Enabling High-Bandwidth Vehicular Content Distribution

Upendra Shevade, Yi-Chao Chen, Lili Qiu, Yin Zhang, Vinoth Chandar, Mi Kyung Han,
Han Hee Song, You Suk Seung
The University of Texas at Austin
{upendra,yichao,lili,yzhang,vinothc,hanmi2,hhsong,ysseung}@cs.utexas.edu

Abstract — We present VCD, a novel system for enabling high-bandwidth content distribution in vehicular networks. In VCD, a vehicle opportunistically communicates with nearby access points (APs) to download the content of interest. To fully take advantage of such transient contact with APs, we proactively push content to the APs that the vehicles will likely visit in the near future. In this way, vehicles can enjoy the full wireless capacity instead of being bottlenecked by the Internet connectivity, which is either slow or even unavailable. We develop a new algorithm for predicting the APs that will soon be visited by the vehicles. We then develop a replication scheme that leverages the synergy among (i) Internet connectivity (which is persistent but has limited coverage and low bandwidth), (ii) local wireless connectivity (which has high bandwidth but transient duration), (iii) vehicular relay connectivity (which has high bandwidth but high delay), and (iv) mesh connectivity among APs (which has high bandwidth but low coverage). We demonstrate the effectiveness of VCD system using trace-driven simulation and Emulab emulation based on real taxi traces. We further deploy VCD in two vehicular networks: one using 802.11b and the other using 802.11n, to demonstrate its effectiveness.

1. INTRODUCTION

Vehicular networks have emerged from the strong desire to communicate on the move [6, 7, 21, 46]. Car manufacturers all over the world are developing industry standards and prototypes for vehicular networks (e.g., [9, 13, 44]). Existing works on vehicular networks often focus on low-bandwidth applications, such as credit card payment, traffic condition monitoring [14], Web browsing [6, 7], and VoIP [7]. We explore how to support high-bandwidth applications (e.g., video streaming) in vehicular networks.

Challenges and opportunities: Cellular networks, despite good coverage, still have limited bandwidth and incur high cost. For example, many cellular service providers in US, like AT&T, T-mobile, Sprint, Verizon, charge around $60 per month for 5GB data transfer and $0.2/MB afterwards [33]. 5GB data transfer can only support 0.1Mbps for 111 hours ($<$ 5 days)! The cellular service price in many other countries are similar or even higher [48]. Moreover, many mobile broadband providers restrict or limit large data exchanges, including streaming audio, video, P2P file sharing, JPEG uploads, VoIP and automated feeds [33]. According to the international poll of 2700 Devicescape customers [38], 81% smartphone users prefer Wi-Fi over 3G cellular for data services. Therefore there is strong need for supporting high-bandwidth applications in vehicular networks using Wi-Fi.

A natural way is to let a vehicle download content from the Internet when it meets an access point (AP) [7, 21]. However, it is challenging to meet high bandwidth requirement since vehicles often move at a high speed and thus the contact time between vehicles and APs tends to be short (e.g., [14] reported that 70% of connection opportunities are less than 10 seconds). In addition, it is often expensive to provide dense high-speed Internet coverage at a large scale. As a result, if vehicles fetch desired content on-demand from the Internet during their contact with an AP, the amount of data fetched may be insufficient to sustain the data rate required by applications such as video streaming when vehicles are outside the communication range of any APs.

With recent advances in wireless technology, Wi-Fi capacity has grown rapidly and can be at least an order of magnitude higher than typical Internet access link connectivity. For example, IEEE 802.11n can offer up to 600Mbps PHY data rate using 4 antennas. We performed a measurement experiment using a laptop equipped with NetGear WNDA3100 on a vehicle communicating with a NetGear WNDR3300 AP deployed near the road. We got 4.6Mbps using 802.11b, 22.2Mbps using 802.11g, and 39.7Mbps using 802.11n (2x2 MIMO) on 2.4GHz frequency, and 56.1Mbps using 802.11n on 5GHz. In comparison, DSL throughput ranges between 768Kbps to 6Mbps [3], which is an order of magnitude slower. The gap between the wireline and wireless capacity is likely to increase further (e.g., due to the availability of new spectrum, such as whitespace, and advances in antenna and signal processing technology). Such large gap suggests that in order to enjoy high wireless capacity, we should proactively replicate content beforehand to the APs that a vehicle is likely to visit. While the idea of replication is natural, *how to replicate the content given the limited wireline and wireless resources and uncertainty in vehicular trajectory* is an open research question that we address.

