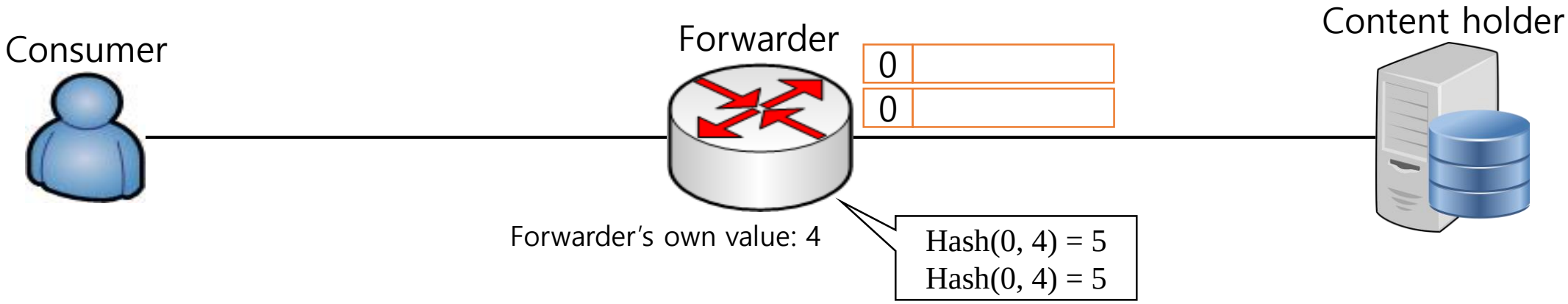
Data return



- Datas are created and trajectory ids are initialized
  - Initial trajectory id: Content holder's own value (hash of MAC addr, etc.)
- Datas are forwarded by breadcrumbs using PIT entries

SMIC operation

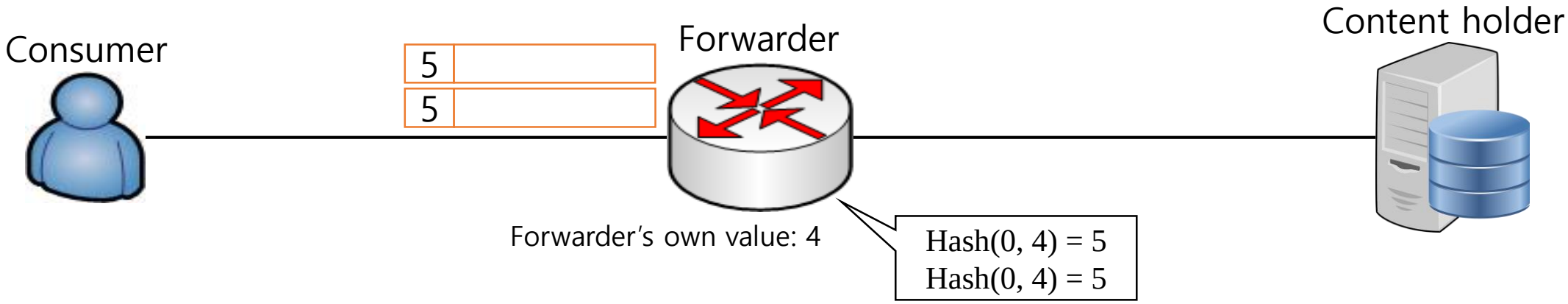
Trajectory id update



- Forwarder updates trajectory ids in Datas
  - Hash trajectory id with forwarder's own value
- Trajectory ids can guarantee their uniqueness on real paths, if
  - Size of trajectory id is sufficient
  - Good hash function is used

