TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik, Institut für Nachrichtentechnik, Lehrstuhl Kommunikationsnetze

Implementation of Decentralized Coded Caching in Erroneous Networks using KODO

Motivation

Network traffic has a strong temporal variability. This leads to congestion during peak hours and underutilizes resource during off-peak hours. In order to solve them, we use caching technology, which "moves" some of the traffic from peak hours to off-peak hours.

Decentralized Coded Caching

Comparing with Centralized Coded Caching[1], which utilizes a central coordination in placement phase, Decentralized Coded Caching is more realistic. Because the number of the clients and their identity are usually unknown. Next, we use an example to explain the principle of decentralized coded caching.

Firstly we consider a server connected through an shared link to two clients, the size of each client's cache memory is M bits. This server has two files, respectively A and B. The size of each file is F bits. Each user caches subset of bits of each file independently and randomly, satisfying the memory constraint. Now the file A and file B are divided into 4 subsets respectively. The all subsets of A and B are included in two set $s_A$and s_B. on the other word:

$$s_A = \{0,1,2,12\},\ s_B = \{0,1,2,12\}$$

And the subsets of A {1,12} are cached in client 1 and the subsets of B {2,12} are cached in client 2. Then the placement phase are accomplished . As shown in the figure 1 below:

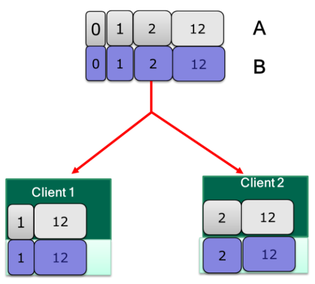


Figure1: Placement phase

Multicast Data

Next phase is the delivery phase, we assume that client 1 requests file A and client 2 requests files B. The server responds the clients with sending two types of data: XORed data and original data(no XORed). So now we will explain how to create the XORed data. There are two steps.

The first step:

• Search the all clients which ask for the same files. If every two clients of them has different parts of the requested file, then exchange these parts of file between themselves.

For example:

Client n requests file R and has the α part of file R: $V^n_{R,\alpha}$

Client m requests file R has the different part β of file R: $V^m_{R,\beta}$

Hence, we can gain the XORed data:

$$V^n_{R,\alpha} \otimes V^m_{R,\beta}$$

The second step:

• Search the all clients which ask for the different files. If every two client have files which are needed each other, then exchange these files between themselves.

For example:

Client n requests the file R and has α part of the file G: $V^n_{G,\alpha}$

Client m requests the file G and has β part of the file R: $V^m_{R,\beta}$

Hence, we can gain the XORed data:

$$V^n_{G,\alpha} \otimes V^m_{R,\beta}$$

Following the search algorithm, server finds two XORed subsets : $V^1_{B,\alpha} = \{1\}$ and $V^2_{A,\beta} = \{2\}$. After this server also find that subsets {0} of file A and file B are not cached in the clients memory, subsets{0} are clarified into no XORed data, i.e. subsets{0} are transmitted in the form of original data by the server in the delivery phase, this is shown in the figure 2 below:

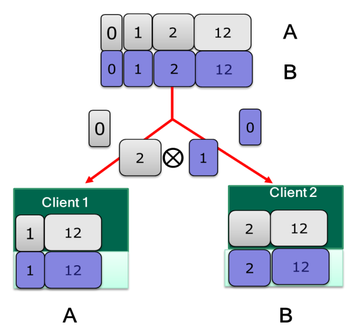


Figure 2: Delivery phase

When the clients receive the XORed data, by encoding them to fetch the original data. Specific decoding with XOR operation is shown in figure 3:

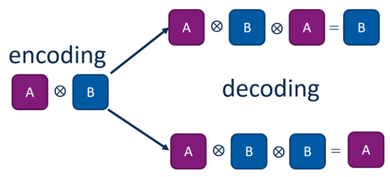


Figure 3: the method of decoding

KODO and RLNC

Why use KODO[2]?

As we all know, multicast base on UDP protocol, its weakness is the unreliability. If Internet is not so stable, the video files, which server send and are received by clients, can not be played normally in the clients because of the lost packets and the reordering. In order to solve these problems we get help from KODO Library. KODO implements a number of different erasure correcting codes. In our project, the Full RLNC
(Random Linear Network Coding) is used.

