

ARTICLE TYPE

Peer-Assisted Content Delivery Network by Vehicular Clouds: Algorithm and Evaluation[†]

Lewis Tseng^{*1} | Juan Mantica¹ | James DeAntonis¹ | Moayad Aloqaily² | Takamasa Higuchi³ | Onur Altintas³

¹Computer Science, Boston College, MA, USA

²Carleton University and Gnowit Inc., ON, Canada

³Toyota InfoTechnology Center, U.S.A., CA, USA

Correspondence

*Lewis Tseng, St. Mary's Hall, 2nd floor S.
Boston College, Chestnut Hill, MA 02467.
Email: lewis.tseng@bc.edu

Abstract

In this paper, we study how to improve the performance of content delivery networks (CDNs) using a novel concept of *vehicular clouds*. Several studies have shown that the traditional CDN is *insufficient* to handle the rapidly increasing data content and traffic. Our main idea is to use a cluster of connected vehicles (i.e. vehicular clouds) to act as virtual servers in CDNs in order to relieve the workload of physical CDN servers. The key benefits of our system include guaranteed content availability/reliability in terms of *mean time between failures* and optimized *data storage workload* at vehicular clouds. Workload is an important metric as it is directly related to the latency of content delivery. Concretely, we present a content placement algorithm that has an *optimal workload* at vehicular clouds, and formally prove its optimality within a family of algorithms. Finally, we verify the system efficacy using our own simulator based on SUMO and Python.

KEYWORDS:

CDN, Vehicular Clouds, Availability/Reliability, Peer-assisted CDN, Micro Clouds

1 | INTRODUCTION

Content delivery networks (CDNs) are an important component of today's internet. They are deployed to serve content to customers in geographical proximity, and are, in fact, now responsible for serving the majority of current internet traffic. Several studies have shown that the traditional CDN is *insufficient* to handle the rapidly increasing data content and traffic. To tackle the scalability issues, abundant research has investigated the potential of using peer-to-peer (P2P) systems to assist CDNs, namely *peer-assisted CDNs* (e.g. [1, 2, 3]). In this work, we propose a novel concept of peer-assisted CDNs, namely VC-CDN, which integrates the emerging concept of *vehicular clouds* with content delivery networks.

The concept of vehicular clouds (VCs) was proposed by several research groups independently, because cars are starting to get manufactured with built-in computation and communication equipment. Vehicular clouds bring the mobile cloud (or edge [4]) computing model to a cluster of vehicles and vehicular networks. In [5], we have proposed a new kind of vehicular cloud, namely a *macro-micro cloud (MMC)*. The MMC is specifically designed for long-lasting location-based services, for example, the monitoring service for intelligent transportation systems.

This work presents an efficient mechanism for integrating a macro-micro cloud with a CDN. The core of our mechanism is a content placement algorithm that: (i) divides data content into small partitions to reduce the workload, and (ii) assigns multiple data partitions to vehicular clouds to increase availability (or reliability). We prove that within a general family of algorithms,

[†]This research is supported in part by a research gift from Toyota InfoTechnology Center, U.S.A.

our algorithm achieves *optimal workload* – the maximum amount of data partitions stored in vehicular clouds. Moreover, our algorithm is lightweight (compared to other general optimization techniques) and hence can be used to handle dynamic scenarios in vehicular clouds. Finally, we build a simulator on top of SUMO and Python to verify the efficacy of our system.

Background

Below, we discuss close related work on peer-assisted CDN, and the foundation of macro-micro cloud (MMC) [5].

Peer-Assisted CDN: To achieve higher scalability and lower deployment cost, peer-assisted CDNs distribute the workload among multiple peers. Typical peer-assisted CDNs have two types of nodes: (i) reliable CDN servers, which are physical machines located in a CDN datacenter, and (ii) unreliable peers which are usually small and fragile machines hosted by users and may suffer churn behavior. In peer-assisted CDNs, if peers have sufficient bandwidth and are available, then a user should download content from these peers to reduce the workload and bandwidth consumption at CDN servers; otherwise, a user should contact CDN servers directly. While most peer-assisted CDNs share a similar architecture, each system addresses slightly different issues, for example, quality of service, flash crowd, etc. We refer interested readers to [1, 2, 3] for more details. In this work, we focus on the *availability* issue, which has not been studied rigorously in prior work. Concretely, we build a system that guarantees a provably level of availability.