SMIC operation

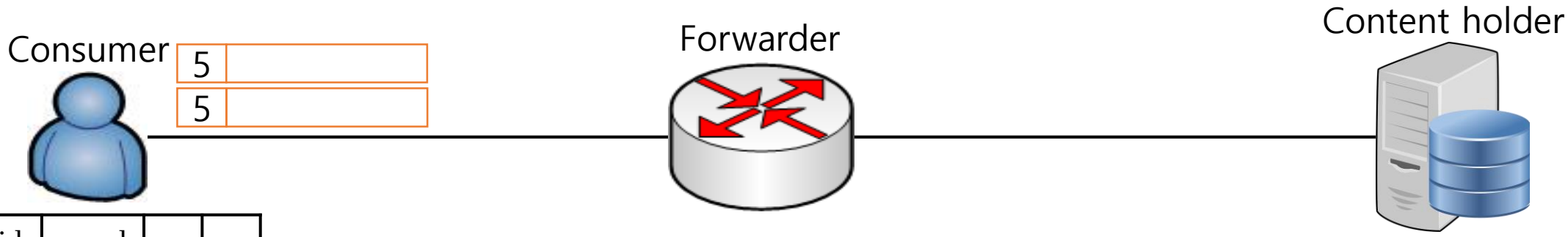
Trajectory id update



- Forwarder updates trajectory ids in Datas
  - Hash trajectory id with forwarder's own value
- Trajectory ids can guarantee their uniqueness on real paths, if
  - Size of trajectory id is sufficient
  - Good hash function is used

SMIC operation

Consumer actions



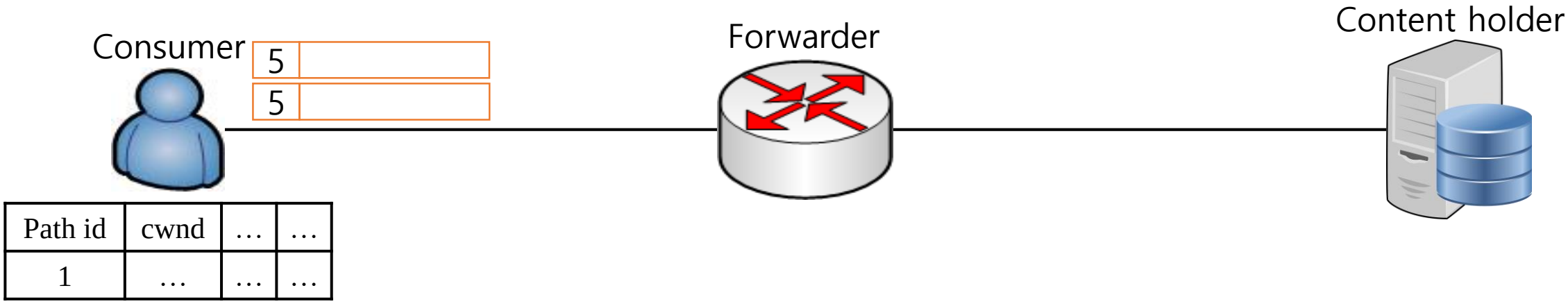
Path id	cwnd	…	…
1	…	…	…
7	…	…	…

Same trajectory id
➔ merge

- Consumer merges path ids with same trajectory id
  - Same trajectory id means Data packets traverse exactly same path
- Consumer changes subflow-relevant information
  - cwnd size, RTT, etc.

