Application aware access and distribution of digital objects using Named Data Networking (NDN)

July 4, 2017

Rahaf Mousa

Supervisor: dr.Zhiming Zhao

University of Amsterdam

System and Network Engineering

Motivation

- In **big data infrastructures**, research data objects often have a **persistent identifier (PID)** .
- A typical PID is the **Digital Object Identifier (DOI)**. (e.g.,**DOI:10.1594/PANGAEA.842191**)
- A **data centric application** (such as a scientific workflow) often requires different data objects from multiple locations, e.g., when reproducing the results of a scientific paper.
- **Optimize the access** of multiple data objects is crucial for the system performance.
- **Information Centric Networking (ICN)** provides a suitable solution for big data infrastructure.
- One of ICN approaches is **Named Data Networking (NDN)**.

Named Data Networking (NDN): a typical ICN

- ICN replaces the **client-server** model with a new **publish-subscribe** model
- How would you get a stapler?
- From delivering the packet to a given **destination** address to fetching data identified by a given **name**.
- Ask for the "stapler", not its location(1).

(1)http://www.networkworld.com/article/3060243/internet/demystifying-the-information-centric-network.html

Research Questions

- How can we facilitate fetching of DOI identified objects via NDN network

- How can we optimize NDN network performance using application side knowledge, such as objects’ sizes?

Related Work

- A Persistent Identifier infrastructure stack for NDN, by Schmitt, Majchrzak and Bingert.
- Evaluating Caching Mechanisms In Future Internet Architectures, by Yuxin Jing. They concluded that LFU (Least Frequently Used) is the most effective cache replacement strategy.
- Interest Set Mechanism to Improve the Transport of Named Data Networking, by Xiaoke Jiang and Jun Bi. Proposed Interest Set packet for names that share the same prefix.

How NDN works? (1/2)

Naming:

- Hierarchically structured names, *e.g.*, /uva/os3/rp2/presentation/123
- Opaque to the network (only separators are recognized).

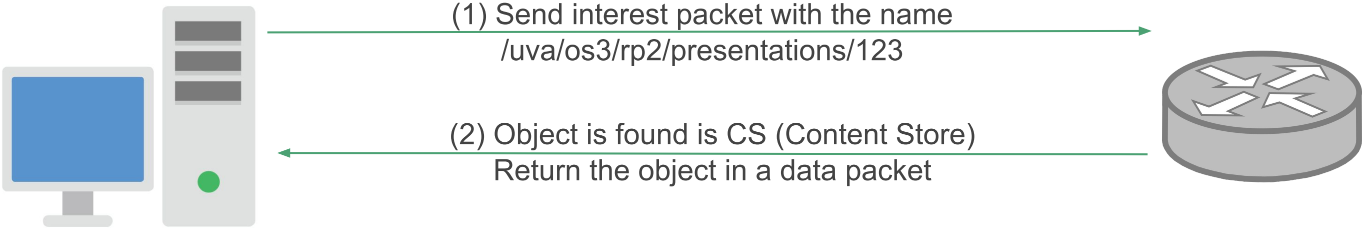
Types of packets:

- Two types of packets are exchanged; **Interest packets** and **Data packets**

NDN router data structure:

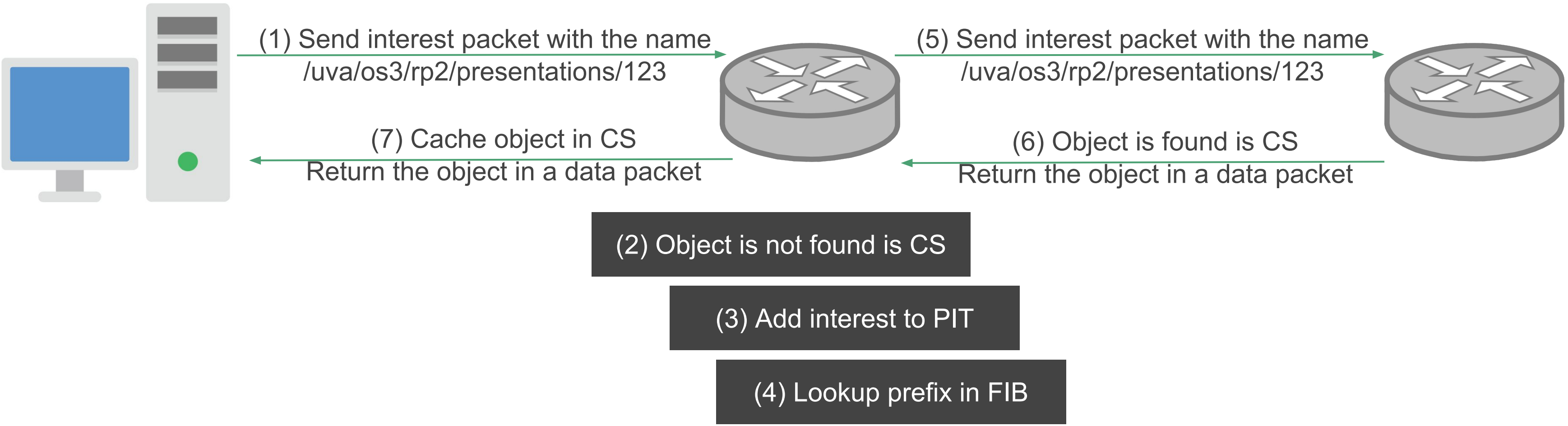
- Pending Interest Table (**PIT**), Forwarding Information Base (**FIB**), and Content Store (**CS**)

How NDN works? (2/2)



