Mobile Caching In Continuous Environment With Dynamic Data
Shivangi | ss5947@rit.edu
Advisor : Minseok Kwon

OBJECTIVE
To design a mechanism for cooperative caching in a distributed wireless system where robots are continuously moving and sensing data in the random direction within their allocated range. This application keeps most requested data; observed locally and shared from other robots; in the vicinity of a robot for faster access than getting data through the network when required, eventually increasing cache hit ratio.

INTRODUCTION

- Caching: Refers to the idea of keeping a remote copy of data locally in order to reduce access time.
- This project is designed to improve the efficiency of cache memory in a distributed system.
- Cache space is limited and data unlimited, so we need to make good use of available space.
- Robots only walk in allocated distance range.
- The hard disk stores the data that is replaced from the cache.
- Most popular data is shared with neighbors which helps in increasing cache hit ratio.

ALGORITHM

- INSERTION
  Reads data from the environment, and add it in cache making sure that most recent element stays on top of the cache. Updates if data already exist. This procedure uses LRU replacement policy when the cache is full.

- LOOKUP
  Looks for data corresponding to given location in the own cache, HD, and neighbors, in that order. It follows broadcast approach for lookup and this process is done dynamically. The popularity of data is determined from the user request.

ARCHITECTURE

- SHARING
  In this procedure robot receives a request from neighbors and shares most popular data from it’s cache with the requesting neighbor. Received temperature is compared with Threshold for fire alerts. This method uses MFU for sharing.

- FAILURE HANDLING AND RECOVERY
  This handles failure if the robot is out of coverage area or is not responding. Try to reconnect with the robot by pinging until TTL expires. If robot reconnects, HD maintains it states, cache starts makes a new start.

RESULTS

- Data sharing resulted in higher cache hit ratio compared to traditional approach. And sharing is done based on the frequency of data helps to keep popular data close to the user.
- Sharing using the same cache for both local and shared data performed better for random search over total grid area whereas separate caches were better when search spectrum was limited to robot allocated area and it’s close by robot.

CONCLUSION

References