ACM CoNEXT 2010, November 30 – December 3 2010, Philadelphia, USA.

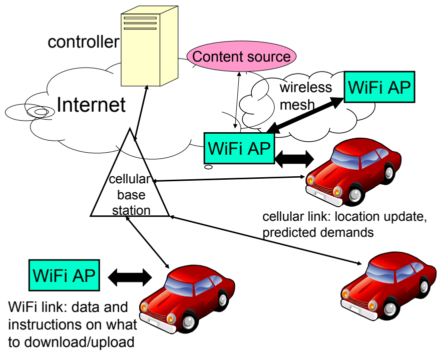


Figure 1: VCD architecture

Approach and contributions: In this paper, we develop a replication strategy that effectively exploits the synergy among (i) Internet connectivity, which is persistent but has limited coverage and relatively low bandwidth, (ii) local wireless connectivity, which has high bandwidth but short contact duration, (iii) vehicle relay connectivity, which has high bandwidth but high delay, and (iv) mesh connectivity among APs, which is persistent and has high bandwidth but low coverage. In particular, we optimize replication through wireline network and wireless mesh networks based on predicted mobility and traffic demands. Moreover, we opportunistically exploit the mobility of the vehicles to extend the coverage of the Internet and mesh connectivity. Even if only a small fraction of APs have Internet and mesh connectivity, by having the vehicles themselves relay content, one can potentially replicate content to a much larger number of APs. In essence, vehicle mobility has the potential to significantly increase the network capacity [23] and reduce future content access delay. Note that many mobile devices, such as smartphones, support the use of cheap external storage cards, which can help mitigate potential storage concerns regarding carrying traffic for other users in the system [43].

To this end, we develop a novel **V**ehicular **C**ontent **D**istribution (VCD) system for enabling high-bandwidth content distribution in vehicular networks. As illustrated in Figure 1, VCD consists of vehicles, APs with and without Internet access (some of which may form a mesh network), content server on the Internet (e.g., Web servers), and a controller. Vehicles submit location updates and content requests to the controller via cellular links. The controller optimizes the replication strategy based on predicted mobility and traffic demands, and instructs the APs to carry out the replication strategy. To enhance reliability and scalability, the controller can be replicated on multiple nodes. APs are deployed along road sides (e.g., at gas stations and/or coffee shops) to allow vehicles on the road to opportunistically communicate with them. The APs prefetch content as instructed by the controller. Whenever a vehicle encounters an AP, the AP tries to send the requested content from its local storage if the content is available locally. Otherwise, the AP tries to fetch the content from an AP in the same mesh network if one is available. If no such AP is found, it fetches content from the Internet when it has Internet connectivity. In addition to sending the content that the vehicle itself needs, the AP may also send the vehicle content that can then be relayed to other APs, or fetch from the vehicle content that can be served to other passing vehicles later.

VCD systems are easy to deploy in practical settings. For example, a vehicular service provider (VSP) can install its own APs and/or subscribe to existing wireless hotspot services. Since it is easy to place a stand-alone AP than hooking it up with Internet connection, VCD is designed to explicitly take advantage of APs with and without Internet connectivity. An AP without Internet connectivity is still useful since it can serve as a static cache, which vehicles can upload content that can be served to other passing vehicles in the future.

VSPs can offer content distribution service to taxis, buses, subways, and personal vehicles. We focus on taxis and buses that offer high-bandwidth content distribution as a value added service to their passengers. These vehicles have low-cost mobile devices on board for playing downloaded content. Such mobile devices can be installed by either the taxi/bus companies or VSPs. Since the mobile devices can be powered by the vehicles, power consumption is not an issue. The mobile devices interact with APs and the VCD controller to report required information (e.g., location update and predicted traffic demands) and follow their instructions.