Vehicular Clouds: The main purpose of a vehicular cloud is to provide cloud services by leveraging computation and communication resources of a cluster of connected vehicles. Research topics include integration with the vehicular ad-hoc network (VANET) [6, 7], MapReduce-like computation [8], and information-centric networking [9]. To the best of our knowledge, however, none of the existing literature has thoroughly investigated the idea of using vehicular clouds to assist CDNs. Particularly, the content placement problem has not been considered in prior work. In [10], we presented a preliminary study on using vehicular clouds as a CDN. In this work, we present an optimal algorithm, its proof, and simulation results.

Micro-Macro Cloud (MMC): As mentioned previously, the proposed system, VC-CDN, in this work is based on the macro-micro cloud (MMC) presented in our previous work [5]. We briefly go over the design of MMC and its characteristics here. The MMC is essentially a *hierarchical vehicular cloud* which consists of two components: vehicular micro clouds and vehicular macro clouds: *With vehicular micro clouds* (or simply, micro clouds), geographically nearby cars form local clusters. A micro cloud can be logically *tied* to a specific geographic region, and the cars belonging to the same micro cloud coordinate with each other to provide long-lasting services such as data storage and network services. Note that depending on the configuration, a vehicle might belong to multiple micro clouds as far as it has sufficient computation and communication resources. *With vehicular macro clouds* (or simply, macro clouds), *multiple* micro clouds behave as special nodes in a large network of connected vehicles, typically a multi-hop network. The idea of a macro cloud can be seen as an extension of Car4ICT [7], where individual connected vehicles offer data content and a variety of computation and communication services on a peer-to-peer basis over vehicle-to-vehicle (V2V) networks. Data content storage application was briefly discussed in MMC [5] and Car4ICT [7]; however, no availability study was conducted in [5, 7].

Here we summarize a number of MMC characteristics that would benefit the design of a CDN: (i) An MMC can be made to support long-lasting caching and delivery services through the coordination of cars in the micro clouds (so that the state information is not lost due to a vehicle's mobility); (ii) Wide-area service discovery and provisioning is enabled through the V2V network in the MMC. This can effectively reduce the bandwidth consumption between users and CDN servers; and (iii) For each micro cloud i in the MMC, we can quantify the probability of *availability* in terms of the *mean time between failures (MTBF)* t_i . Note that the micro cloud (MC) in the MMC might fail to remain intact due to communication failure, lack of vehicles in a geographical region, and/or machine failure. This number t_i characterizes the average time between two occurrences of failures of the micro cloud i , and can be calculated based on historical data such as penetration rate, vehicle density, traffic pattern, communication quality, etc [11]. By definition, $FR(i) = 1/t_i$ is the failure rate, i.e. a probability that the MC i suffers one failure during a unit of time. Our content placement algorithm (presented in Section 3) uses these numbers of MCs to decide the location of content placement and the number of replicated data partitions.

2 | OUR SYSTEM: VC-CDN

In this section, we present the design of our system, a vehicular cloud-based CDN (VC-CDN), which essentially integrates a CDN and an MMC. We also use a simulation to show that with a simple content placement strategy, VC-CDN caches and delivers the content efficiently. In section 3, we discuss how to optimize the performance, especially in the case when micro clouds may fail or become unavailable.

Our VC-CDN system has three components: physical CDN servers, virtual CDN servers, and an administrator. Physical CDN servers are physical machines located inside datacenter(s), and behave similarly to the traditional cloud-based CDN system. Physical CDN servers can be accessed via the internet. Virtual CDN servers behave similarly to the *peers* in peer-assisted CDNs. Virtual servers are built on top of *micro clouds* in the MMC, which can be accessed via V2V networks. An administrator is a cloud-based server that provides the directory service and can be accessed via the internet. Moreover, it is responsible to execute the content placement algorithm (presented in section 3) that assign contents to micro clouds.

