SIMILARITY-AWARE POPULARITY-BASED CACHING IN WIRELESS EDGE COMPUTING

Xianglin Wei, Jianwei Liu, Junwei Wang, etc.

National University of Defense Technology, Nanjing 210007, China

wei_xianglin@163.com

OUTLINE

- Introduction
- Dynamic Caching Framework
- Content Popularity Prediction
- Evaluation and Results
- Conclusion

BACKGROUND

- Edge computing aims to bring computation, storage, and communication resources to the network edge, enabling source end-data processing and response.
- Compared with centralized cloud computing paradigm, Edge computing has the following unique characteristics:
  - Low latency
  - Context aware service
  - Security enhancement where data is generated
  - Privacy leakage prevention
  - Bandwidth reduction at the core network
  - Content caching at the edge
- In collaboration with the cloud computing, edge computing could benefit the service providers, network providers, and end-users a lot.

PROBLEM STATEMENT

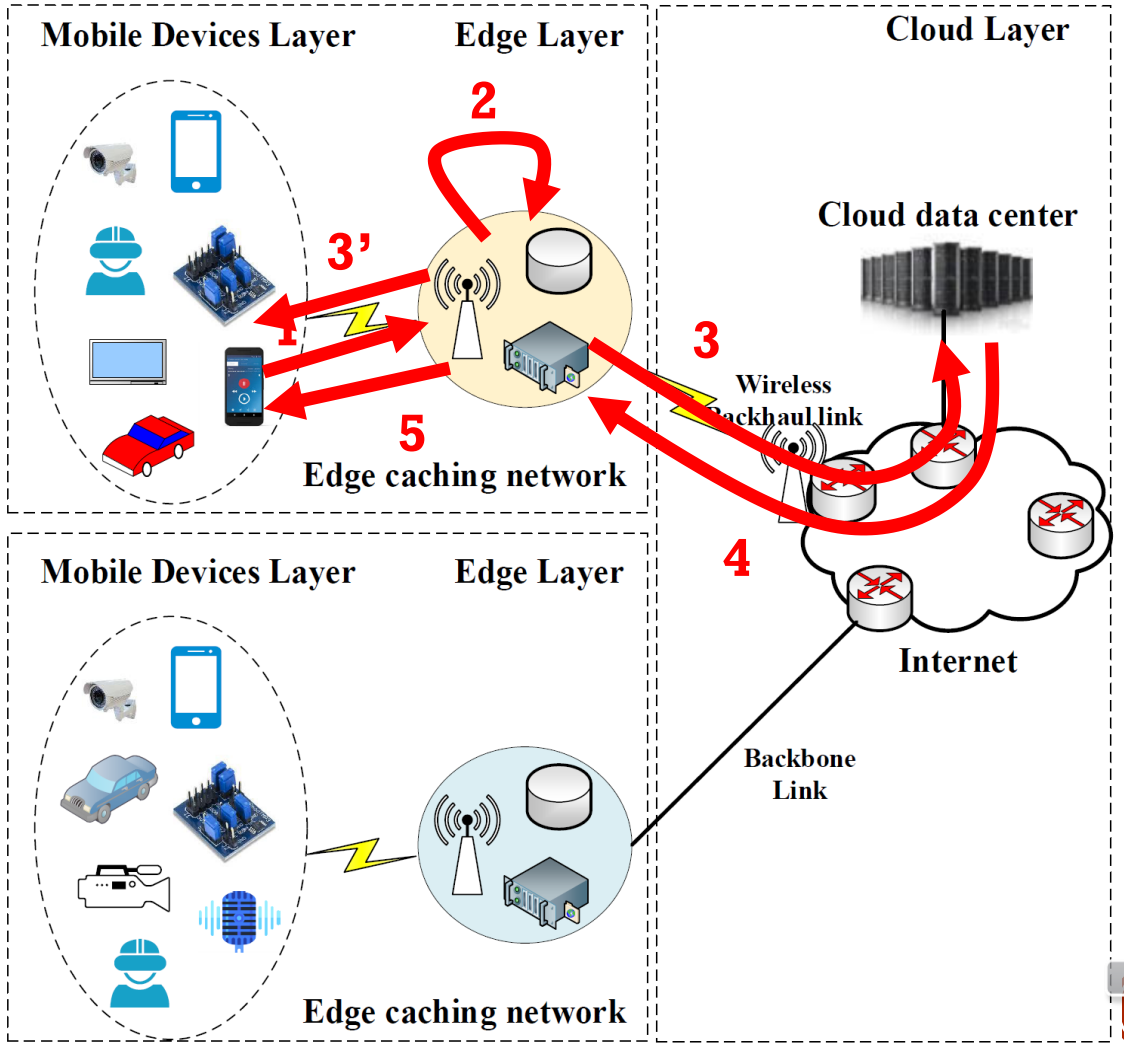
- From the perspective of content sharing, an edge server could serve as the ‘cache’ at the network edge, and can be the bridge between end-user devices and the content servers at the cloud.
- The edge caching service at wireless network edge is the research background of this paper.
- Traditionally, whether a content would be cached by the edge server is determined by the edge server based on the content’s requesting history.
- However, for the newly arrived contents in a dynamic scenarios, where both end-user devices and contents arrive and leave the system dynamically, their popularity is hard to estimated based on their 0 request history.
- This phenomenon looks like a slow start for the newly arrived contents at the edge caching server.

