Applying a rateless code in content delivery networks

Suherman1,*, Muhammad Zarlis1, Sahat Parulian Sitorus1, Marwan Al-Akaidi2
1Universitas Sumatera Utara, Kampus USU Medan 20155, Indonesia
2Arab Open University, Kuwait.
*suherman@usu.ac.id

Abstract. Content delivery network (CDN) allows internet providers to locate their services, to map their coverage into networks without necessarily to own them. CDN is part of the current internet infrastructures, supporting multi server applications especially social media. Various works have been proposed to improve CDN performances. Since accesses on social media servers tend to be short but frequent, providing redundant to the transmitted packets to ensure lost packets not degrade the information integrity may improve service performances. This paper examines the implementation of rateless code in the CDN infrastructure. The NS-2 evaluations show that rateless code is able to reduce packet loss up to 50%.

1. Introduction
Content delivery network (CDN) is a collection of servers, replicating server content to other servers, to shorten the server and client proximity and to enhance service performance. CDN allows a web server (or other internet services) installed in one server, and copied to other server if necessary to make clients’ accesses easier [1]. In order to do so, server selection algorithm is required. Client requests may be addressed to origin server (surrogate) and redirected to the nearest server if necessary. This requires IP redirection technique. Other solution is by using a DNS server, which forwards requests to the best server. Server selection may use polling algorithm such as round robin (RR), weighted round robin (WRR), least connection (LC) [2], geographical mapping using a domain name system (GeoDNS) [3], a web mapper [4] or other algorithms. Server update is required to maintain the integrity of data on multiple servers.

This paper examines the impact of applying rateless code [5] on CDN servers when sending short packets to client. The purpose of the implementation is to increase the service reliability for traffics that are transmitted using unreliable protocol such as user datagram protocol (UDP). Why such traffics require more reliable service? Firstly, there has been growing applications that require short packet transmissions such as whatsapp, facebook and other social media. Secondly, if those social media applications employ UDP, then it has no guarantee on data delivery [6]. Applying rateless code may improve the probability that packets are received successfully.

Rateless code adds redundancy on packet transmissions. The original m packets in sender are processed to generate m+n transmitted packets. Receiver will be able to reconstruct message if it receives at least m packets from any of m+n packets [6]. By doing this, lost on certain packet sequence could have no impact to the message integrity and the performance may increase.
Figure 1a show how the rateless code works in CDN. Originally, a social media user, M, uploads data to a social media server X (1). User N accesses data belongs to M on a social media server that is addressed to CDN DNS (2). CDN DNS selects the best (perhaps the nearest) social media server, say server Y. Server Y checks the updated version of user M by requesting update from server X (3 & 4). Server Y then sends data M to user N (5). Since the transmitted data from server X to server Y is predicted to be short, containing only update information or status, rateless code may be implemented from server X to server Y. Since some social media provides its user interface in user equipment (a gadget for instant) which store previous information, user N may only need an update data of user M. Rateless code may be implemented from server Y to user N.

Server Y generates n redundant packets from m original packets (Figure 1b). User N needs to receive minimum m packets, not necessarily in order to reconstruct the transmitted information. User interface on user N decodes m packets into original message and embeds it to application data.

![Diagram](image)

**Figure 1.** Content delivery network

### 2. Research method

In order to evaluate the rateless code performances in delivering social media application on CDN, a multi-server network is set on NS-2 simulator [7] as depicted in Figure 2. There are three servers serving 19 clients. These three servers exchange updated data among each other. 10 clients request data from servers while other 9 clients generate background traffics.
The CDN server selection is implemented by using DNS on node 4 using round robin (RR), weighted round robin (WRR), least connection (LC) and geographically clustered service area using GeoDNS. The server, DNS, and GeoDNS coverage area are plotted in Figure 3.

The rateless code encoder is implemented in server and decoder is in client. The rateless code is set to generate 50% and 100% packet redundancy. The transmitted packet is 1054 bit length with rate less than 300 kbps. The evaluated performances are in term of packet delay, packet jitters and packet loss.
3. Simulation results

3.1. Single server versus CDN

The first assessment is to compare the performances of servers when each server works as a single server; each server is assigned to a certain area (GeoDNS); server selection using LC, RR and WRR algorithm. Figure 4 shows delay performance of each method. Single server experiences the worst up to 279 ms, while GeoDNS has the lowest delay down to 75 ms.

![Figure 4. Delay single server vs CDN](image)

In term of packet loss and jitter (Figure 5), least connection based server selection suffers the least. Based on these figures, assessment on rateless code impact on CDN is based on least connection server selection.

![Figure 5. Jitter and packet loss single server vs CDN](image)

3.2. Impact of Rateless Code on CDN performance

Increasing number of the transmitted packets results delay increment on packet transmission (Figure 6). Rateless code with 50% redundancy experiences 20% delay increment to about 90 ms, while 100% raises delay to 58% (119 ms). Likewise, jitter worsens by value of 20.7% and 41%. However, these values are still in acceptable range [8].
However, number of packet loss decreases more than 50% for redundancy 50% and about 25% for redundancy 100% (Figure 7). This happens as rateless code ignores losses if the received packet reaches $m$ packets, the same as the original packets.

On the other hand, increasing packet redundancy from 50% to 100% reduces the rateless code performance shown by the increase of packet loss. It occurs as redundant packets congested the network.

4. Conclusion
The conducted CDN assessments by using NS-2 simulations show that CDN performs much better than single server. The assessment also reveals that GeoDNS server selection generates the lowest delay as clients are served by the nearest server. However, server selection based on the least connection within the serving machine results the lowest packet loss.

Rateless code implementation on the selected CDN technique shows positive impact as number of packet loss decreases significantly. The 50% rateless code redundancy decreases
lost packets up to 50%, while 100% redundancy drops the packet loss up to 25%. Even though rateless code increases delay and jitter, the values are still under the accepted level. In addition, 50% redundant performs better than 100% redundant. Therefore, redundancy rate should be carefully calculated.

References

[1]. Haßlinger, Gerhard, 2014, "Efficiency of caching and content delivery in broadband access networks." Advanced Content Delivery, Streaming, and Cloud Services: 71-89.

[2]. Soni, Gulshan, and Mala Kalra, 2014 "A novel approach for load balancing in cloud data center." In Advance Computing Conference (IACC), 2014 IEEE International, pp. 807-812. IEEE.

[3]. Khiyaita, A., H. El Bakkali, M. Zbakh, and Dafir El Kettani, 2012, "Load balancing cloud computing: State of art." In Network Security and Systems (JNS2), 2012 National Days of, pp. 106-109. IEEE.

[4]. M. Andrews, B. Shepherd, A. Srinivasan, P. Winkler and F. Zane, June 2002, “Clustering and server selection using passive monitoring.” In Proc. IEEE INFOCOM, pp. 1717-1725.

[5]. Luby, Michael, Mark Watson, Tiago Gasiba, Thomas Stockhammer, and Wen Xu. 2006, "Raptor codes for reliable download delivery in wireless broadcast systems." In Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE, vol. 1, pp. 192-197. IEEE.

[6]. Suherman, Suherman, and Marwan Al-Akaidi, 2014, "Increasing uplink broadband video streaming protocol performance in WiMAX network." International Journal of Internet Protocol Technology 7, no. 3, pp. 176-185.

[7]. Issariyakul, Teerawat, and Ekram Hossain, 2011, Introduction to network simulator NS2. Springer Science & Business Media.

[8]. ETSI Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON), October 1998, General aspects of Quality of Service (QoS) TR 101 329 V1.2.5.