The VC-CDN system provides two basic functionalities of CDNs: caching and delivering content. We present the content caching protocol in Algorithm 1, and content delivering protocol in Algorithm 2 below.

- 1: User contacts administrator to cache a data content
- 2: Administrator computes a content placement strategy
- 3: Administrator distributes the content to physical CDN server(s) and peers (micro clouds)
- 4: Administrator updates data directory
- 5: Administrator sends acknowledgment to the user

Algorithm 1: Content Caching Protocol

- 1: User contacts the administrator to download a content
- 2: Administrator looks up the directory and sends a list of servers and peers with the content to the user
- 3: **For** each data partition D **do**
- 4: **If** some peer p with D are available **then**
- 5: User downloads D from peer p
- 6: **Else**
- 7: User downloads D from physical CDN servers
- 8: **If** there are missing data partitions **then**
- 9: User download them from physical CDN servers

Algorithm 2: Content Delivering Protocol

The key to our design is in Step 2 of the content caching protocol in which the content can be partitioned into multiple pieces (*data partitions*), and each partition may be *replicated* to multiple peers. Data partition allows for a distribution of the workload among peers, and replication increases the availability in terms of the MTBF. Moreover, the existence of multiple copies of a data partition allows users to download different data partitions in parallel, which reduces average download latency.

Content delivering is similar to prior design in peer-assisted CDNs. We present the protocol in Algorithm 2 for completeness. Note that users access peers (micro clouds in an MMC) through V2V networks and access physical servers via the internet. Therefore, our design helps reduce the workload and bandwidth consumption of the physical CDN servers. We use simulation to validate this claim.

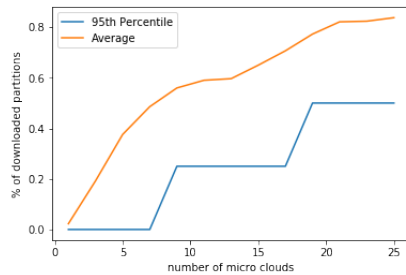


FIGURE 1 Simulation in a Manhattan grid of 10 by 10 blocks of size 200 meters.

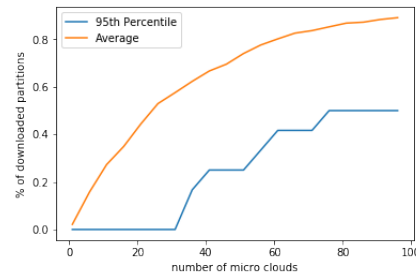


FIGURE 2 Simulation in NYC based on the SUMO tool OSMWebWizard.

Simulation and Results: Using a simulator that we built based on SUMO and Python, we can evaluate our design. In our simulator, we first run a SUMO simulation and record the position of each vehicle. Then, we feed the position data into our Python program which has two phases: preprocessing and discrete-time simulation. In the preprocessing phase, we generate micro clouds' position based on the given traffic flow following the method in [11], and divide one content file for each vehicle into 12 data partitions and randomly place the partition to the micro clouds. In the discrete-time simulation phase, we execute the traffic trace from SUMO to check whether a vehicle can download a content or not. Whenever a vehicle is close to the micro cloud with its content (specified using a *communication range* parameter), it can start downloading a data partition on a stable rate (specified using a *communication bandwidth* parameter).¹ We consider the scenario of a stable download rate and no packet loss. The purpose is to show that even with an unoptimized strategy (at the administrator), users can download a large proportion of the data partitions from peers. As a result, the workload of the physical servers is reduced in this scenario.

¹In our simulation, we adopt a simple mechanism; hence, if a vehicle is moving too fast, then it might fail to download a data partition in time.

Figures 1 and 2 show that even with simple strategy, users can download a large proportion of the content with a moderate number of micro clouds. More precisely, in Figure 1, with 17 or more micro clouds in a Manhattan grid, users can download close to 80% of partitions on average, and 95% of users can download 50% of partitions. Similar pattern can be observed in Figure 2. In these simulations, it takes 5 seconds to download each data partition and communication range is 150 meters.

3 | OPTIMAL CONTENT PLACEMENT ALGORITHM

In this section, we show how to fully make use of the strength of the macro-micro cloud by proposing a novel content placement algorithm at the administrator (Step 2 in Algorithm 1). Given a content C , the administrator needs to decide how C is partitioned and replicated, and where the data partition is stored. As mentioned previously, each micro cloud i has a parameter $FR(i)$ (failure rate). Our algorithm uses these parameters to improve the performance. More precisely, we want to have a CDN system that has guaranteed availability and reduced workload at both the physical and virtual CDN servers. A reduced workload has the benefit of delivering the data with low latency and low bandwidth consumption.

Our system has two inter-correlated goals. First, we need to ensure that content is mostly *available*, meaning that the user can download data from multiple micro clouds most of the time. Second, it is important that the *workload* at each peer is small; that is, the data stored at each micro cloud needs to be small. Our algorithm addresses these two goals simultaneously.

To facilitate the discussion, we first define the notion of availability (in terms of MTBF) for a given content c and the workload on the micro clouds to store c .

Definition 1 (MTBF of Content Delivery). For the content c , the $MTBF(c)$ is defined as the $MTBF$ of downloading the entire content from the micro clouds (without contacting the physical CDN servers).

Consider, for example, that content c is divided into $\#_c$ data partitions. In this case, a user would fail to download content c if there is a loss of a data partition(s). A partition may be lost if the hosting micro cloud(s) fail.

A data partition can be replicated multiple times to different micro clouds to increase the MTBF. If $MC(i, j)$ denote the micro cloud that stores the j -th copy of data partition i , and $FR(M)$ is micro cloud M 's failure rate, then we have

$$MTBF(c) = \frac{1}{\sum_{i=1}^{\#_c} \left[\prod_{j=1}^{\#_i} FR(MC(i, j)) \right]}$$

where $\#_i$ is the number of copies of partition i in the system.

Definition 2 (Workload of Content Caching). For the content c , the workload is defined as the maximum amount of data (related to content c) stored among all micro clouds.

Given these two definitions, the first goal is equivalent to maximizing the $MTBF(c)$, whereas the second goal is equivalent to minimizing the *workload*. For the sake of this paper, we present the algorithm that works for one content. It is then straightforward to extend it to handle multiple contents.

An obvious method for addressing the placement problem is to perform a multi-objective optimization. Unfortunately, in a practical setting, it usually requires extensive computation (e.g. using genetic or evolutionary algorithms). Therefore, this is not suitable for frequent re-computation. In the framework of the MMC, the parameters FR (and hence the $MTBF$) are time-varying due to changes in vehicle traffic patterns; hence, to ensure a stable performance, the content and data partitions need to be relocated by recalculating the optimal placement.

An Optimal Algorithm

Below we present a simple *lightweight* greedy algorithm. The key is that we do *not* optimize availability and workload at the same time. Instead, we set a threshold on the $MTBF$ that needs to be satisfied. Then under such a constraint, our algorithm optimizes the workload by replicating the “least available” data partition to the micro cloud with the lowest workload so far. Later we show that our algorithm is *optimal* in the case when all data partitions are equal size.

For lack of space, we present an algorithm to place one content only. However, we stress that it is straightforward to extend it to multiple contents. Suppose that there are n micro clouds whose $FR(i)$ is known and we have a queue Q which contains MCs in a decreasing order of $FR(i)$. For brevity, we use R to denote the $MTBF$ as per Definition 1, and T to denote the desirable

threshold of the availability. Finally, the *least reliable data partition* refers to the one with the largest failure rate, that is the data partition j with largest $\prod_{i=1}^{\#} FR(MC(i, j))$. Our algorithm is presented in Algorithm 3.

```

1:  $Q \leftarrow$  all  $n$  MCs with the decreasing order of  $FR(i)$ 's
2:  $k \leftarrow n$ 
3: while  $k > 0$  do
4:    $S \leftarrow$  dequeue the first  $k$  MCs from  $Q$ 
5:   Divide data into  $k$  partitions
6:   Place a partition to each MC in  $S$ 
7:   Calculate the availability  $R$  using Definition 1
8:   if  $R > T$  then return the placement
9:   end if
10:  while  $R < T$  and  $Q$  is non-empty do
11:     $d \leftarrow$  the least reliable data partition
12:     $m \leftarrow$  dequeue the first MC from  $Q$ 
13:    Replicate  $d$  and place it to MC  $m$ 
14:    Recalculate  $R$  using Definition 1
15:  end while
16:  if  $R > T$  then return the placement strategy
17:  end if
18:   $k \leftarrow k - 1$ 
19:   $Q \leftarrow$  all  $n$  MCs with the decreasing order of  $FR(i)$ 's
20: end while
21: return no available placement strategy

```

Algorithm 3: Content Placement Algorithm

Note that it is possible that for the current scenario, there does not exist an available strategy. A trivial example is where no micro cloud exists, and hence, the data content cannot be placed to achieve the desirable availability.

Time Complexity and Optimality Proof: Our algorithm has time complexity of $O(n^2)$, since (i) there are at most n iteration of the outer while loop (k ranges from n to 1), and (ii) in each iteration, the inner while loop (line 9 to line 14) takes at most n iteration as well. Hence, our algorithm is lightweight compared to general optimization algorithms.

Theorem 1. Among all the algorithms that divide data into partitions with equal size, Algorithm 3 achieves the optimal *workload* as per Definition 2.

Proof. The proof is given in a standard “swap” argument for showing optimality of greedy algorithms. Suppose there exists an optimal content placement Algorithm A which divides the content into a partitions $p_1, \dots, p_a$, and uses n_a MCs. Then we need to show that Algorithm 3 comes up with an assignment using the same number of partitions. Since each partition has the same size, the workload is the same for the two algorithms.

We define an *inversion* as the place of different assignment between Algorithm A and Algorithm 3. Then we use induction to prove that claim. The base case is when Algorithm A has only one inversion. Without loss of generality, assume that (i) inversions happen at partition p_1 ; (ii) A assigns p_1 to MC i_1 ; and (iii) Algorithm 3 assigns p_1 to MC i_2 . Next we show that by swapping the assignment for p_1 , the availability does *not* become smaller. That is, we need to show

$$R(p_1)FR(i_1) \geq R(p_1)FR(i_2) \quad (1)$$

where $R(p_1)$ denotes the assignment for partition p_1 excluding MC i_1 or MC i_2 . Note that since all the other partitions have the same assignment, inequality (1) above implies that after swapping, the resulting assignment still satisfies the availability threshold T . By assumption, partitions have the same size for both Algorithm A and Algorithm 3, workload remains the same. Inequality (1) holds because Algorithm 3 processes MCs in a decreasing order of FR 's. The only reason that MC i_2 is used instead of MC i_1 in Algorithm 3 is because $FR(i_1) \geq FR(i_2)$. The same argument can be used for the inductive step. That is, for an optimal algorithm with k inversions, we can always find an optimal algorithm with $k - 1$ or $k - 2$ inversions that still satisfies the threshold T . Hence, by induction, the theorem is proved. $\square$

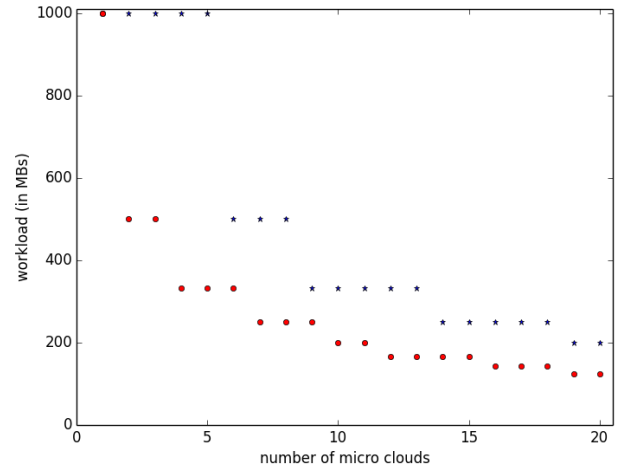


FIGURE 3 Workload under higher MTBF threshold (red dots) and lower MTBF threshold (blue dots). The parameters are: (i) Size of content: 1000 MBs. (ii) MTBF of micro cloud: 2.5 units of time. (iii) Threshold of red dots: $MTBF(\text{content}) = 1$ unit of time; Threshold of blue dots: $MTBF(\text{content}) = 5$ units of time.