(1) Send interest packet with the name
/uva/os3/rp2/presentations/123

(2) Object is found is CS (Content Store)
Return the object in a data packet



(1) Send interest packet with the name
/uva/os3/rp2/presentations/123

(7) Cache object in CS
Return the object in a data packet

(5) Send interest packet with the name
/uva/os3/rp2/presentations/123

(6) Object is found is CS
Return the object in a data packet

(2) Object is not found is CS

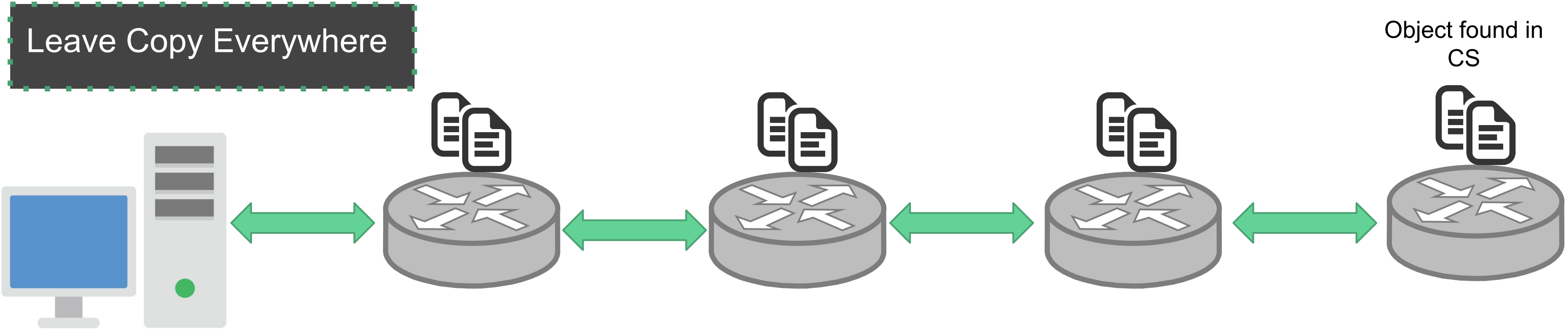
(3) Add interest to PIT

(4) Lookup prefix in FIB

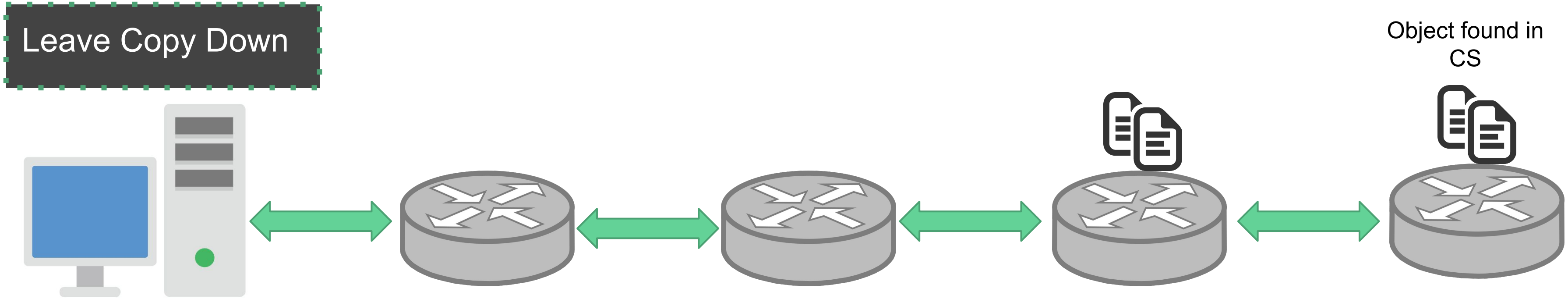
Caching in NDN (1/2)

Decision Strategy (DS)

Leave Copy Everywhere



Leave Copy Down



Caching in NDN (2/2)

Replacement Strategy (RS):

- First In First Out (FIFO)
- Least Recently Used (LRU)
- Least Frequently Used (LFU)
- Random Replacement (RR)

Fetching DOI objects to NDN network

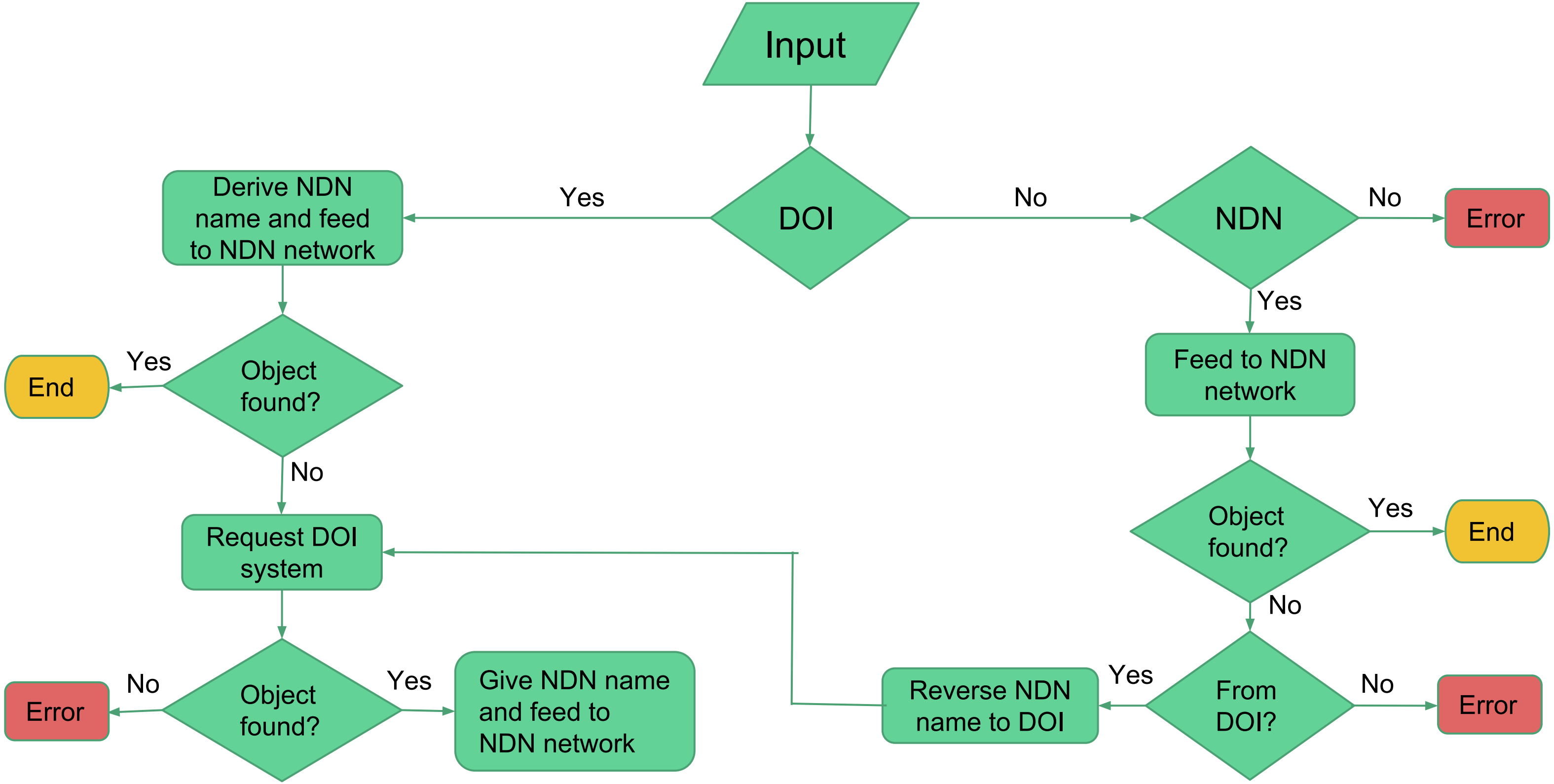
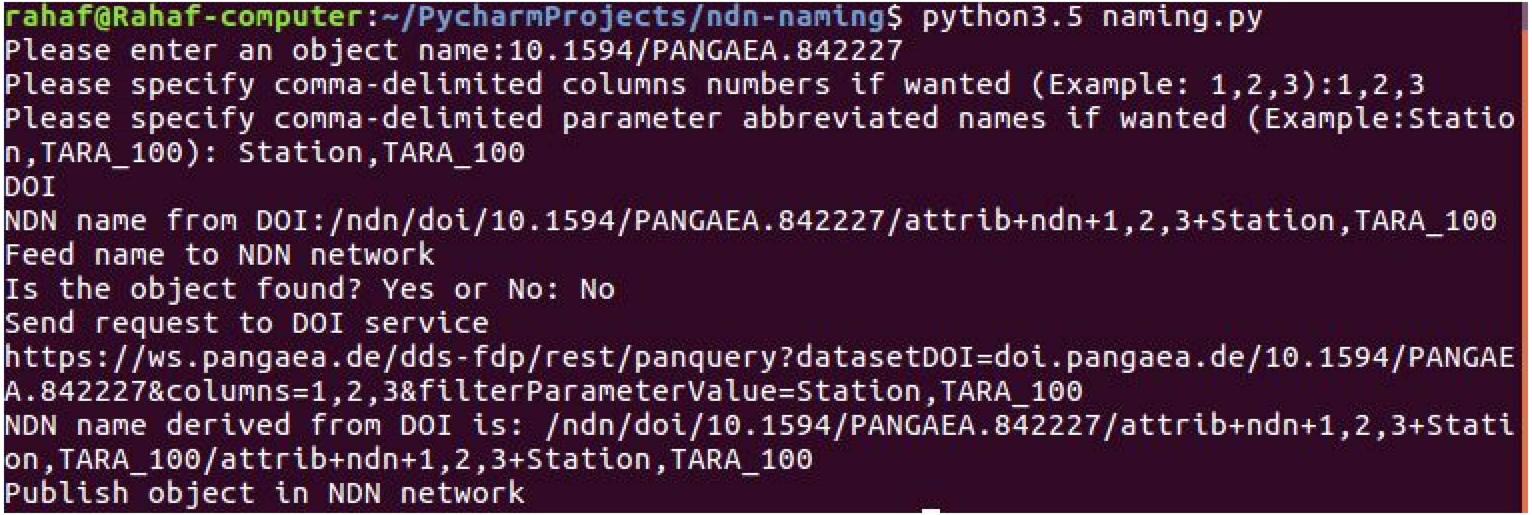


Figure 1: Flowchart of the proposed Approach

Q1: Software prototype and results

- Proof of concept with the help of PANGAEA download service (scientific data).
- It allows choosing columns and tests locations.



```
rahaf@Rahaf-computer:~/PycharmProjects/ndn-naming$ python3.5 naming.py
Please enter an object name:10.1594/PANGAEA.842227
Please specify comma-delimited columns numbers if wanted (Example: 1,2,3):1,2,3
Please specify comma-delimited parameter abbreviated names if wanted (Example:Statio
n,TARA_100): Station,TARA_100
DOI
NDN name from DOI:/ndn/doi/10.1594/PANGAEA.842227/attrib+ndn+1,2,3+Station,TARA_100
Feed name to NDN network
Is the object found? Yes or No: No
Send request to DOI service
https://ws.pangaea.de/dds-fdp/rest/panquery?datasetDOI=doi.pangaea.de/10.1594/PANGAE
A.842227&columns=1,2,3&filterParameterValue=Station,TARA_100
NDN name derived from DOI is: /ndn/doi/10.1594/PANGAEA.842227/attrib+ndn+1,2,3+Stati
on,TARA_100/attrib+ndn+1,2,3+Station,TARA_100
Publish object in NDN network
```

Figure 2: The written python script functionality

Q2: Application aware NDN optimization

The second research question:

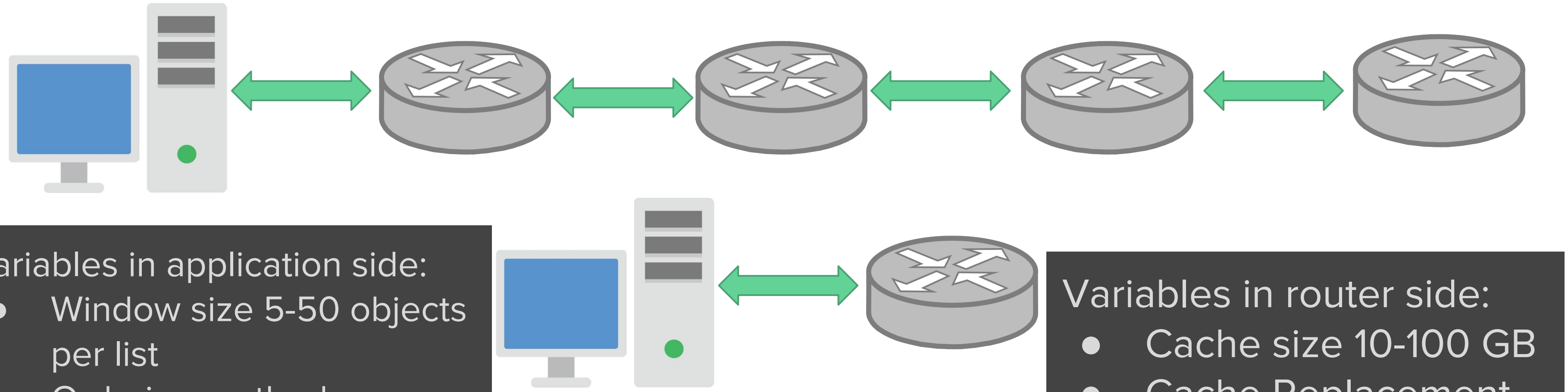
- **How can we optimize NDN network performance using application side knowledge, such as objects' sizes?**

The proposed approach:

- Assuming that the sizes of objects are known to the application (via metadata catalogues).
- The application aggregates a list of wanted objects (window size).
- The application orders the objects in ascending or descending order.

Q2: Ordering the requests using application info.

The experiment setup:



Variables in application side:

- Window size 5-50 objects per list
- Ordering method: Random, Ascending and Descending
- Set of object requested: 30 objects with sizes between 50KB-10GB

Variables in router side:

- Cache size 10-100 GB
- Cache Replacement strategies: FIFO, LRU, LFU and RR

Experiments

Simulation software:

- Consumer and producer python scripts which are a part of ndn-cxx library.
- We edited both files for ordering in consumer side and caching in producer side.

The experiments in numbers:

- In each experiment one static value of each variable is used.
- In each experiment 1000 interest packets are sent.
- Each experiment was repeated 10 times.
- The experiments output was the **cache hit ratio** (object is found in cache).

Results (1/3)

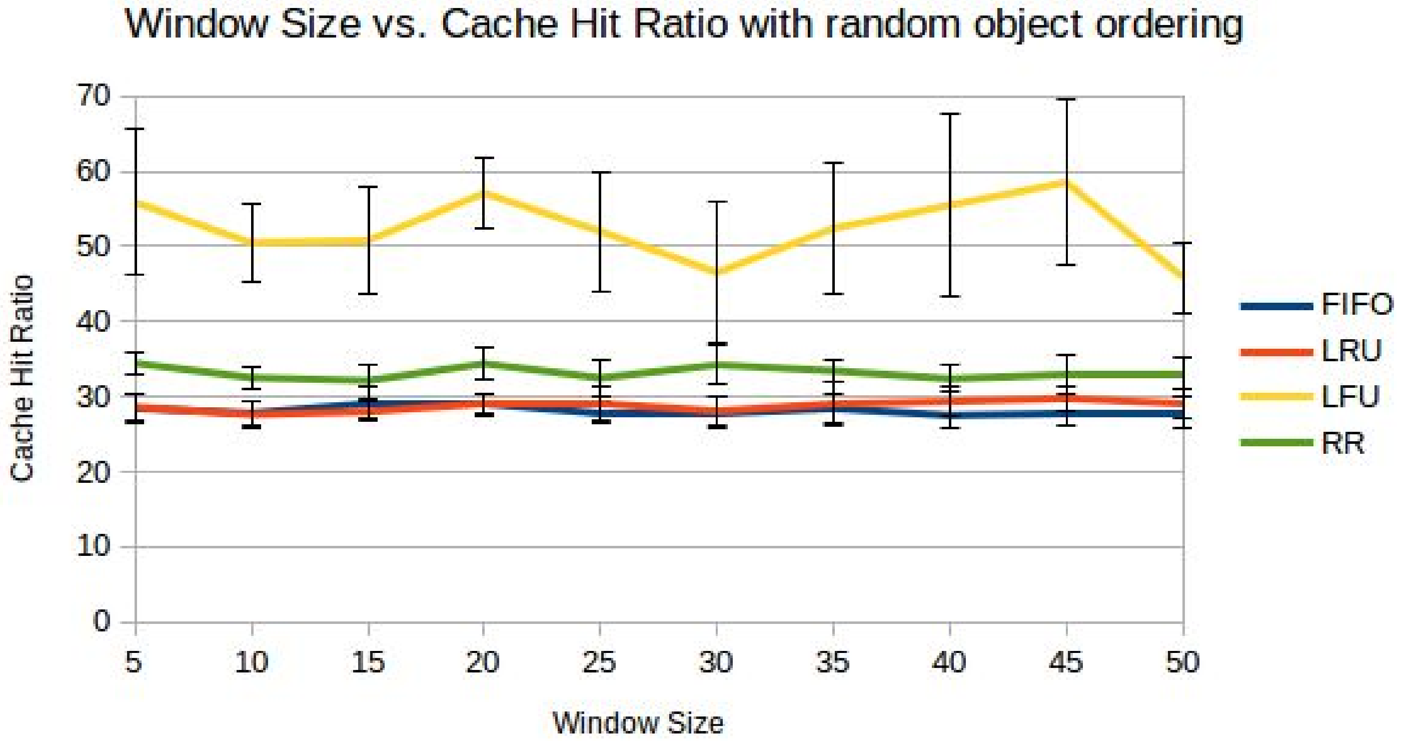


Figure 3: Window vs. cache hit ratio with random object ordering

Results (2/3)

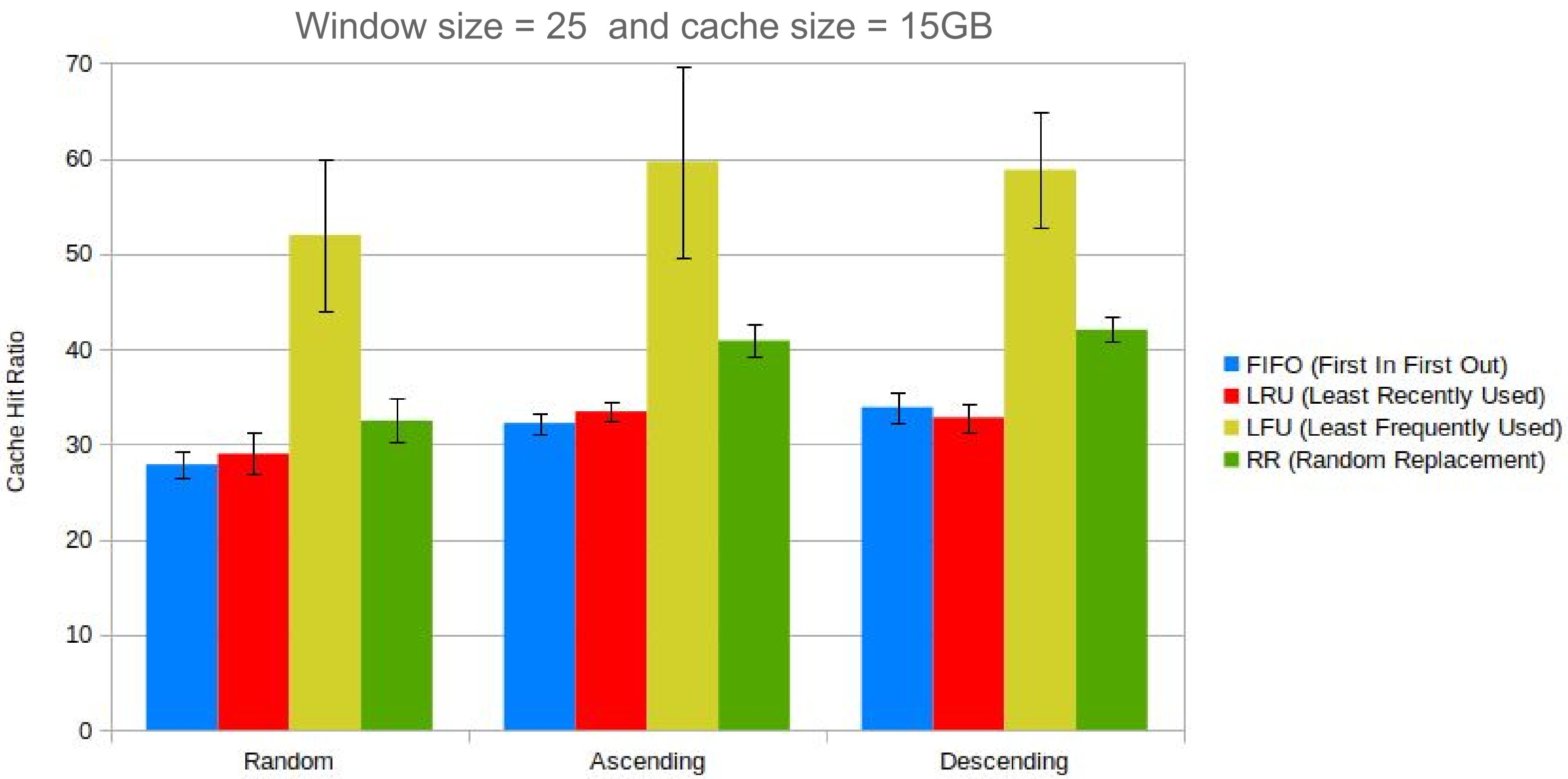


Figure 4: Bar chart of the different ordering methods with the available cache replacement strategies

Results (3/3)

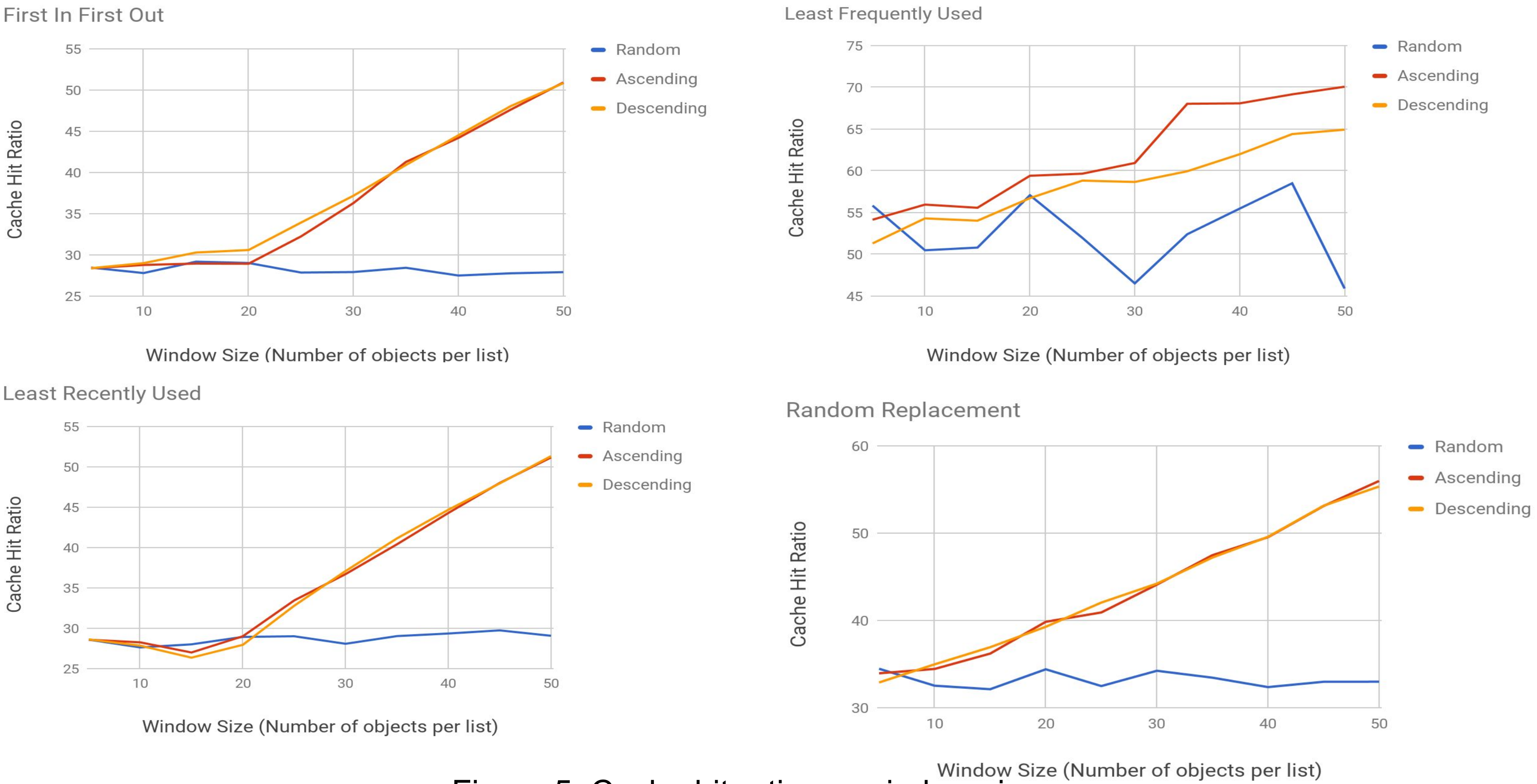


Figure 5: Cache hit ratio vs. window size

Discussion

- What can we do when application does not provide information of object size?
- Proposed solution: application ordering mechanisms in the router side.
- It is part of our future work.

Conclusion

First Question:

- It’s possible to integrate DOI objects with NDN network.

Second Question:

- Implementing ordering based on object size on the application level enhances the network performance.
- LRU cache replacement strategy gives the highest cache hit ratios with the proposed ascending ordering by size method.
- For FIFO, LRU and RR cache replacement strategies gave close cache hit ratio values for both ascending and descending ordering methods.

Future Work

- Enhancing the proposed approach for fetching objects identified with a DOI to NDN network to cover the different naming systems available.
- Implementing the aforementioned approach in a real NDN network setup.
- Implementing the ordering methods in the NDN content router and testing it.

Questions?