SMIC operation

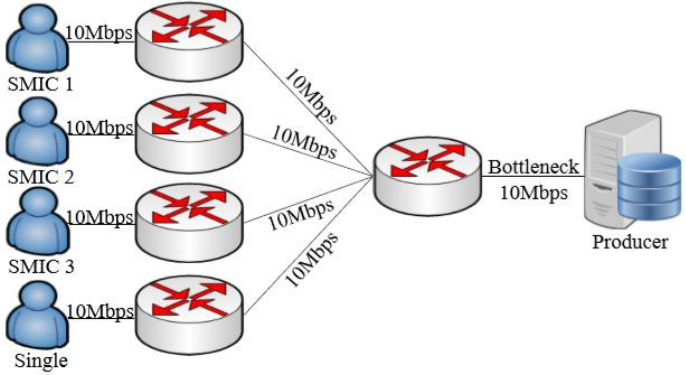
Consumer actions

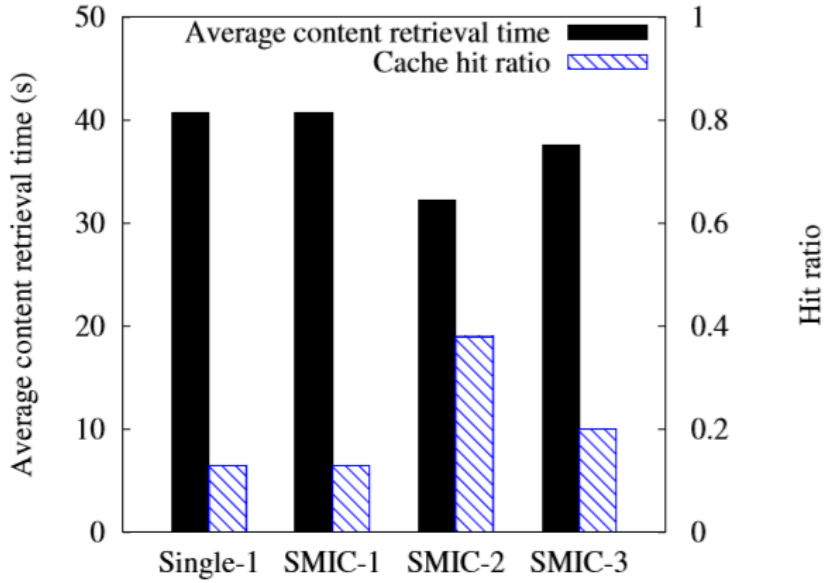


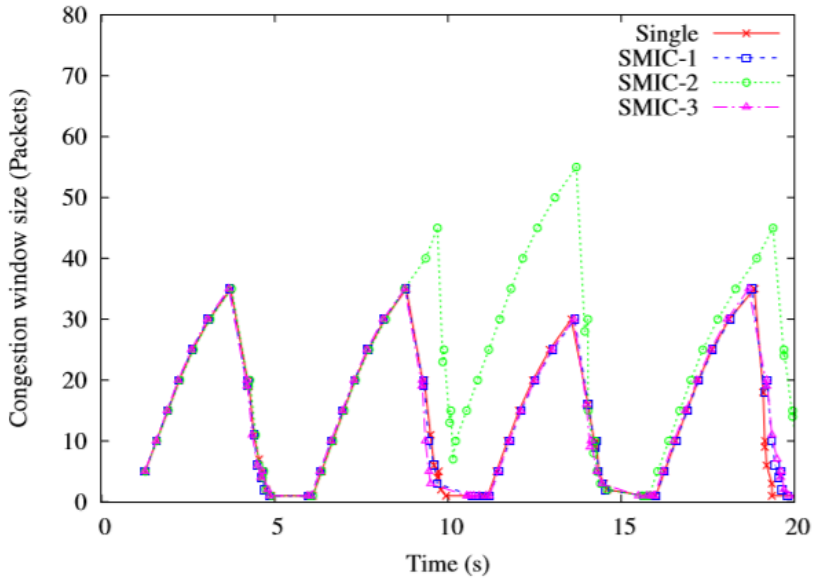
- Consumer merges path ids with same trajectory id
  - Same trajectory id means Data packets traverse exactly same path
- Consumer updates subflow-relevant information
  - cwnd size, RTT, etc.