Evaluation: We evaluate our algorithm using different scenarios in simulation. Figure 3 presents the result with the content of 1000 MBs. It indicates that VC-CDN is *scalable*, because as the number of micro clouds increases, (i) workload decreases, and (ii) the overhead for ensuring a higher MTBF decreases (the gap between two cases shrinks). Note that there are some missing dots in the figure, because given the number of micro clouds, it is impossible to achieve the required availability (MTBF) threshold, i.e. no feasible content placement strategy exists. We also share two interesting observations: (i) Increasing the number of micro clouds sometimes does *not* necessarily help to reduce the workload (flat part) due to the nature of the MTBF's definition; and (ii) Jumps in the figure arise from a "critical point" where adding one additional micro cloud reduces the workload. This is because the content can be divided into smaller partitions and replicated while maintaining a similar MTBF.

4 | SUMMARY

In this paper, we present a novel peer-assisted CDN built on top of vehicular clouds. We describe a simulator we built to test our system and present the evaluation. Finally, we offer a lightweight greedy content placement algorithm and prove its optimality.

References

- [1] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on peer-assisted content delivery networks," *Comput. Netw.*, vol. 116, no. C, pp. 79–95, Apr. 2017. [Online]. Available: <https://doi.org/10.1016/j.comnet.2017.02.008>
- [2] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: A practical approach to reducing cross-isp traffic in peer-to-peer systems," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 363–374. [Online]. Available: <http://doi.acm.org/10.1145/1402958.1403000>
- [3] Y. Nicolas, D. Wolff, D. Rossi, and A. Finamore, "I tube, youtube, p2ptube: Assessing isp benefits of peer-assisted caching of youtube content," in *IEEE P2P 2013 Proceedings*, Sept 2013, pp. 1–2.
- [4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.
- [5] L. Tseng, T. Higuchi, and O. Altintas, "When cars meet distributed computing: Data storage as an example," in *1st Workshop on Storage, Control, Networking in Dynamic Systems (SCNDS)*, 2017.
- [6] M. Eltoweissy, S. Olariu, and M. Younis, *Towards Autonomous Vehicular Clouds*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–16. [Online]. Available: https://doi.org/10.1007/978-3-642-17994-5_1
- [7] O. Altintas, F. Dressler, F. Hagenauer, M. Matsumoto, M. Sepulcre, and C. Sommer, "Making Cars a Main ICT Resource in Smart Cities," in *34th IEEE Conference on Computer Communications (INFOCOM 2015), International Workshop on Smart Cities and Urban Informatics (SmartCity 2015)*. Hong Kong, China: IEEE, April 2015, pp. 654–659.
- [8] S. Arif, S. Olariu, J. Wang, G. Yan, W. Yang, and I. Khalil, "Datacenter at the airport: Reasoning about time-dependent parking lot occupancy," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2067–2080, Nov 2012.
- [9] E. Lee, E. K. Lee, M. Gerla, and S. Y. Oh, "Vehicular cloud networking: architecture and design principles," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 148–155, February 2014.
- [10] L. Tseng, J. DeAntonis, T. Higuchi, and O. Altintas, "Peer-assisted content delivery network by vehicular micro clouds," in *7th IEEE International Conference on Cloud Networking, CloudNet 2018, Tokyo, Japan, October 22-24, 2018*, 2018, pp. 1–3. [Online]. Available: <https://doi.org/10.1109/CloudNet.2018.8549196>
- [11] T. Higuchi, F. Dressler, and O. Altintas, "How to keep a vehicular micro cloud intact," in *VTC Spring*. IEEE, 2018, pp. 1–5.

How to cite this article: L. Tseng, J. Mantica, J. DeAntonis, M. Aloqaily, T. Higuchi and O. Altintas (2019), Peer-Assisted Content Delivery Network by Vehicular Clouds: Algorithm and Evaluation, *Internet Technology Letters*, 2019;00:1–6.