BASIC IDEA

- To alleviate the slow start phenomenon, we need to accelerate the caching determination process at the edge server for newly arrived contents.
- The basic idea to achieve this goal is the observation that content requests in the served area covered by the same edge server are related to the context in the served area, including events happened, common concerns in the area (weathers, maps, disasters, etc.), user tastes, etc.
- Therefore, for a newly arrived contents, its popularity will be similar to those contents that are similar to it according to the above mentioned properties. Therefore, its initial popularity value could be calculated based on its similarity with existing contents rather than its requesting history.
- Based on this idea, this paper presents a similarity-aware popularity-based caching algorithm in edge computing scenario.

DYNAMIC CACHING FRAMEWORK

- 1. Content requests initiated by the end-user device
- 2. Content lookup at the edge server, and request history update
- 3'. Reply directly when cache hits
- 3. Content requests relayed to the cloud by the edge server once cache missing
- 4. Reply from the cloud
- 5. Reply to the end-user device



CONTENT POPULARITY PREDICTION

- Factors considered when deciding an existing content's popularity in the future:

- Freshness:

$$f_{C_k}(t) = \left| t - t^{g}_{C_k} \right| \tag{3}$$

Current time

Content generation time

- Short-term popularity:

$$s_{C_k}(t) = \left| t - t^{R}_{C_k} \right| \tag{4}$$

Recent requested time

- Popularity:

$$P(C_k) = \frac{R_k}{(f_{C_k}(t) + \epsilon)(f_{C_k}(t) + \delta)} \tag{5}$$

Number of requests

Small constants

CONTENT POPULARITY PREDICTION

- Each newly arrived content is described as a vector L-dimensional vector $\mathbf{v}_{C_k} = \{v^1_{C_k}, v^2_{C_k}, \ldots, v^L_{C_k}\}$
- The similarity between two contents C_k and C_i are measured by their Euclidean distance:

$$s_{ki} = \sqrt{\omega_1 (v^1_{C_k} - v^1_{C_i})^2 + \omega_2 (v^2_{C_k} - v^2_{C_i})^2 + \cdots + \omega_L (v^L_{C_k} - v^L_{C_i})^2} \tag{6}$$

- Popularity:

$$P(C_j) = \frac{\alpha \sum_{i=1}^{K} s_{ji} P(C_i)}{K} \tag{7}$$

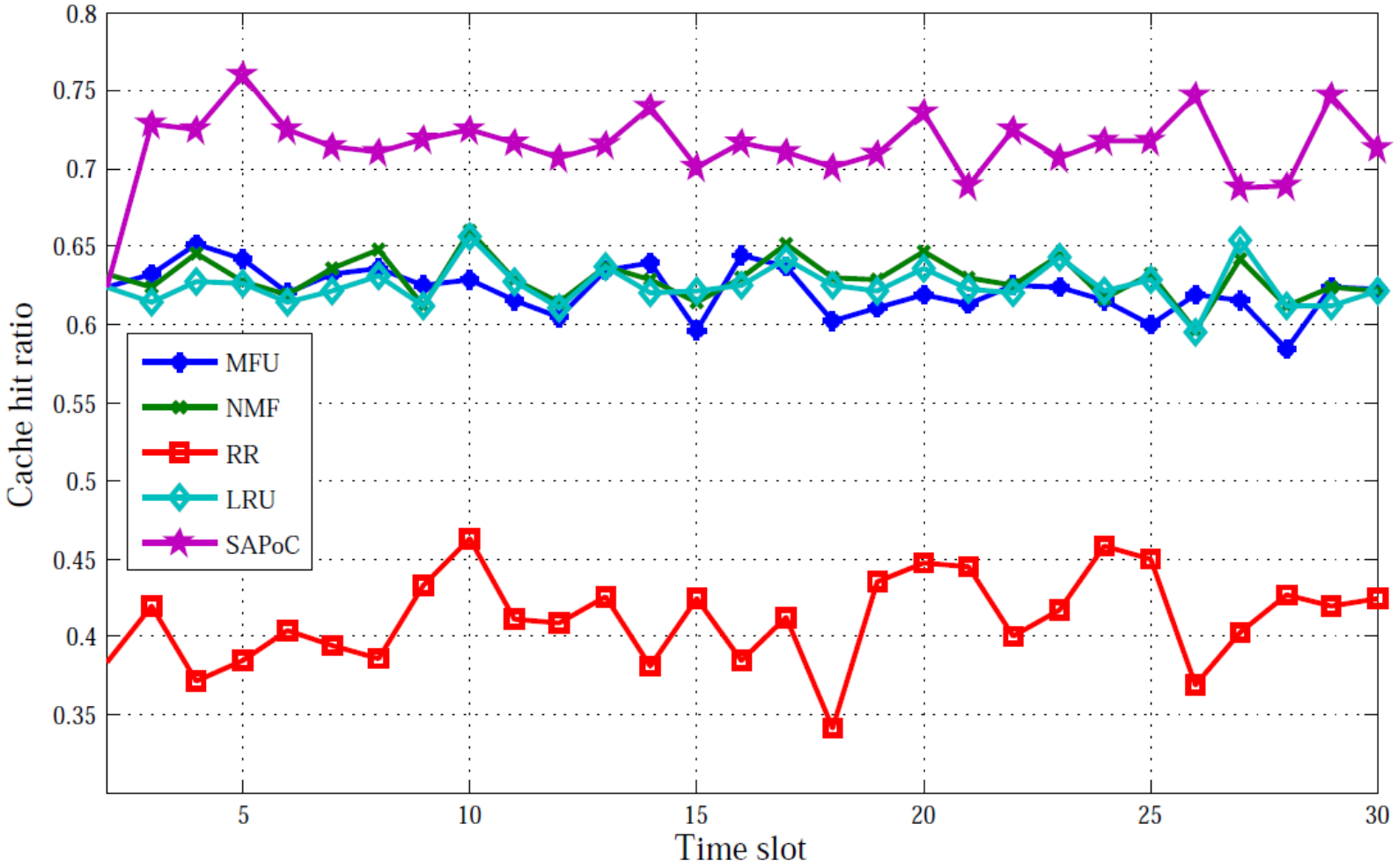
SAPOC ALGORITHM DESCRIPTION

- At first, the edge server's cache is reset to be empty, and the server calculates a content's popularity value using (5) if the content is an old content; otherwise (7) is adopted by the algorithm to derive its popularity value in each timeslot.
- Moreover, each content's caching probability is calculated using (9).Then, the contents are sorted based on their popularity values in descending order.
- If the cache space is not full, the popular contents will be stored directly; otherwise, the algorithm replaces the most unpopular $\beta \times C$ contents with the most popular $\beta \times C$ contents in the cache, where β is a coefficient between 0 and 1, and C is the size of the cache. $\beta = 0$ means no replacement is allowed while $\beta = 1$ indicates that all contents in the cache is replaceable in each timeslot.