Evaluation

Friendliness with single-path flows

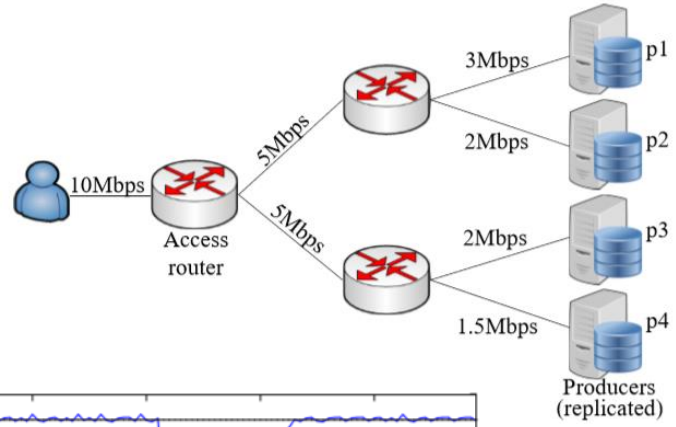


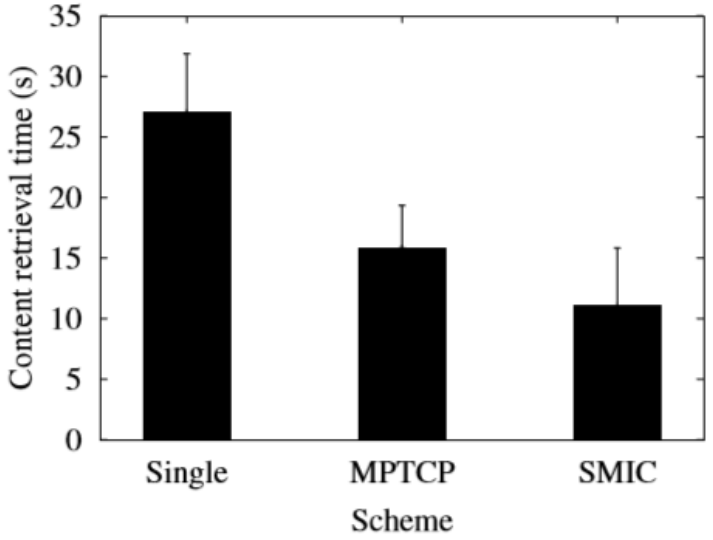


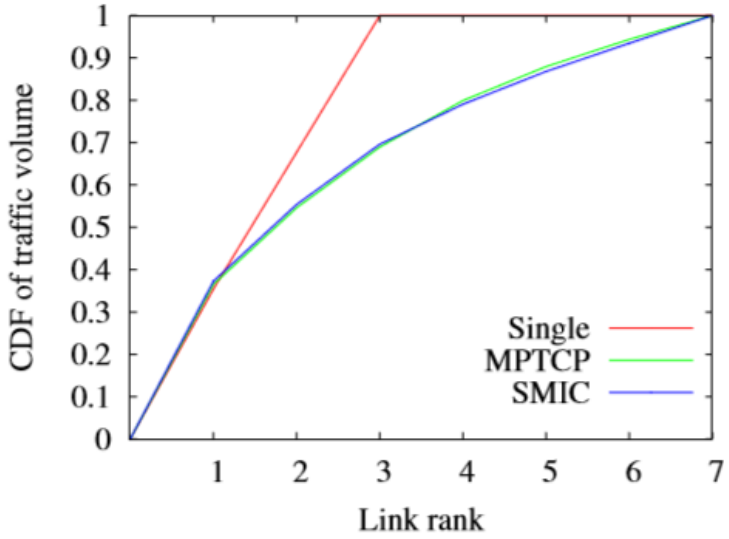


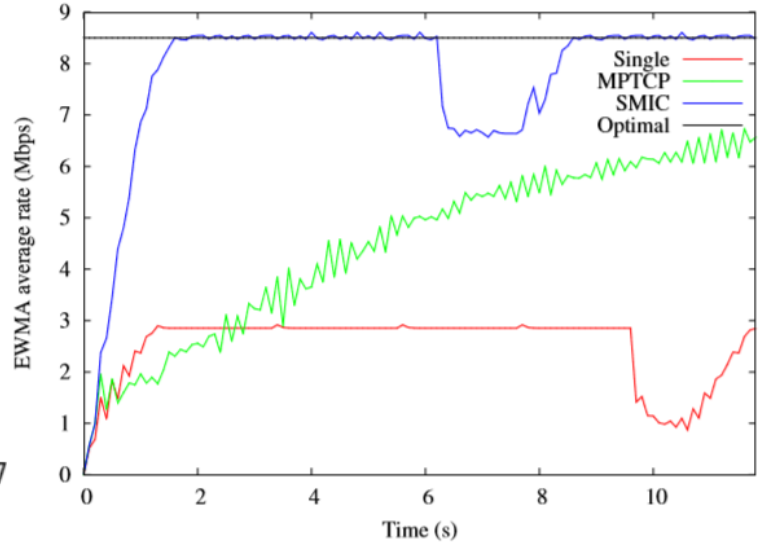
- 3 SMIC flow and 1 single flow share the bottleneck
- All of flow shows similar content retrieval time
- Cwnd tracking shows similar tendency
- Difference comes from cache hit ratio

Exploiting available network resources









- SMIC and MPTCP utilize overall network resources, but single does not
- SMIC and single show faster convergence time than MPTCP
- SMIC shows the best performance due to fast convergence + network resource utilization

Conclusion

- There are challenging issues to design multi-path Interest control for ICN
  - How to identify each path (subflow)?
  - How to forward Interests to specific path?
  - How to control Interest rate?

- We propose SMIC, subflow-level multi-path Interest control mechanism
  - Path identifier & trajectory identifier for identification/forwarding
  - Subflow-level Interest window
  - Bottleneck-sharing subflow-aware window control

- We show SMIC performs better than single or MPTCP flow on ICN

Thank you for your attention!

Appendix A.

SMIC window control

Algorithm 1 SMIC window control algorithm

1: $\mathcal{R}$ = set of all subflows
2: b_r = estimated bottleneck bandwidth history of subflow r
3: t_r = timeout history of subflow r
4: S_r = set of subflows sharing bottleneck with subflow r (including r itself)
5: $|S_r|$ = the number of elements in S_r
6: $w_r \leftarrow$ current value of subflow r's cwnd
7: **if** $Data$ arrives through subflow r **then**
8: update b_r
9: $w_r \leftarrow w_r + \frac{1}{|S_r|^2 w_r}$
10: **end if**
11: **if** $timeout\ loss$ is detected on subflow r **then**
12: $w_r \leftarrow w_r/2$
13: update t_r
14: **for all** $i \in \mathcal{R}$ **do**
15: **if** t_r, b_r is similar with t_i, b_i **then**
16: $S_r \leftarrow S_r \cup \{\text{subflow i}\}$
17: $S_i \leftarrow S_i \cup \{\text{subflow r}\}$
18: **end if**
19: **end for**
20: **end if**

Window increase

- If a subflow *r* does not share bottleneck with other subflows
  ➔ increase w_r by $1/w_r$
- If a subflow *r* shares bottleneck with *n* other subflows
  ➔ increase w_r by $1/n^2 w_r$

Window decrease

- divide w_r by 1/2

Bottleneck-sharing subflow detection

- similarity between timeout histories
- similarity between estimated bottleneck bandwidth

Appendix B.