There are two very important parameters in KODO: *Symbols* and *SymbolSize*.
"*Symbols*" denotes the number of symbols in a block. With increasing this number will have the following effects:

- Computational complexity will increase.
- Decoding delay will be larger

"*Symbolsize*" denotes the size of each symbol in bytes. Maximum size of symbol is around 1300 bytes.

The size of each block equals to *Symbols*SymbolSize*. In our project, *SymbolSize* is constant(512 Bytes). The realization with KODO is shown below as figure 4:

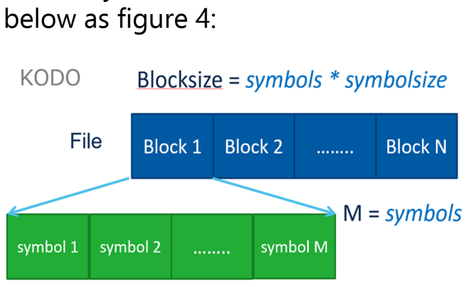


Figure 4 : Structure of KODO

Figure 5 shows the graphical representation

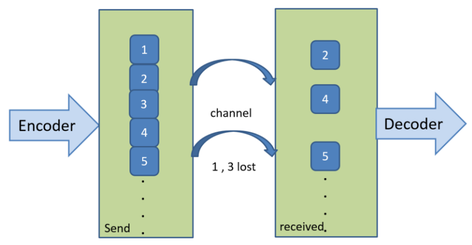
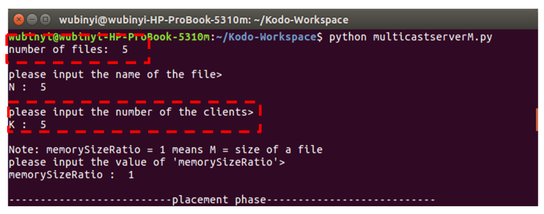
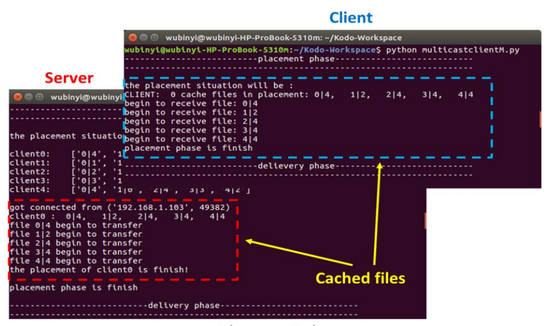


Figure 5: graphical representation

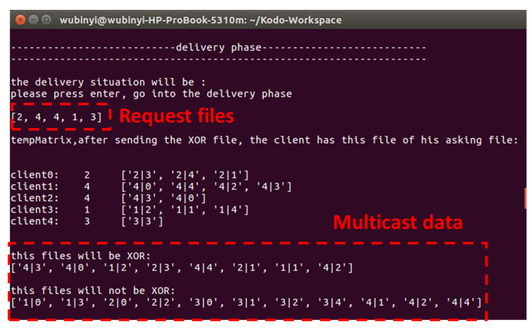
Results of runing program



Initial



Placement phase



Multicast data

Delivery phase

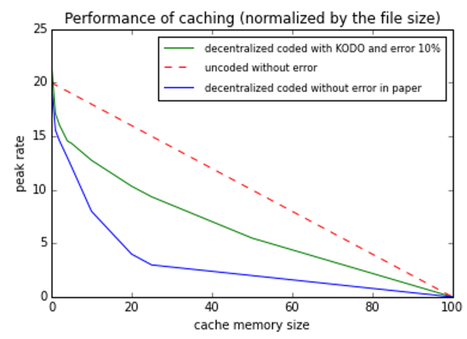


Figure 7: Simulation for 100 files and 20 clients

Conclusion

- Using Decentralized coded caching reduces the peak rate
- Decentralized caching is proposed in the paper[1] but without errors. We consider a more practical situation with lost packages and solve this problem with KODO.
- We have designed an algorithm to search XOR files.

Reference

- [1] Mohammad Ali Maddah-Ali and Urs Niesen, "Decentralized Coded Caching Attains Order-Optimal Memory-Rate Tradeoff", 28 Mar. 2014
- [2] Kodo document in steinwurf, http://docs.steinwurf.com/overview.html

Participants: Binyi Wu
Beibei Wu
Kaixuan Guo

Betreuer: Dr. Anil Chorppath

ZWEITLOGO

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur