International Journal of Future Generation Communication and Networking
Vol. 8, No. 2 (2015), pp. 9-20

An Efficient Video Content Delivery Strategy for Radio Access Network Environments

Po-Jen Chuang and Hang-Li Chen

Department of Electrical Engineering
Tamkang University
Tamsui, New Taipei City, Taiwan 25137, R. O. C.
E-mail:pjchuang@ee.tku.edu.tw

Abstract

When a radio access network (RAN) environment cannot handle the tremendous mobile data traffic, users may experience degraded or declined services. To help process the large number of user video requests in RANs, a content delivery network (CDN) architecture has been recently introduced, which establishes a number of cache servers outside the source server for users to get the needed data from the nearest cache server. To upgrade the performance of a wireless CDN, this paper presents a new caching strategy based on analytical results of real user video request traces and specific caching considerations for videos with different popularity degrees. The new strategy first caches videos with high popularity to all helpers and then caches those with low popularity to the remaining helper storage to enhance the overall request hits. Performance evaluation exhibits that, when compared with other caching strategies, our new strategy is able to yield better request hits with low complexity and average delay time.

Keywords: *Content delivery networks (CDNs), radio access networks (RANs), caching strategies, video request hits, complexity, delay, experimental evaluation*

1. Introduction

According to the visual networking index [1] released by Cisco, the average monthly global mobile data traffic was 1.5 EB in 2013 and will grow up to 15.9 EB in 2018 with about 70% of it coming from video traffic data. By that time, the radio access network (RAN) environments (as shown in Figure 1) will have very serious problems absorbing the estimated huge growth of mobile data traffic. In fact, the trouble may emerge earlier in 2017 as Strategy Analytics has forecasted. When it happens, users may experience degraded or even declined services because the base station in RANs will have difficulties handling the large number of user video requests [2]. If viewing the base station as a server, we can thus describe the foreseeable problem: A single server may fail to manage too many user requests and will then respond to these requests with extra delay time. To solve the incoming problem, some researchers have recently introduced the content delivery network (CDN) architecture [3-5] to assist the practice of RANs.

In the CDN architecture, a number of cache servers are established outside the source server. The source server then caches its contents to the cache servers so that users, when filing requests, can get the needed data from the nearest cache server. Such a practice can increase the delivery speed and reduce the burden on the source server. The performance of CDN architectures is subject to a proper caching strategy by which the source server can decide how to set up desirable cache servers and to cache what contents to them. A number of caching strategies have been proposed for the CDN architecture, including [6-15]. For instance, to facilitate the performance of a wireless CDN architecture, the **Greedy** strategy [6-7] creates several cache nodes (the so-called helpers) in a single cell. It faces a major problem: When the video volume grows much bigger than the cache

ISSN: 2233-7857 IJFGCN

storage of helpers, the strategy can only cache the videos with high popularity. For the considerable number of videos with low popularity, it will turn over very limited user request hits.

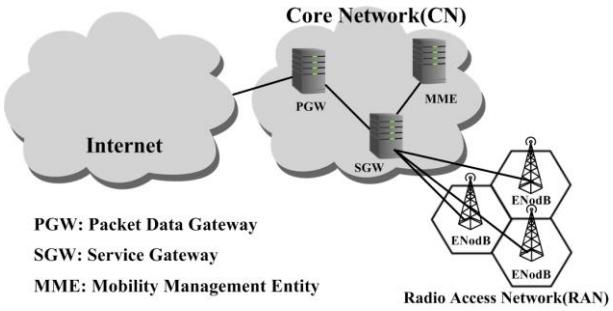

Figure 1. An Illustration of the Radio Access Network (RAN)

The main goal of this investigation is to enhance the practice of RANs by building a new caching strategy over the wireless CDN architecture. Different from the Greedy strategy, our new caching strategy analyzes the real user video request traces and, based on the results, adopts different approaches to handle videos with different degrees of popularity. Our basic idea is to cache videos with high popularity to all helpers first and then cache those with low popularity to the remaining storage (if the total storage of helpers is not full). By doing so, we can effectively lift up the request hits of those less popular but numerous videos.

Extensive simulation runs are conducted to evaluate the performance of our cache strategy and other CDN cache strategies, including the **Greedy** [6, 7], **Popular** [7], and **Fuzzy Decision** [8] strategies. The obtained results show that, when the cache storage of helpers goes below 50% of the total videos (i.e., with limited helper cache storages), the **Greedy** strategy attains only few hits for the big number of less popular videos. The **Popular** strategy is shown to yield much lower request hits because it caches only those highly popular videos, and the **Fuzzy Decision** strategy misses even the requests of highly popular videos because it does not consider video popularity at all. By contrast, our new strategy performs following the real user request traces and therefore is able to achieve higher request hits with lower complexity. We also use the widely applied LTE-sim [16-19] to simulate these target strategies and obtain the average video download delay time per user, with the advantage to our strategy as well.

2. The Proposed Strategy

When caching a video, the **Greedy** strategy must ensure that each user can find the video in a nearby helper. But, it is difficult to predict accurately which user will make a request in which position and, meanwhile, wrong user location predictions will bring up huge impact. If users are evenly distributed, the strategy needs to cache every video to most of the helpers. On the other hand, if the video volume largely exceeds the total helper storage, helpers can cache only limited videos. In such a situation, the Greedy strategy which has attempted to handle most user requests by nearby helpers will have to leave a significant amount of requests untended. To solve the problem, we build a new caching strategy based on the analytical observations of real user request traces to increase the probability of having helpers handle user requests, i.e., to increase the user request hits.

To facilitate our investigation, we examine the traces of the YouTube requests extracted from the wired campus network, between the second half of 2007 and the first

half of 2008, at the University of Massachusetts at Amherst [20-21]. Table 1 gives one example of these request traces.

Table 1. The Trace of a YouTube Request

Timestamp	YouTube server IP	Client IP	Request	Video ID	Content server IP
1189828805.208862	63.22.65.73	140.8.48.66	GETVIDEO	lML9dik8QNw	158.102.125.12

We take the number of requests for each video as its popularity and observe that, each day, the average user requests for videos with higher popularity (i.e., with more requests) stands about 30% of the total requests. We also observe that, when removing these 30% more popular videos, the remaining 70% videos exhibit little popularity differences. To facilitate our later discussions, we now define those more popular videos as *popular videos* and the rest as *non-popular videos*.

Assume that there are *N* videos and the popularity (*i.e.*, the number of requests) of video *i* is *Pi*. We first accumulate the popularity of the *N* videos in set *S*, in descending order, *i.e.*, *Pi*-1>= *Pi*.

$$S = \{P_1, P_2, \ldots, P_N\}$$

Asmentioned previously, the requests forpopular videos stand about 30% of the total video requests. We can thus definethe total requestsfor popular videos (denoted as *Ppopular*) as

$$P_{total} = \sum_{i=1}^{N} P_i$$

$$P_{popular} = P_{total} \times 30\%$$

With the above definitions, we can start our first phaseof operation: caching all popular videos to all of the helpers. Figure 2 gives the details of the caching flow.

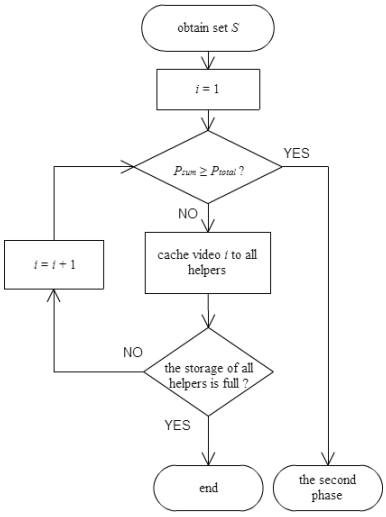

Figure 2. The Flow Chart of the First Phase: Caching Popular Videos to All Helpers

In Figure 2, P_{sum} (initially 0) indicates the total popularity of the cached videos. When P_{sum} remains below $P_{popular}$, cache the next popular video to all helpers; repeat the same operation until the caching storage of all helpers is full orP_{sum}equals/exceeds$P_{popular}$.

If the cache storage of helpers is not used up after the first phase, we will start the second phase of operation. Assuming that we have cached *H* videos out of the total

N videos, $H<N$, to all helpers in the first phase, we now have the following $S_{non-popular}$ (the N-H non-popular videos) set

$$S_{non-popular} = \{P_{H+1}, P_{H+2}, \dots, P_{N-1}, P_N\}$$

As the analytical results of the YouTube requests exhibit little popularity differences for non-popular videos, we thus evenly distribute these non-popular videos to the helpers without considering the popularity. If we assume all videos are with the same size and each helper has the cache storage for K videos, we will attain the total cache storage of M helpers as

$$Storage_{total} = M \times K \ videos$$

Excluding the H videos which have been cached in the first phase, we now have the following remaining cache storage

$$Storage_{surplus} = M \times (K - H) \ videos$$

That is, we can distribute the remaining non-popular videos to the helpers according to the obtained $Storage_{surplus}$. Figure 3 depicts the flowchart of our second phase of operation.

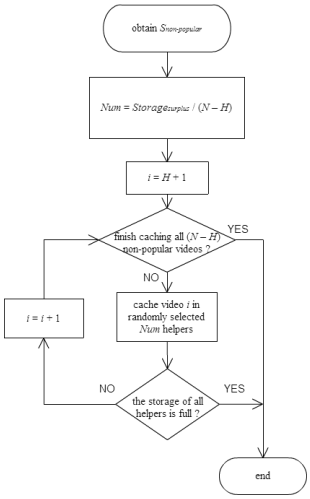

Figure 3. The Flow Chart of the Second Phase: Caching Non-Popular Videos

In Figure 3, *Num* indicates the quotient of the remaining cache storage divided by the total number of non-popular videos; each non-popular video hence can be cached to *Num* helpers. For a non-popular video i, $i = H+1, H+2, \dots, N-1, N$.Whena video (starting from $H+1$) is cached to the *Num* helpers randomly, the remaining cache storage will be decreased until the$N-H$ videos are all cached or $Storage_{surplus}$is used up.

To sum up, the proposed caching strategy adopts different approaches to handle popular and non-popular videos. In the first phase, it caches the popular videos -- one by one, in the order of popularity -- to all helpers until using up the helper storage or the popular videos. This is a different approach in contrast to the **Greedy** strategy which starts by caching each video to most of the helpers and consequently occupies too much helper storage in the beginning. By caching popular videos, instead of all videos, in the beginning, we can practically save storage and ensure helpers to come up with proper request hits. Such a design can also avoid the loophole of the **Fuzzy Decision** strategy. (Note that helpers in the **Fuzzy Decision** strategy may fail to handle quite a few *popular* requests as it will randomly cache

videos several times and select the best result without considering popularity degrees.)

After the practice of the first phase, if the helpers still have cache storage, our strategy will start the second phase of operation. As the above trace analysis of the YouTube requests shows little popularity difference for non-popular videos, our strategy evenly distributes these non-popular videos without considering the popularity to the helpers in the second phase. Our key consideration in this phase is to cache possibly more non-popular videos to the helpers. By doing so, we can further enhance the request hits, given that most of the video requests are for non-popular videos (e.g., 70% in the above You Tube case). Unlike the **Greedy** strategy which caches very few non-popular videos or the **Popular** strategy which caches only popular videos, our strategy can cache significantly more non-popular videos to the helper (so as to increase the request hits), thanks to its second-phase practice. **Our strategy** is relatively simple because it deals with the popularity degrees of videos only, not involving user location predictions as the **Greedy** strategy. For better understanding we list critical features of different caching strategies in Table2.

Table 2.Critical Features of Different Caching Strategies

	Greedy	**Popular**	**Fuzzy Decision**	**Ours**
Environment	wireless	wireless	wired	wireless
Popularity information	required	required	not required	required
Users‘ positions	required	not required	required	not required

3. Experimental Evaluation

3.1. The Simulation Environment

To evaluate the performance of our new caching strategy (**ours**) and other related strategies, including the **Greedy**, **Popular** and **Fuzzy Decision** strategies, we carry out extensive simulation runs in a created single cell of the RAN environment with a number of helpers as Figure 4 depicts. We pick up user request traces by randomly selecting from the You tube requests (specified in the previous section) which had been filed during the four *hottest* hours (i.e., hours with the most incoming requests)on January 29, 2008:a total of 5252 requests filed for 3888 different videos. Each requested video is assumedly with resolution = 640 × 360, video length = three minutes and video size = 30MB. To attain fair simulation, we suppose the popularity of each video is known in advance and meanwhile follow the environment settings of the Greedy strategy in[7] --with cell radius = 400 meters and a random set of helpers in the cell = 32.The performance of the four target strategies is collected and compared in terms of *video request hits* and *complexity*. (Note that our simulation involves LTE- sim [16] to obtain the average delay each user takes to download a video.)

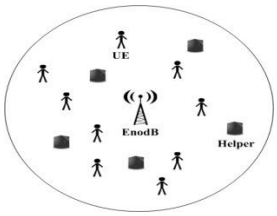

Figure 4.The Simulation Environment

We build the helper's model like the **Greedy** strategy. The connections between the helpers and users are Wi-Fi-like D2D (device-to-device) communications [7], and the transmission speed between the helpers and users is assumed by using 802.11n [22]. Table3 gives related parameters for the helper's model.

Table 3. The Helper's Model

Parameter	value
helper range	100m
data rate(Mbps)	R = 3 + (4/100) * (100 - d)

Table 4 gives related parameters of the base station and channel models. We use the LTE-sim with the following parameters to simulate the environment of 3GPPSpecificationsRelease 8.

Table4. The Base Station and Channel Models

Parameter	value
bandwidth	15MHz
subcarrier spacing	15kHz
number of subcarriers per OFDM symbol	12
slot duration	0.5ms
scheduling time (1 TTI)	1ms
schedulers	proportional fair(PF)
Path Loss Model For Urban Environment	$L = 128.1 + 37.6*\log_{10} d$

3.2. Simulation Results

3.2.1. Video Request Hits vs. the Helper Storage

To see how helper storage relates to the video request hits, we carry out simulation runs with different volumes of helper storage ranging from 30GBs to 90GBs. Figure 5depicts the numbers of video request hits versus the helper storage (in GBs) for different strategies. As we can see, when the helper storage is small, the **Popular** strategy yields fewer video request hits because it caches only the popular videos. For the **Fuzzy Decision** strategy which considers no popularity degrees of the videos, increasing the helper storage does not distinctly increase the request hits. It yields better request hits than **Popular** only at smaller cache storage (likely because of caching more different videos). **Our strategy** depicts the best hits between cache storage 30~60GB because we heed not only the hits for popular videos but also that for the non-popular videos. The **Greedy** strategy generates more

desirable video hits than **our strategy** between cache storage 60~90GB, mainly because it can follow user location predictions to make sure videos be cached in the neighboring helpers when there is sufficient helper storage.

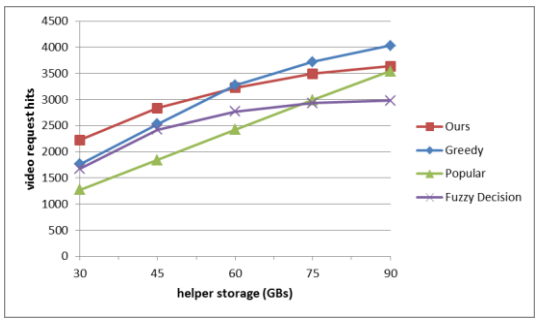

Figure 5. Video request hits vs. the helper storage (GBs).

Figure 5. Video Request Hits vs. the Helper Storage (GBs)

Note that the 5252 video requests covered in our simulation are the accumulated requests in four consecutive hours, each respectively having 824, 1457, 1732 and 1239 requests. Figure 6 exhibits the accumulated request hits of the four hours with helper storage = 30GB (where n in the x-axis indicates the first n hours). The results show that, for the **Popular** strategy, users make requests for popular videos mainly in the first two hours (which have more focused accumulated request hits). In the last two hours, we can see more sporadic non-popular videos have been requested. **Our new strategy**, by contrast, enables the request hits to grow continuously even during the last two hours. This happens because we endeavor to raise the request hits for both the popular and non-popular videos. The **Greedy** strategy displays quite leveling performance during the final hour, a result of yielding lower hits for non-popular video requests.

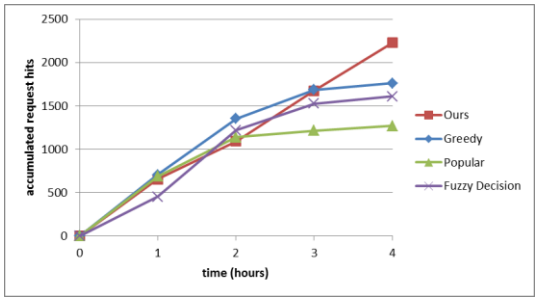

Figure 6. Accumulated request hits vs. time (hrs) (30GB helper)

Figure 7 depicts the accumulated request hits of the same four hours, with 60GB helper storage which is able to cache about 50% of the total videos. With 60GB helper storage, the **Greedy** strategy can upgrade its request hits to catch up with **our strategy**, because it has bigger helper storage to cache more videos and is with the assistance of user positions. As for the **Popular** strategy, we can see it attains better request hits for the first two hours because users make more requests on popular videos during that period of time. The result also reveals that the request hits for the **Fuzzy Decision** strategy display the same developing trend as that for **our strategy**, but the **Fuzzy Decision** strategy constantly yields lower request hits than **our strategy** because it does not take the popularity of videos into account.

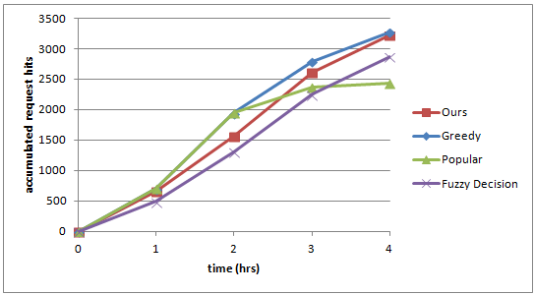

Figure 7. Accumulated request hits vs. time (hrs) (60GB helper).

Figure 8exhibits the accumulated request hits of the above four hours with 90GB helper storage which can cache about 80% of the total videos. When the helper storage grows to 90GB, each helper will be able to cache most of the videos. In such a situation, the **Greedy** strategy turns over higher request hits than **our strategy** with the aid of user positions. That is, when the helper storage grows bigger, caching more kinds of videos (popular or non-popular) turns less significant. It helps explain why, in Figure 8, our strategy trails behind the **Greedy** strategy in request hits. However, as mobile data traffic will grow tremendously in the future (mentioned in the beginning of Section 1), it will be impossible for a helper to cache over 50% of the total videos by that time because, to do so, a helper will need extremely massive caching storage which may group to hundreds of GBs. With hundreds of GBs storage, we can foresee escalating transmission delay due to a lot of factors, including cache update. In view of this prospect, caching strategies should be designed for practice in reasonable environments, i.e., when helpers are with more reasonable caching capability.

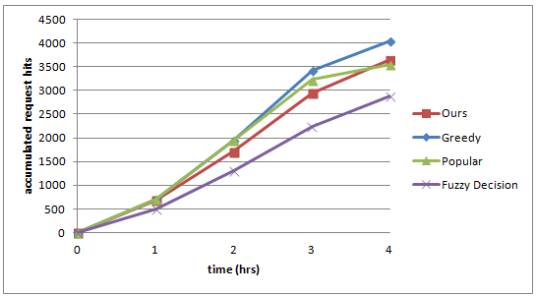

Figure 8. Accumulated Request Hits vs. Time (hrs) (90GB Helper)

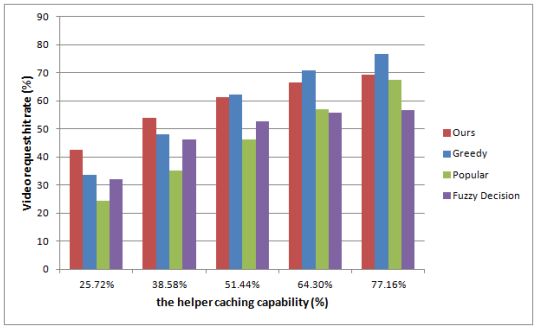

Figure 9. Video Request Hit Rates vs. the Helper Caching Capability

3.2.2. Video Request Hits vs. the Helper Caching Capability

Figure 9 gives that result of video request hits versus the helper caching capability. It further illustrates the advantage of our strategy, that is, when helpers are with reasonable caching capability, our special design out of the popularity degrees of videos can ensure more desirable performance in request hits. Note that the helper caching capability in Figure 9 indicates the percentage of total videos each helper can cache and the request hit rate indicates the number of request hits/the number of total requests. As we can see, our strategy yields higher request hit rates when the helper caching capability is under about 50% -- which is the more reasonable case.

3.2.3. Computational Complexity

Figure 10 gives computational complexity denoted by the number of lookups in the helper storage, such as the lookup for a requested video or the lookup to see if the helper storage is full upon caching a video. The result shows that the complexity of the **Popular** strategy equals the total number of videos to be cached by the helpers, and that **our strategy** (which randomly caches the non-popular videos) produces slightly higher complexity than the **Popular** strategy. It also shows that the **Fuzzy Decision** strategy produces higher complexity than **Popular** and **our strategy** because it needs 10 runs of random caching before reaching caching decisions. Among all strategies, the **Greedy** strategy takes the most complexity because it needs to search all user locations in order to decide on the appropriate helper for caching each of the videos.

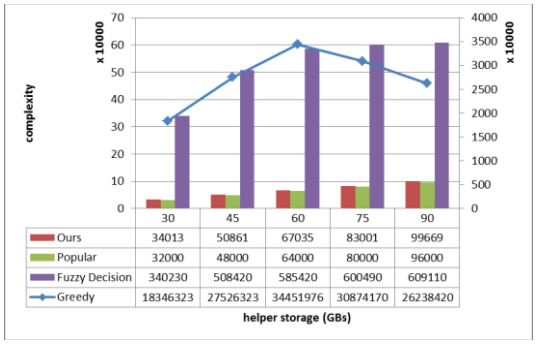

Figure 10. Complexity vs. Helper Storage

3.2.4. Average Delay Time

In the simulation, we also use LTE-sim and the helper model to simulate the average delay time for a user to download a video by the strategies. The average delay time will be attained by the following steps:

(1) Use LTE-sim to simulate delay time for a user to download one video from the base station.
(2) Use the helper model to calculate the delay time for a user to download one video from the helper.
(3) Average the sum of the above two delay times to get the estimated average delay time.

The result of the average delay versus the helper storage is plotted in Figure 11. Note that the **Baseline** strategy in this simulation assumes all requests are served by the base station, *i.e.*, it uses no wireless CDN architectures. As the result illustrates, each strategy--except **Baseline** can shorten the delay time in a trend corresponding to the results of

request hits in Figure 5. In fact, the undesirable result of the **Baseline** strategy helps prove the feasibility of using wireless CDN architectures to deal with the problems which will confront the RAN sin the near future.

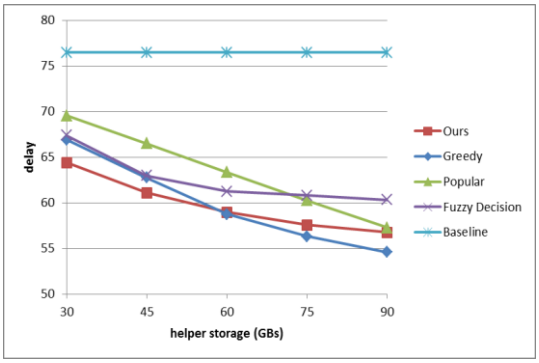

Figure 11.The Average Delay vs. the Helper Storage

4. Conclusions

Considering that the radio access network (RAN) environments will face very serious challenges in managing colossal growth of mobile data traffic in the near future, a content delivery network (CDN) architecture has been proposed to assist the practice of RANs. The CDN architecture sets up a number of cache servers outside the source server which then caches its contents to them. When a user files requests, it can get the needed data from the nearest cache server to increase the delivery speed and reduce the burden on the source server. A proper caching strategy is therefore critical to the performance of CDN architectures as it helps decide how to set up desirable cache servers or to cache what contents to them. To facilitate the performance of a wireless CDN architecture, this paper introduces a new caching strategy based on the analytical results of real user video request traces. The new strategy adopts different caching approaches to handle videos with different degrees of popularity. It first caches videos with high popularity to all helpers and then those with low popularity to the remaining storage. The special design helps increase the request hits of those less popular but large-numbered videos and upgrade the overall performance of a CDN. Experimental evaluation shows that, in contrast to related strategies, our new caching strategy is able to achieve higher request hits with significantly lower complexity and average delay time.

References

[1] Cisco Visual Networking Index, Forecast and Methodology, Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html, (2013-2018).
[2] S. Rudd, “Global Backhaul Investment Gap of $9.2 Billion could increase Customer Churn.” Available: http://www.strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=5332.Lazar and W. Terrill, “Exploring content delivering networking,” in IT Professional, vol. 3, no. 4, **(2001)** August, pp. 47-49.
[3] G. Pallis and A. Vakali, “Insight and Perspectives for Content Delivery Networks,” in Communications of the ACM - Personal information management, vol. 49, no. 1, **(2006)** January, pp. 101-106.
[4] M. Pathan and R. Buyy, “A taxonomy of CDNs, Content Delivery Networks,” in Germany: Springer-Verlag, **(2008)**.
[5] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch and G. Caire, “Fem to caching: Wireless video content delivery through distributed caching helpers”, Proc. 2012 IEEE INFOCOM, **(2012)** March, pp. 1107-1115.