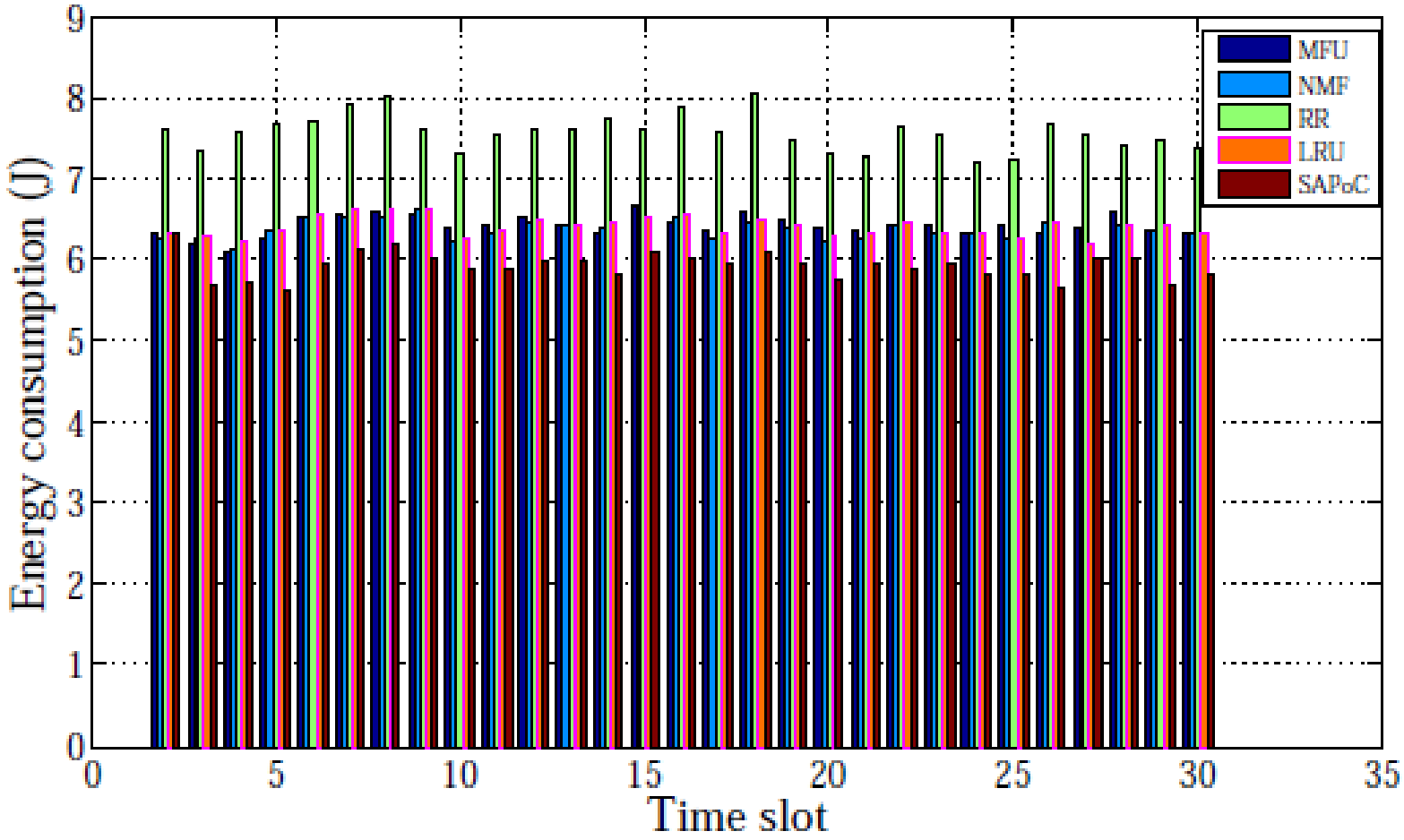
EVALUATION AND RESULTS

- Simulation settings:
  - one cloud data center, one edge server, and multiple mobile devices
  - New contents arrive the cloud data center following a Poisson distribution
  - The size of each content varies from a to b and follows a random distribution
  - The content request process is modeled by a Zipf distribution
- Comparison benchmark
  - Cache hit ratio
  - Energy consumption

RESULTS ANALYSIS – CACHE HIT RATIO



RESULTS ANALYSIS – ENERGY CONSUMPTION



CONCLUSION AND FUTUREWORK

- A similarity-aware popularity-based caching algorithm is put forward in this paper.
- Results have shown that our algorithm, named SAPoC could promote the cache hit ratio while reducing the energy consumption of the system.
- A collaborating scenario among multiple edge servers will be considered in the future.

THE END

Thanks for Listening!

Any Questions?