Content Routing In the Vehicle Cloud

Yu-Ting Yu, Thomas Punihaole, Mario Gerla and M. Y. Sanadidi
Computer Science Dept.,
University of California, Los Angeles, CA 90095, USA
{yutingyu, punihaol, gerla, medy}@cs.ucla.edu

***Abstract*—The advance of vehicle technology has brought to us the concept of Vehicle Cloud (VC). When Vehicle-to-Infrastructure (V2I) communications are unavailable, have failed or are too costly, Vehicle-to-Vehicle (V2V) communications fill the gap enabling VC services. In this paper, we discuss an important VC service, content-based routing, that allows future VC applications to store, share and search data totally within the Cloud. We review the critical issues of content discovery and forwarding in a content-based VC. Moreover, we introduce Bloom-Filter Routing (BFR), a proactive content discovery scheme for popular contents, to tackle the mobility and large content population challenges. BFR is compared to popular reactive content discovery schemes in practical VANET scenarios. The results show that proactive content discovery (i.e. BFR) suits non-sharable data, while reactive content discovery works well with popular sharable data. Consequently, we propose an adaptive hybrid approach that combines proactive and reactive strategies.**

***Index Terms*—Vehicle Cloud, VANET, Routing, Hybrid**

I. Introduction

Recent developments in the car industry have suggested the emergence of a Vehicle Cloud (VC) as a framework for vehicular services. Like the Internet Cloud, VC will provide various services such as communication, storage, and computing for a range of applications from safe navigation to on-road entertainment, live video streaming, Internet access, to urban surveillance and massive net games.

One challenge in VC design is the ability to connect to the Internet and to its resources, including Cloud Services. While Vehicle-To-Infrastructure (V2I) communications (via DSRC, WiFi and 3G/4G) are becoming increasingly more reliable in civilian settings, in environments such as battlefields and urban emergence, infrastructure access will be very limited, suggesting that Vehicle-To-Vehicle (V2V) communications will be used for critical vehicle cloud services like internal data storage, searching and sharing.

Recently, Content-Based Networking (CBN) has attracted much attention as a method for searching content. In CBN, a data object (viewed as a chain of chunks) is searched and retrieved based on its identity instead of the IP address of the node on which it resides. While several independent CBN designs [1-3] exist, all designs have the following common attributes: (1) receiver-oriented chunk based transport, (2) in-network per-chunk caching, (3) name-based forwarding, and (4) uniquely identifiable content naming. Intuitively, in-network caching is beneficial to highly mobile vehicular scenarios. A data retrieval failure due to intermittent connectivity can be recovered more quickly by leveraging distributed caches.

A major design issue in CBN is content discovery and request forwarding. The content consumers initiate requests. Request forwarding latency is critical to overall time-sensitive data retrieval (e.g. orders and battlefield images in military networks), in particular when the content producers are mobile [5]. Two common approaches to mitigate latency are frequent routing updates in a proactive approach and content replications in a reactive approach. However, both solutions can backfire in VANET. Routing update overhead can become excessive in high mobility and large name population; content replication may not be feasible for large or real-time data. In this paper, we advocate a hybrid forwarding framework that adaptively performs proactive or reactive content discovery and forwarding based on the content characteristics for time-sensitive data. We observe that the services generating time-sensitive data can be categorized as three types: popular sharable data services, popular non-sharable data services, and unpopular data services. Therefore, we propose to utilize the hierarchical content naming [2] to categorize the time-sensitive data and apply the most suitable strategy to each category. To realize proactive content dissemination and discovery in VC, we propose a Bloom-Filter based Routing algorithm (BFR). In conjunction with a self-organized geographical hierarchy, BFR reduces the content discovery cost between clusters.

The organization of this paper is as follows. In section II, we review and evaluate existing name-based routing approaches. In section III, we propose our hybrid design intended to provide effective support to the various categories. We introduce BFR in section IV, and report on its implementation and preliminary results in section V. Section VI concludes the paper.

II. Name-Based Routing

Content name-based routing is different from host IP-based routing in two aspects. First, the number of content names to consider in name-based routing is significantly larger than the number of IP addresses. Second, in host-based networks, each IP address is associated with one host interface. In contrast, copies of a data object chunks may exist at different locations in CBN. The resultant multi-source nature of data transfers must

This paper is based on work supported by the National Science Foundation under Grant CNS-1205757 and Singapore DSO National Lab under Grant DSCCO10350.

be taken into account in name-based routing algorithm design.

We discuss below three name-based routing approaches for the vehicle cloud: reactive routing, proactive routing, and opportunistic forwarding.

A. Reactive name-based routing

The reactive name-based routing consists of two phases: content discovery and content forwarding. Reactive content discovery is accomplished via flooding. Consumers initiate content retrieval by flooding their requests. Once the data object is found, it is forwarded back to the consumer by an appropriate forwarding scheme. In vehicular network, popular forwarding schemes are based on AODV[6], DSR[7], or GPSR[8].

Content-Centric Network (CCN) [2] is an example of content-based network with reactive routing design. In CCN, the requests are flooded if the routing information is unknown. Data is forwarded in reverse on the path(s) from which the query arrived.

Flooding-based content discovery can quickly find the nearest cached data objects. However, forwarding algorithms such as AODV are vulnerable in a vehicular setting due to path intermittence [9]. The reactive forwarding approach also suffers from the cached fragment phenomenon. That is, path discovery may lead to a cache that holds only a few data chunks and thus gets quickly exhausted. One must frequently re-flood to discover the remaining chunks. Frequent flooding from multiple data requesters may cause network congestion and nullify the cache benefits.

B. Proactive name-based routing

Another routing approach is based on proactive data object advertising. Flooding is not required since the object locations are pre-announced. Yet, the in-network storage is still useful if there are cached copies along the request path. Thus, the foreground search overhead is much lighter than that in reactive routing. On the negative side, there is significant background advertising overhead.

DONA [1] is an example of proactive routing. DONA maintains a name-resolution hierarchy. Nodes must register all data objects and their locations to Resolution Handlers (RHs), which form a hierarchical structure; content searches are done by following a recursive resolution process similar to DNS.

Using proactive approaches in vehicular network is challenging because of several reasons. A CBN would require storing location information of at least 10^{12} data objects [10]; maintaining the location information for all content names means significant cost in vehicular network. Thus, only a selected set of prefixes can be advertised.

C. Opportunistic forwarding

The previous two approaches require an active (i.e., re-active or a pro-active) content discovery phase. To avoid the associated overhead, a passive type of discovery/forwarding algorithm, called opportunistic forwarding, has also been pursued. Opportunistic forwarding is originally designed for delay-tolerant networks [10][11]. Requests and data are disseminated opportunistically through node encounters. The Haggle [12] architecture is built based on this approach.

The "carry-and-forward" behavior significantly reduces the request flood and location announcement overhead. However, opportunistic forwarding was developed for delay tolerant networks and is obviously not suitable for time-sensitive applications.

III. Hybrid Name-Based Routing Framework

Observing that existing routing approaches have complementary advantages and disadvantages, we propose a hybrid routing framework for time-sensitive data services. The hybrid design uses different routing methods for each content type. In the following section, we classify time-sensitive data services and pair them with suitable routing approaches.

A. Data service categorization

We classify the service generating time-sensitive contents into the following three categories:

(1) Popular sharable data services (Type A): these services generate time-sensitive sharable data. Sharable data are generally accessible by most users at least in a local geographic scope and are small enough to be cached at relay nodes. General orders and emergent announcements are examples of this category. Since these data can be cached by most vehicles, content re-discovery can quickly retrieve data without large-scale flooding. Reactive routing is the most suitable routing scheme for this category.

(2) Popular non-sharable/non-cacheable data services (Type B): This type indicates popular services providing data that are non-sharable or non-cacheable. For example, contents such as access-restricted orders and highly-credential information are non-sharable. Large-size popular video clips that are too large to be completely cached at relay nodes in high mobility are non-cacheable; cached fragment phenomenon is likely to emerge. Yet, the service providers may be "popular" since many content consumers will ask for non-sharable data from the same providers. For popular time-sensitive data services, it is worth to proactively advertise the path to mobile providers in order to shorten response time and save search traffic overhead.

(3) Unpopular data services (Type C): one example of this type is private messaging. For unpopular contents, the individual chunks are not expected to stay cached in the network for very long due to the limited in-network storage. Therefore, the in-network caching is beneficial mainly for recovery in very intermittent connectivity. Large-scale blind request floods for unpopular contents should be avoided if at all possible. In delay tolerant applications, opportunistic forwarding can be used to avoid route maintenance costs. However, if the data is strictly time-sensitive, it may be necessary to sacrifice some content discovery overhead for better quality of service, depending on the strictness of time requirements. In this case, the reactive routing may be the most reasonable mechanism in a receiver-driven content-based network since it can achieve lower response time without topology knowledge for all service providers. In this paper, we focus our study on the content-based

routing performance for the popular data services.

B. Content naming

To adapt the routing to content types, we name each data chunk with the following format: ***/Category/Service_name/****content_name/****.*** *Category* represents one of the three above mentioned categories. It is then easy for the system to decide which routing approach to apply at runtime. *Service_name* is an identifier of a service. Note that a service may be provided by multiple provider nodes. *content_name* includes the unique content identifier that is defined by the application. For example, a map service may name the Westwood map as /type_a/map/los_angeles/westwood/, and a battlefield video service may name an access-restricted video clip as /type_b/video/*video_clip_name*/. For popular data services, we assume the existence of a pre-defined, possibly changing-over-time, prefix list that is retrievable from known locations.

IV. BLOOM-FILTER ROUTING (BFR)

The hybrid framework requires an efficient proactive name-based routing algorithm for popular non-cacheable/sharable data. However, due to the high mobility and limited bandwidth in vehicular networks, previous proactive designs that require all contents to be announced may create too large overhead for VC. Fortunately, although popular non-sharable data service traffics may dominate the Content Based Network traffic, these services are likely to be provided by a few mobile providers. We assume the popularity of non-sharable data services follows Zipf's distribution as the web traffic [13-14].

Therefore, it is reasonable to use bloom-filters [15] to announce only the popular prefixes. We define the prefix as the part /Category/service_name/ in the name. Bloom-filters are widely used in applications such as Internet caching and P2P content discovery [16-18]. The advantages of bloom-filters include (1) bloom-filter size adjustable to the max number of prefixes and; (2) bloom-filter aggregation.

To handle the large size of typical urban VANETs, the network is hierarchically organized into clusters that match the geographic partitions of an urban layout. Figure 1 shows a rectangular partitioning that fits a Manhattan grid topology as an example. The bloom-filters advertise the presence of name prefixes in the corresponding clusters. The bloom-filter advertisements (BFAs) are propagated level-by-level in the cluster hierarchy. At the node level, only the mobile providers of popular non-cacheable/sharable data services are required to send BFAs. Cluster heads are responsible for aggregating BFAs into a cluster BFA and announce it to the next level. The content search mimics a DNS query that moves up level by level. Note that each node handles BFAs in the same way as they handle all other contents. That is, relay nodes cache the overheard BFAs during the process. Therefore, the routing service installed at each node can leverage cached BFAs to redirect requests to the end destination cluster and thus improve the content search performance. This BFA caching ability is especially useful under mobility since the relay nodes will continue forwarding requests even when a cluster head moves out of range. We call this method Bloom-Filter Routing (BFR).

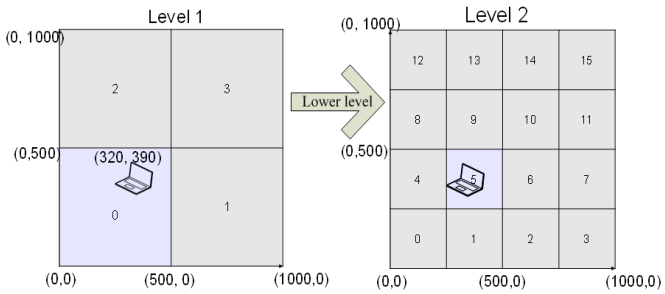

Fig 1. Rectangle clustering example

A. Clustering

The cluster hierarchy is organized based on road map knowledge. We explain the hierarchical clustering by Figure 1. The network spans a rectangular area. The area is recursively divided into multiple levels. At level 1, the total area is divided into four quarters. At level 2, each quarter is further divided into four small quarters. The clusters at the bottom level are called *leaf clusters*.

In practice, leaf clusters represent road segments. Road segments intersections form a higher level clusters and so on. Each vehicle reads the number of levels and the cluster maps when it enters the local VANET. A node associates itself to a cluster at each level based on its current location. For example, the node at (320, 390) in figure 1 belongs to cluster 0 at level 1 and to cluster 5 at level 2. The reader may note a similarity between the BFR hierarchical structure and the GHT (Geographic Hash Table) structure in [4]. Briefly speaking, in GHT, the data is saved to one or several locations that are calculated by the hash of the data name and is then retrievable via the hierarchical structure. Both BFR and GHT scale with *O(logN)*, where *N* is the name space size. BFR offers the advantage of aggregation with the risk of false positive. In all, BFR was judged more practical than GHT for the non-sharable data services since it propagates the provider information instead of publishing and replicating all the non-sharable or large data that are published. In BFR, the requests are directed toward the service providers via a structured request propagation network. The load balancing problem is naturally addressed by the in-network caching.

B. Inter-Cluster Forwarding

At the MAC level all transmissions in BFR are broadcast or pseudo-broadcast. At the network level, to ensure robustness under high mobility, geographical routing is used for both inter-cluster and intra-cluster data packet forwarding [8]. Request packets are routed using the BFR hierarchy. Each request packet contains three routing fields: *destination cluster ID*, *destination cluster level* and *last hop distance* to destination cluster. Nodes receiving a packet forward the packet only when it is closer to the destination. To calculate the distance to destination cluster, we define the *anchor coordinate* of a cluster as the coordinate of the central point of the corresponding

geographic partition. A node approximates its distance to the destination cluster by calculating its distance to the cluster's anchor coordinate. Initially, the first destination of a request is set to the anchor coordinate of the leaf cluster the node initiating the request belongs to. Unless some relay nodes have up-to-date knowledge regarding the data or service locations, that is, up-to-date cached BFAs, the requests are forwarded level-by-level up to the root cluster head and then level-by-level down to the leaf cluster head of the providers. Note that, however, since the BFA propagations also utilize in-network caching, it is likely that the request is redirected toward the service provider before the request reaches the root cluster head.

C. Intra-Cluster Forwarding

We mimic the geographical routing inside a cluster using a technique similar to LFBL [19]. Nodes monitor distance to other nodes by reading the distance travelled field in every packet. The distance can be used to emulate geographical routing in absence of GPS and destination coordinates [19].

When a node receives a packet, it backs off a random period proportional to the estimated distance to the destination. If it does not overhear transmissions of the same packet during the back-off period, it broadcasts the packet.

The estimated distance is not maintained by explicit control packets and it is cleared after a short expiration time. Thus, distances are not always available. If distance to a node is unavailable, the packet is flooded within the cluster. Since the mobiles must frequently communicate with the cluster head, we expect the distance between cluster head and mobiles to be usually available.

D. Cluster head election

To perform BFA propagation and content search, each cluster must elect a cluster head. When a node enters a cluster, it broadcasts a *join* message to the leaf cluster to request the identity of the leaf cluster head. If no response is received, the node declares itself as the cluster head.

Ideally, the cluster head should be close to the cluster center (and/or partition center) so that the BFA and request propagation traffics are minimized. However, cluster head switching overhead may become large under high mobility. Therefore, we keep the cluster head switching frequency low by re-electing the cluster head only when the previous cluster head leaves its leaf cluster, and reduce the traffic by utilizing BFA caching and request redirection.

The next level cluster head is the node closest to the center among the leaf clusters. This design makes it easier to narrow down the searching scope of non-leaf cluster heads and helps reduce content search traffic.

E. Prefix announcement

A mobile node periodically sends its own BFA, which summarizes the popular non-sharable/non-cacheable data service prefixes it holds, to its leaf cluster head. Note that only the providers who can provide the full contents should announce the prefix, nodes holding partially cached chunks do not claim themselves as providers. If a node is not a provider of any such services, it does not send node-level BFAs. We assume the service providers can judge if the contents they provide are sharable. The popularity of services is decided based on prior knowledge such as the statistics collected from previous missions.

The node BFA is stamped with *node ID*, *leaf cluster ID* and *level*. Level *i*-cluster heads collect BFAs from cluster members and aggregate them into cluster BFAs. The cluster BFA is stamped with the cluster head's *level i cluster ID*, *level i-1 cluster ID* and *i-1* and then sent up to level *i-1* cluster head.

To balance the tradeoff between overhead and accuracy, the update frequency is set proportional to the degree of change of current BFA. The degree of change is defined as the number of bits newly set compared to previous BFA.

The nodes cache the overheard BFAs. These BFAs expire after a short time period. BFA caching is helpful because: (1) the overheard BFAs are used to redirect the requests during the content search process thus assisting the cluster heads in the routing of queries, and; (2) BFAs serve as backups when the cluster head is temporarily down, e.g., during the cluster head re-election process.

F. Name-based content search

To search for a data object, a node sends a request that carries the *content name* and *destination address* to the leaf cluster head. The destination address is a *(target cluster ID, target cluster level)* pair when the destination node is unknown. Otherwise, the destination address is the destination node's address.

A relay node, upon receiving a level *i* request packet, searches its cached BFAs for the name prefix. If a relay node has a match in a cached level *j*-BFA and *j* is greater than *i*, it redirects the request to the destination indicated in the matched BFA. Otherwise, the request is forwarded up to the cluster head. Cluster heads follow the same process as relay nodes. However, if there is no match in cluster heads' BFA, the request is forwarded to level *i-1* cluster head. If no match is found in the top BFA, the query is flooded using CCN routing.

Note that since all transmissions are broadcast and the non-leaf cluster heads' locations are restricted in the center of the clusters, the chance that useful BFAs are cached on the path is high. Therefore, it is likely that requests do not have to traverse the whole hierarchy to find the data. In addition, once a node has received the data it requested from the service provider, it can skip the content discovery phase and send its following requests directly to the content provider using techniques similar to the intra-cluster forwarding.

V. Implementation and Experiments

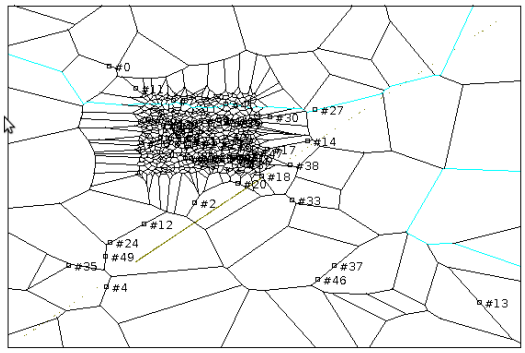

Fig 2. Experiment scenario

TABLE I
POPULAR SHARABLE DATA: AVERAGE RESULTS

		BFR	CCN
Response time (ms)	Mean	221.44	179.68
	CI	[75.8, 364.4]	[86.1, 273.3]
Request traffic (Bps)	Mean	88.53	91.02
	CI	[74.9, 102.2]	[77.0, 105.0]
Data traffic(Bps)	Mean	4795	4990
	CI	[3661.7, 5928.3]	[3793.6, 6186.4]

We have implemented both reactive and proactive name-based routing procedures. The reactive name-based routing implementation follows the CCN [2] design. The proactive name-based routing implementation follows our BFR design. The code was written in C. The hash functions used for BFR are elf32, SDBM, DJB, DEK and BP. We evaluate the two implementations with Common Open Research Emulator (CORE) [20]. CORE is a scalable and efficient emulator that virtualizes the network stack. The MAC and physical layer connectivity is emulated using Extendable Mobile Ad-hoc Network Emulator (EMANE) [21].

We simulate a city VANET scenario (see figure 2) in which 50 vehicles with speeds range from 15 mph to 35 mph are moving in the 1km^2 map. The mobility trace is generated by VanetMobiSim [22]. It simulates vehicle behaviors considering traffic lights, congestion, lane changing, acceleration, car following model, etc. We measure the response time, which is defined as the interval from first request sent, to first data chunk received. We also measure request and data traffic rates. All experiments are repeated 30 times for all scenarios using different starting times. Confidence intervals (CI) are reported.

A. Type A: popular sharable data service

Our type-A service scenario includes ten data consumers downloading the same data (for example, traffic conditions) from one mobile data provider. All downloads start at the same time after the network is initialized. The routing algorithms are assumed stabilized at the beginning of each run.

Table 1 summarizes the average results of all flows in 30 runs. As expected, CCN achieves shorter response time in this scenario while generating slightly higher request and data traffic. The reason is that since the data is sharable, the flooding-based content discovery leverages the cache and does not create too much traffic or congestion. In contrast, BFR requests must traverse longer paths to find data.

To take a closer look, we present snapshots of the response

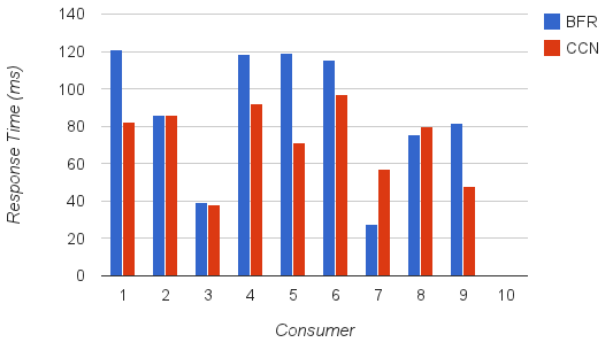

Fig 3. Popular sharable data: response time

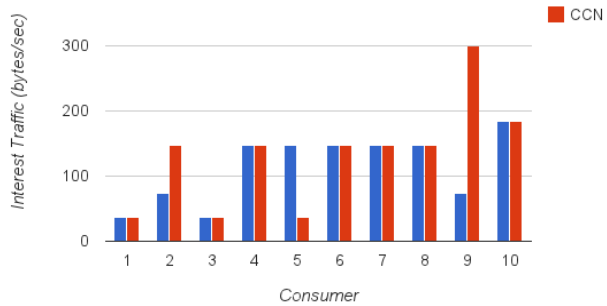

Fig 4. Popular sharable data: request traffic

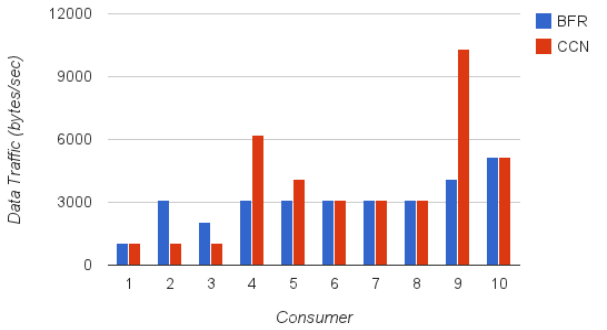

Fig 5. Popular sharable data: data traffic

time, request traffic, and data traffic of BFR and CCN for each consumer from one run. Consumers experience different performance due to different distance from the source. In Figure 3, CCN achieves better response time for the majority of the consumers. Figure 4 confirms that flooding does not generate significantly more traffic than BFR in this scenario. In figure 5, we observe that CCN data traffic is basically proportional to interest traffic. The reason is that in CCN data travels back on all the paths as from which came the requests [2]. Thus, redundant data is received from multiple paths when the request is flooded. Multi-path forwarding improves robustness. However, in heavy network load, this strategy aggravates congestion.

B. Type B: popular non-sharable data service

Here we use the same scenario again. However, ten data consumers request ten different real-time generated files from one single data provider. This may reflect, for example, ten soldiers check credential video information for the next mission destinations. The information is delay-sensitive, non-cacheable and not very sharable across soldiers who have been assigned different missions, especially if the information is

TABLE II
POPULAR NON-SHARABLE DATA: AVERAGE RESULTS

		BFR	CCN
Response time (ms)	Mean CI	88.03 [84.7,91.4]	194.44 [189.1,199.7]
Request traffic (Bps)	Mean CI	202.85 [181.9,223.8]	381.15 [363.3,399.0]
Data traffic(Bps)	Mean CI	5179.33 [4378,5980.6]	7800 [7356, 8244]

access-restricted or personalized.

Table 2 shows the average results. Since the data is non-sharable, flooding does not benefit from caching. Therefore, frequent flooding results in congestion and triggers the hidden-terminals problem even in the simple ten flow scenario. BFR improves the response time by about 55% compared to CCN.

The snapshots in figure 6 and 7 confirm that CCN introduces a much higher request volume and consequently much more data traffic than BFR due to multipath flood.

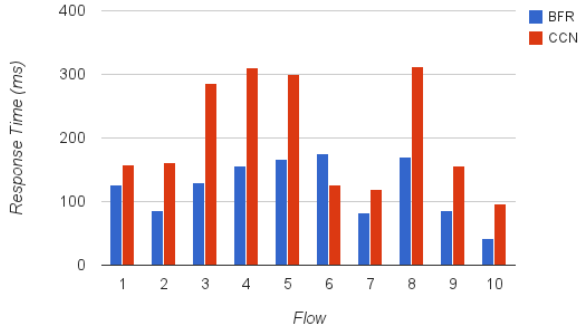

Fig 6. Popular non-sharable data: response time

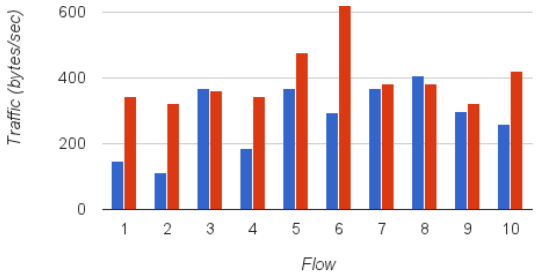

Fig 7. Popular non-sharable data: request traffic

VI. CONCLUSIONS

In this paper, we discussed a hybrid content-based routing framework for the Vehicle Cloud, which combines proactive and reactive content-based routing into one adaptive framework. We compared the proactive mode with BFR in an implementation testbed. Our preliminary results show that the reactive approach improves response time by 19% for popular sharable data but create congestions for non-sharable data. Proactive approach achieves 55% shorter response time and 47% less traffic for non-cacheable/sharable data retrieval. This confirms the validity of a hybrid content-based routing in vehicular clouds. Future work will evaluate the proposed framework under real content distributions and will assess the content-based forwarding performance for unpopular data.

REFERENCES

[1] T. Koponen, M. Chawla, B. G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 181-192, 2007.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in CoNEXT 2009, 5th international conference on Emerging networking experiments and technologies, Dec. 2009, pp. 1-12.

[3] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone, "Design considerations for a network of information," in Proceedings of the 2008 ACM CoNEXT Conference, 2008, pp. 1-6.

[4] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a geographic hash table for data-centric storage," in Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, 2002, pp. 78-87.

[5] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of Information-Centric networking (draft)," 2011.

[6] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), 1999, pp. 90-100.

[7] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: the dynamic source routing protocol for multihop wireless ad hoc networks," in Ad hoc networking, Dec. 2000, pp. 139-172.

[8] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in Proceedings of the 6th annual ACM/IEEE International Conference on Mobile Computing and Networking, 2000, pp. 243-254.

[9] J. Härri, F. Filali, and C. Bonnet, "Performance comparison of AODV and OLSR in VANETs urban environments under realistic mobility patterns," in Med-Hoc-Net 2006, 5th Annual Mediterranean Ad Hoc Networking Workshop, IFIP, Jun. 2006.

[10] Amin Vahdat and David Becker. 2000. Epidemic routing for partially-connected ad hoc networks. Technical report.

[11] B. Burns, O. Brock, and B. N. Levine, "MV routing and capacity building in disruption tolerant networks," in Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., vol. 1., 2005, pp. 398-408.

[12] J. Su, J. Scott, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M. Lim, and E. Upton, "Haggle: Seamless networking for mobile applications," in Proceedings of UbiComp 2007, Sep. 2007, pp. 391-408.

[13] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang, "Does internet media traffic really follow zipf-like distribution?" SIGMETRICS Perform. Eval. Rev., vol. 35, no. 1, pp. 359-360, Jun. 2007.

[14] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in INFOCOM '99, vol. 1, Mar. 1999, pp. 126-134 vol.1.

[15] Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13, 7 (July 1970), 422-426.

[16] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," in Internet Mathematics, vol. 1, 2002, pp. 636-646.

[17] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. 2000. Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Trans. Netw. 8, 3 (June 2000), 281-293.

[18] P. Gu, J. Wang, and H. Cai, "ASAP: An advertisement-based search algorithm for unstructured peer-to-peer systems," Parallel Processing, International Conference on, vol. 0, pp. 8, 2007.

[19] M. Meisel, V. Pappas, and L. Zhang, "Ad hoc networking via named data," in Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture, 2010, pp. 3-8.

[20] Jeff Ahrenholz, Claudiu Danilov, Thomas R. Henderson, and Jae H. Kim. 2008. CORE: A real-time network emulator. In Proceedings of the 27th military communication conference (MILCOM '08). IEEE.

[21] Extendable Mobile Ad-hoc Network Emulator (EMANE), http://cs.itd.nrl.navy.mil/work/emane/index.php

[22] J. Härri, F. Filali, C. Bonnet, and M. Fiore, "VanetMobiSim: generating realistic mobility patterns for VANETs," in Proceedings of the 3rd international workshop on Vehicular ad hoc networks, 2006, pp. 96-97.