The key contributions of VCD include:

- *Optimized wireline and mesh replication.* To fully take advantage of short contact time between APs and vehicles, we replicate content in advance to the APs that will soon be visited by the vehicle. A distinctive feature of our replication scheme is that it is based on optimization. Specifically, we explicitly formulate a linear program (LP) to optimize the amount of data that can be delivered before the deadline under the predicted mobility pattern and traffic demands subject to given resource constraints (e.g., short contact time and limited link capacity). The formulation involves addressing challenging modeling issues and is a valuable contribution. In contrast, previous works either focus exclusively on protocol-level optimization of one-hop communication between vehicles and APs (e.g., [7, 12, 14, 34]), or rely on heuristics to guide data replication [15], or completely ignore the resource constraints [19], which are crucial in vehicular networks. Our formulation is highly flexible and can support both wireline replication (Section 2.2) and mesh replication (Section 2.3). The formulation can be efficiently solved using standard LP solvers (e.g., *lp_solve* [30] and *cplex* [17]) owing to modern interior-point linear programming methods.
- *Opportunistic vehicular replication.* To further extend the coverage of the Internet and wireless mesh networks, we develop vehicular replication to opportunistically take advantage of local wireless connectivity and vehicular relay connectivity (Section 2.4). Different from traditional vehicle-to-vehicle (v2v) communication, our scheme leverages the APs as the rendezvous points for replicating content among vehicles since vehicle-to-AP communication is easier to deploy and such contacts are generally easier to predict than v2v contacts.
- *A new algorithm for mobility prediction.* For our replication optimization algorithms to be effective, it is critical to predict the set of APs a vehicle will visit in a future interval with high accuracy. Given the high driving speeds, diverse and unpredictable road conditions, infrequent location updates, and irregular update intervals, accurately predicting mobility is challenging in vehicular networks.

We develop a new mobility prediction algorithm based on the idea of *voting among K nearest trajectories (KNT)* (Section 3). We also implement several state-of-the-art mobility prediction algorithms based on Markov mobility models [42, 35]. Our evaluation in Section 5 shows that *KNT* achieves better prediction accuracy on our dataset.

- *Thorough evaluation through simulation, emulation, and testbed experiments.* We conduct trace-driven simulations to evaluate the performance of VCD using San Francisco taxi [11] and Seattle bus traces [40] (Section 6). Our results show that VCD is capable of downloading 3-6X as much content as no replication, and 2-4X as much content as wireline or vehicular replication alone; mesh replication further helps to improve throughput by up to 22%. The benefit of VCD further increases as the gap between wireless and wireline capacity enlarges and the AP density increases. In addition, we have developed a full-fledged prototype VCD system that supports real video streaming applications running on smartphones and laptops (Section 4, 7 and 8). We deploy our system in two wireless testbeds using 802.11b and 802.11n. Live road tests suggest that our system is capable of providing video streaming to smartphone and laptop clients at a vehicular speed. To further evaluate the performance of VCD at scale, we run the same AP and controller code as in the testbed together with emulated vehicles in the Emulab [20]. Our experiments show the efficiency of our implementation and that Emulab results closely follow the simulation results.

2. OPTIMIZING REPLICATION

In this section, we first present an overview of our system, and then develop wireline, mesh, and vehicular replication.

2.1 Overview

At the beginning of every interval, the controller (shown in Figure 1) collects the inputs required for computing replication strategy. The controller computes the replication strategy during the current interval so that it can maximize user satisfaction during the next interval (Section 2.2). We use user satisfaction in the next interval as the objective since replication in the current interval is often too late to satisfy the traffic demands in the same interval. The controller then informs the APs of the replication strategy through the Internet or cellular network (in case the APs do not have Internet connectivity). We use cellular networks to send control messages as they are small. A vehicle performs the following actions during its contact with an AP:

- Step 1: The vehicle downloads the content according to the optimization results from this section.
- Step 2: After step 1, the vehicle may still have unsatisfied demand (e.g., due to inaccurate prediction or insufficient capacity to replicate all the interesting content). The vehicle then downloads all the content that it is interested in and is also available locally at the AP.
- Step 3: Next, it downloads the remaining content that it is interested in from the AP's mesh network or the wireline network when the AP has wireline connectivity.
- Step 4: Parallel to the Internet download, the vehicle can take advantage of wireless capacity by opportunistically transferring files to and from APs (Section 2.4).

2.2 Optimized Wireline Replication

Problem formulation: Our goal is to find a replication strategy that maximizes user satisfaction subject to the available network capacity. Specifically, we want to determine how to replicate files to APs during the current interval to maximize the amount of useful content that can be downloaded by vehicles when vehicles meet the APs in the next interval. To support delay sensitive applications, only content that are downloaded before the deadline counts and the other content that already misses the deadline will be excluded from consideration for replication. This replication problem involves the following issues: (i) in what form to replicate the content, and (ii) how much to replicate for each file.

Applying network coding: To answer the first question, we note that directly replicating original content introduces two major problems. *First*, it is inefficient for serving multiple vehicles. Suppose multiple vehicles are interested in the same file and have downloaded different portions of the files before their contacts with an AP. If they visit the same AP, in order to satisfy all vehicles we need to replicate the union of the packets they need, which is inefficient. For example, vehicles 1 and 2 are both interested in file 1. Vehicle 1 has downloaded the first half and vehicle 2 has downloaded the second half before they encounter the AP. We need to replicate the complete file to satisfy both vehicles. *Second*, replicating original files is also unreliable. Consider a vehicle is expected to visit three APs but in fact it only visits two of the three APs, which is quite common due to prediction errors. If we just split the file into three and transfer one part to each AP, then the vehicle will not get the complete file. However, if we split the files into two and transfer one part to each AP, the vehicle still may not get the complete file since it may get two redundant pieces (e.g., when it visits the two APs that both have the first half of the file).

We apply network coding to solve both problems. Specifically, we divide the original content into one or multiple files, each containing multiple packets. We use random linear coding to generate random linear combinations of packets within a file. With a sufficiently large finite field, the likelihood of generating linearly independent packets is very high [25]. For a file with n packets, a vehicle can decode it as long as it receives n linearly independent packets for it.

Network coding solves redundancy problems in the multiple-vehicle case since each linearly independent packet adds value. In the above example of two vehicles, we only need to replicate one half worth of file content to satisfy both users, reducing bandwidth consumption by half. It solves reliability issue in the single vehicle case by incorporating redundancy. In the above example, we can split the file of interest into 2 and randomly generate 3 linear combinations of these 2 pieces and replicate one to each AP. Since any two pieces are linearly independent with a high probability, the vehicle can decode the file once it gets any two pieces.

Note that we need network coding (not just source coding) in order to avoid redundancy during replication without fine-grained coordination. That is, APs should re-encode data before they replicate data to vehicles and other APs. For example, AP 1 has a complete file 1, and sends to vehicle 1 half the file, which is relayed to AP2; similarly AP 1 sends half of the file 1 to vehicle 2, which relays it to AP2. In order to

$\triangleright$ *Input* : $Intv, WCap, InCap, OutCap, CT, AP, size, has, Q$
$\triangleright$ *Output* : $x(f,i,a)$ *and* $D(v,f,a)$

Maximize: $\sum_{v}\sum_{f}\sum_{a\in AP(v)} Q(v,f)D(v,f,a) - \gamma\sum_{i\in I}\sum_{a\in A}\sum_{f} x(f,i,a)$

Subject to:

[C1] $\sum_{f} D(v,f,a) \le WCap(a) \times CT(a,v) \quad \forall v, a \in AP(v)$

[C2] $\sum_{a\in AP(v)} D(v,f,a) \le size(f) - has(v,f) \quad \forall v, f$

[C3] $D(v,f,a) \le has(a,f) + \sum_{s\in I} x(f,i,a) \quad \forall v, f, a \in AP(v)$

[C4] $\sum_{i\in I} x(f,i,a) \le size(f) - has(a,f) \quad \forall f, a \in A$

[C5] $\sum_{i\in I}\sum_{f} x(f,i,a) \le InCap(a) \times Intv \quad \forall a \in A$

[C6] $\sum_{a\in A}\sum_{f} x(f,i,a) \le OutCap(i) \times Intv \quad \forall i \in I$

Figure 2: Optimizing wireline replication, where v is a vehicle, f is a file, a is an AP, i is a node with wireline connectivity (which may or may not be an AP, e.g., a Web server), $Intv$ is an interval duration, A is the set of all the APs, I is the set of all the nodes with wireline connectivity, $AP(v)$ is the set of APs that vehicle v will visit, $Q(v,f)$ is the probability that v is interested in file f, $D(v,f,a)$ is the amount of traffic in file f vehicle v should download from AP a during a contact in the next interval, $x(f,n_1,n_2)$ is the amount of traffic in file f to replicate from node n_1 to node n_2 during the current interval, $CT(a,v)$ is average contact time of vehicle v at AP a, $WCap$ is wireless capacity, $InCap$ is incoming wireline access link capacity, $OutCap$ is outgoing wireline access link capacity, $has(n,f)$ is amount of file f a node n has, and $size(f)$ is the size of file f.

avoid replicating duplicates to AP 2, AP 1 should re-encode the data before sending to the vehicles. In Section 4.2, we describe network coding cost and optimization.

Optimizing replication traffic: Using network coding, we transform the original problem of determining which packets to replicate into the problem of determining how much to replicate for each file. To solve the latter problem, we formulate a linear program, as shown in Figure 2. A few explanations follow. The first term in the objective function, $\sum_v \sum_f \sum_{a\in AP(v)} Q(v,f)D(v,f,a)$, quantifies user satisfaction, which is essentially the total traffic downloaded by a vehicle (before the deadline), denoted as $D(v,f,a)$, weighted by the probability for vehicle v to be interested in file f, denoted by $Q(v,f)$. The second term in the objective represents the total amount of wireline replication traffic. We include both terms to reflect the goals to (i) maximize user satisfaction, and (ii) prefer the replication that uses less traffic among the replication strategies that support the same amount of traffic demands. Since the first objective is more important, we use a small weighting factor γ for the second term just for tie breaking (i.e., when the first objective is the same, we prefer the one that has the lowest replication traffic). The value of γ should be small enough to ensure it does not dominate the first term, and our evaluation uses $\gamma = 0.001$. Note that in addition to optimizing the total downloaded traffic, it is also easy to support alternative metrics that are functions of downloaded traffic (e.g., a linear approximation of proportional fairness, which balances between fairness and total downloaded traffic [39]).

Constraint C1 in Figure 2 enforces that the total amount of traffic downloaded from an AP during a contact is bounded by the product of AP's wireless capacity and average contact duration. Constraint C2 ensures that the total content downloaded for each file does not exceed the total file size minus the amount of file the vehicle already has before the download. Constraint C3 encodes the fact that the amount of file the vehicle can download from an AP cannot exceed what AP already has plus what will be replicated to the APs through the wireline network during the current interval. Constraint C4 indicates that the total replication traffic in file f towards an AP is bounded by the file size minus the amount that the AP already has. Constraints C5 and C6 reflect the total replication traffic through the wireline network does not exceed the access link capacity. The formulation can support APs with and without wireline access by setting wireline capacity to zeros for APs without wireline access.

Obtaining input: As shown in Figure 2, we need $Intv$, $WCap$, $InCap$, $OutCap$, CT, AP, $size$, has, and Q. The $Intv$ is a control parameter that determines how frequently the optimization is performed. In our evaluation, we set $Intv$ to be 3 minutes, which gives a good balance between (i) achieving accurate mobility prediction and (ii) limiting the optimization overhead, since both (i) and (ii) decrease as $Intv$ increases. The next three inputs on link capacity—$WCap$, $InCap$, and $OutCap$—are known in advance and change infrequently. CT is estimated using historical data and only needs to be updated infrequently. For ease of estimation, in our evaluation we set $CT(a,v)$ to be the average duration of all contacts from the trace. AP can be obtained by either letting a vehicle run a mobility prediction algorithm locally or have it send several of its recent GPS coordinates to the controller, which will perform mobility prediction. $size$, has, and Q are reported by the vehicles either through a Wi-Fi link during a contact with an AP or via a cellular link during other time. A vehicle predicts what future content to request based on the previous and current requests. For streaming content, it is relatively easy to predict as most users will request the subsequent frames. Demand prediction in general has been a well-researched problem in many domains [36, 6] and we can leverage existing solutions. Note that all the control information is small and can be easily compressed by sending delta from the previous update.