[6] N. Golrezaei, A. G. Dimakis, A. F. Molisch and G. Caire, "Fem to caching and device-to-device collaboration: A new architecture for wireless video distribution", IEEE Communications Magazine, vol. 51, no. 4, **(2013)**, pp. 142-149.
[7] J. B. Chen, "Efficient content placement on multimedia CDN using fuzzy decision algorithm", Applied Mathematics & Information Sciences, vol. 6, no. 2, **(2012)** April, pp. 471-477.
[8] L. Qiu, V. Padmanabhan and G. Voelker, "On the placement of web server replicas," in INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE, **(2001)** April, pp. 1587-1596.
[9] S. Sivasubramanian, M. Szymaniak, G. Pierre and M. van Steen, "Replication for web hosting systems," in ACM Computing Surveys, vol. 36, no. 3, **(2004)** September, pp. 291-334.
[10] B. R. Tan, and L. Massouli´e, "Optimal content placement for peer-to-peer video-on-demand systems," in IEEE/ACM Transactions on Networking, vol. 2, no. 2, **(2013)** April, pp. 566-579.
[11] S. Borst, V. Gupta and A. Walid, "Distributed caching algorithms for content distribution networks," in Information communications, 29th conference, **(2010)** March, pp. 1478-1486.
[12] H. Huang, P. Xia, S. G. Chan, G. Shi and H. Zhang, "Joint optimization of content replication and server selection for video-on-demand", IEEE International Conference on Communications, **(2012)** June, pp. 2065-2069.
[13] W. Jiang, S. Ioannidis, L. Massoulíe and F. Picconi, "Orchestrating massively distributed CDNs," in ACM Co NEXT, **(2012)**.
[14] J. P. Muñoz-Gea, J. Malgosa-Sanahuja and P. Manzanares-Lopez, "Optimizing content placement in a peer-assisted VoD architecture", Peer-to-Peer Networking and Applications, vol. 6, no. 3, **(2013)** September, pp. 340-360.
[15] G. Piro, L. Grieco, G. Boggia, F. Capozzi and P. Camarda, "Simulating LTE cellular systems: An open-source framework", IEEE Transactions on Vehicular Technology, vol. 60, no. 2, **(2011)** Febuary., pp. 498-513.
[16] C. Bouras, N. Kanakis, V. Kokkinos and A. Papazois, "Application layer forward error correction for multicast streaming over LTE networks", International Journal of Communication Systems, **(2012)** February.
[17] C. Xu, F. Zhao, J. Guan, H. Zhang and G. M. Muntean, "QoE-driven user-centric VoD services in urban multi-homed P2P-based vehicular networks", IEEE Transactions on Vehicular Technology, vol. 62, no. 5, **(2012)** November, pp. 2273-2289.
[18] M. Iturralde, T. A. Yahiya, A. Wei and A. Beylot, "Resource allocation using shapely value in LTE networks", IEEE International Symposium on Personal Indoor and Mobile Radio Communications Conference, **(2011)** September, pp. 31-35.
[19] YouTube Traces from the Campus Network, 2008. Available: http://traces.cs.umass.edu/index.php/Network/Network.
[20] M. Zink, K. Suh, Y. Gu and J. Kurose, "Watch Global Cache Local: YouTube Network Traces at a Campus Network - Measurements and Implications", IEEE MMCN, **(2008)** January.
[21] E. Perahia and R. Stacey, "Next generation wireless LANs: throughput, robustness and reliability in 802.11n", Cambridge University Press, **(2008)**.

Authors

Po-Jen Chuang, received the B.S. degree from National Chiao Tung University, Taiwan, R.O.C., in 1978, the M.S. degree in Computer Science from the University of Missouri at Columbia, U.S.A., in 1988, and the Ph.D. degree in Computer Science from the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, U.S.A. (now the University of Louisiana at Lafayette), in 1992. Since 1992, he has been with the Department of Electrical Engineering, Tamkang University, Taiwan, where he is currently a Professor. He was the department chairman from 1996 to 2000. His main areas of interest include parallel and distributed processing, fault-tolerant computing, computer architecture, mobile computing, network security and cloud computing.

Hang-Li Chen, received his B.S. and M.S. degrees in Electrical Engineering in 2011 and 2014 from Tamkang University, Taiwan. His research interests include distributed processing, content delivery networks and mobile computing.