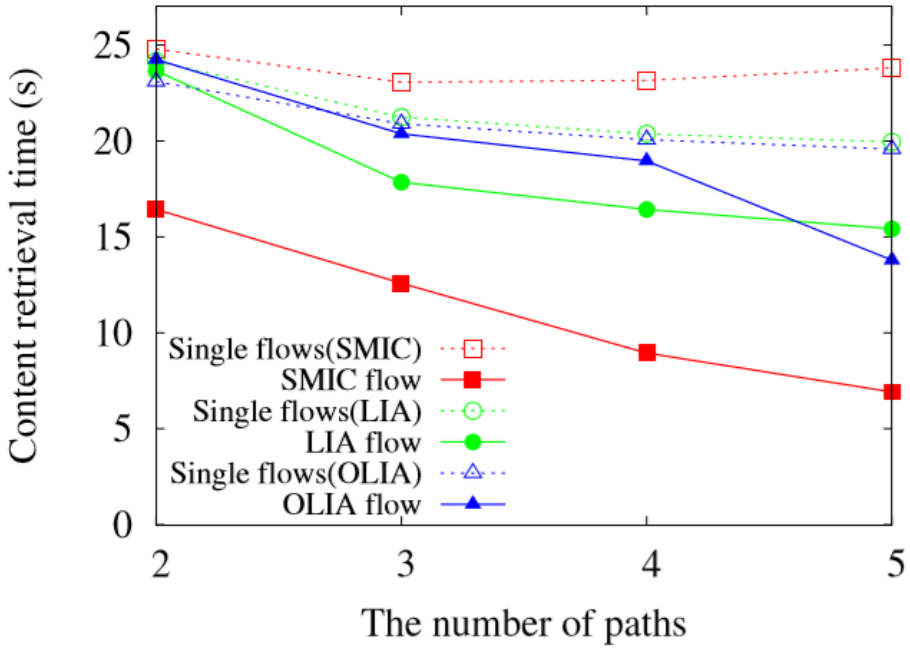
Evaluation environments

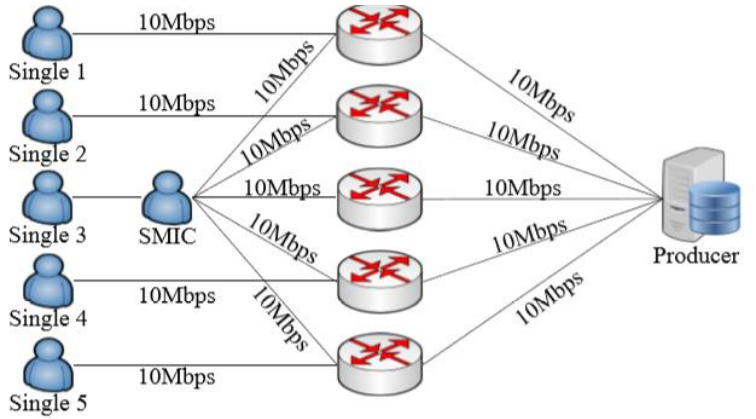
Parameter	Description
Content request	Requests follow Zipf distribution, skewness parameter α = 1.0
Number of objects per prefix	10,000
Content object size	10MB
Content store size	100MB
Topology	Bottleneck 1-2, Tree, Competition
Congestion control schemes	ICP [3], MPTCP LIA [16], OLIA [11]

- Simulator: Customized ndnSIM (based on ns-3 network simulator)
- Content store is enabled
- Requests of consumers are independent

Appendix C.

Competing with single-path flows





- 2~5 single flows with 1 SMIC flow utilizing 2~5 subflows (red lines)
- 2~5 single flows with 1 MPTCP LIA flow utilizing 2~5 subflows (green lines)
- 2~5 single flows with 1 MPTCP OLIA flow utilizing 2~5 subflows (blue lines)

- MPTCP schemes yield their bandwidth share to single flow due to slow convergence
- SMIC equally use bandwidth with single flow at each bottleneck

Appendix D.

Comparison between SMIC and alternatives

	SMIC	**MIRCC**
Congestion control algorithm	Window-based	Rate-based
Additional roles of routers	Hash-modulo operations	1. Rate calculation 2. Path steering 3. Path identification
Additional fields in Interests	Path identifier	Path steering hint
Additional fields in Data packets	Trajectory identifier	1. Link rate $R(t)$ 2. PathId
Convergence time	Longer	Shorter

- SMIC as a representative of window-based scheme
- MIRCC (ICN `16) as a representative of rate-based scheme