Chunk-based In-network Caching and Diffusion in Content-Centric Networks

Kideok Cho, Munyoung Lee, Kunwoo Park, Ted “Taekyoung” Kwon, Yanghee Choi, *Seoul National University*

MMLab Multimedia and Mobile communications Laboratory

Motivation

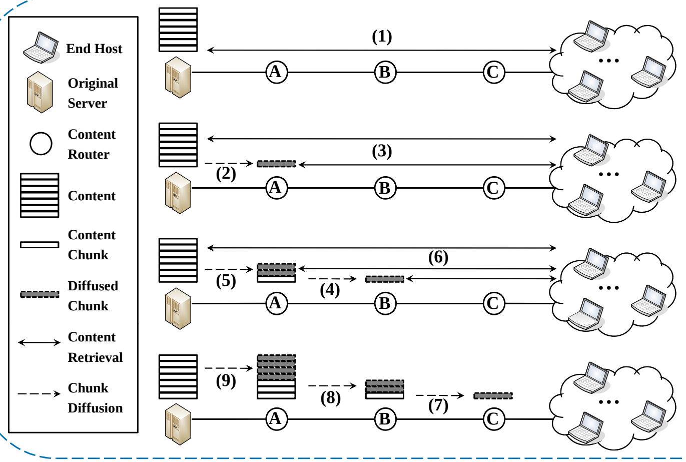
- Depending on caching strategies, the overall performance of CCNs can vary significantly
  - In terms of cache hit ratio and storage utilization
- Previous works cannot be directly applied to the CCNs
  - Topological limitations, explicit coordination between caches, prior knowledge on request pattern, …
  - Without cooperative caching, there might be duplicated caching, resulting in storage waste
- Also, inter-chunk relation raises interesting issues in chunk-based CCNs
  - E.g., sequential delivery, …
- We propose a “Simple”, “Decentralized”, and “Popularity-based” caching and diffusion algorithm in CCNs, called WAVE

Proposed Idea (WAVE) Overview

- Distribute/diffuse content chunks to the network entities (such as routers)
  - Diffuse chunks as the content request changes
  - To make “the cache hits of contents” happen earlier (closer to end users)

- As a consequence,
  - Network utilization will be improved: the number of duplicate content delivery (thus total traffic volume) can be reduced
  - Caching efficiency will be enhanced: chunks of popular contents will be cached more
  - The overhead of cache management will be reduced

WAVE Operation Illustration



Content request will be guided towards the original server

With another request for the same content, a content chunk will be diffused

The number of content chunks to be diffused will be increased exponentially

The cache hits of contents will happen earlier in the network as the # of request increases

WAVE Algorithm

Algorithm 1 Chunk Diffusion Algorithm

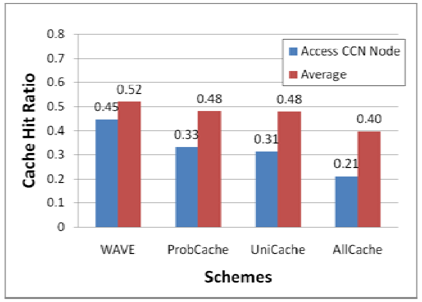
```
x: diffusion base (e.g., 2,3,...)
n: chunk window state (initial value: 0)
t: total number of cached content chunks
sent: chunk id sent until now (initial value: 0)
i: id of requested chunk

Transfer the requested chunk $i$
if $i == t$ then
    diffuse chunks from $sent + 1$ to $min(x^{n+1}, t)$
    $n \leftarrow n + 1$
    $sent \leftarrow min(\sum_{i=0}^{n-1} x^i, t)$
else if $i \leq sent$ then
    $n \leftarrow \lfloor \log_x i \rfloor$
    $sent \leftarrow i - 1$
end if
```

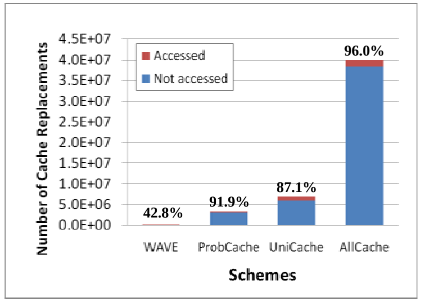
As the access count of a content file increases, WAVE exponentially increases the number of chunks of the content file to be cached and diffused

Exponential Diffusion

Variables Update

Simulation Results





- WAVE achieves the highest cache hit than the other schemes (both access CCN node and on average)
  - By caching the popular chunks more (exponentially increasing caching)
- AllCache shows the lowest cache hit ratio due to its popularity-blind and aggressive caching

- Only 42.8% of chunks are replaced without being accessed in WAVE due to its popularity-based chunk diffusion algorithm
  - Resulting in efficient cache management
- More than 87% of content chunks are not accessed before being replaced in the other schemes

Conclusion & Future Work

- WAVE achieves higher cache hit ratio & less cache replacement counts
- WAVE implementation using CCNx & Large-scale experiments over the testbeds (e.g., cloud, PlanetLab, etc)