Using optimization results: *To enhance robustness against errors in estimating the inputs, we use $x(f,i,a)$ and $D(v,f,a)$ to control the relative replication and download rates across different files using the weighted round robin scheduling.* For example, if $x(f1,i,a) = 2 * x(f2,i,a)$, file 1 is downloaded twice as fast as file 2. In this way, we can still fully utilize network resources even if contact time or network capacity have estimation errors.

2.3 Optimized Mesh Replication

If some APs along the road are close together, they can form a mesh network. The mesh connectivity indicates that (i) we can now replicate content to the APs using mesh connectivity in addition to wireline connectivity, and (ii) if a vehicle meeting AP1 requests a file that AP1 does not have, it is more efficient to fetch from its mesh network (if there is an AP having the file) than fetching via the slow wireline access link. A neighboring AP in the mesh network can have the file either due to explicit replication or opportunistically caching from earlier interactions.

To support (i), we make the following modifications to the replication formulation in Figure 2. Let $MCap(a',a)$ denote the capacity of a wireless link from AP a' to a in the mesh network, which can be different from the capacity of wireless links between vehicles and APs ($WCap$). Let $z(f,a',a)$ denote the amount of content to replicate from AP a' to a for file f through the mesh network. Let $ETX(a',a)$ denote the average number of transmissions required to send a packet from a' to a through the mesh and can be easily estimated by measuring link loss rate using broadcast probes as in [16]. Our modifications include (1) adding $-\gamma\sum_f \sum_{(a',a)\in mesh} z(f,a',a)$ to the objective function to prefer the replication that uses less wireline and mesh replication traffic among the ones that support the same traffic demands, (2) adding $+\sum_{(a',a)\in\ mesh} z(f,a',a)$ to the right hand-side of [C3] to indicate a node can download from AP a any content that is already available at a or replicated to a through either the wireline or mesh network, (3) adding the following two new constraints: $z(f,a',a) \le has(a',f)$ and $\sum_{f,(a',a)\in\ mesh} \frac{ETX(a',a)z(f,a',a)}{MCap(a',a)} \le 1$. The former constraint ensures AP a' cannot replicate more content than it has. The latter is interference constraint, which enforces that total active time of all mesh nodes cannot exceed 100% assuming all nodes in the mesh network interfere with each other. Note that its left-hand side computes activity time by multiplying the replicated content by the expected number of transmissions normalized by the wireless capacity.

To support (ii), when AP a receiving a request for a file that it does not have locally, it first tries to get from AP a' in the same mesh if the end-to-end throughput (approximated as $MCap(a',a)/ETX(a',a)$) is higher than the wireline access link; only when no such AP is found, does it fetch using the wireline access link.

2.4 Opportunistic Vehicular Replication

In addition to wireline and mesh replication, content can also be replicated using vehicles – a vehicle can carry content from one AP to another as it moves. This new form of replication is more effective than traditional vehicle-to-vehicle (V2V) replication, because V2V needs a very large number of vehicles to be effective whereas even a small number of APs can significantly enhance the performance by leveraging the Internet and mesh connectivity among them [8].

One way to support this new vehicular replication is to augment the LP formulation in Figure 2 with vehicular replication terms, which can produce wireline, mesh and vehicular replication as the final output. However, due to unpredictability in vehicular relay opportunity, we find the effectiveness of such optimization is rather limited. Interestingly, we find the following simple opportunistic vehicular replication scheme is effective.

Since the wireline fetch is bottlenecked by the slow access link, the wireless link is not fully utilized. Therefore, as mentioned in Section 2.1, parallel to the wireline fetch, a vehicle can take advantage of local wireless connectivity to exchange content with the AP. Such exchange has two benefits: (i) the vehicle can upload content to the AP, which can serve other vehicles later, and (ii) the vehicle can download files, which may serve its own demand in the future or the vehicle can relay the content to other APs for future service. To enhance effectiveness, we order the files to upload based on the expected future demand for the file at the AP, which is estimated as $\sum_{v:\ v \text{ visits } a} Q(v,f)demand(v,f)$, where $demand(v,f)$ is the expected size of file f vehicle v is interested in. While this vehicular replication is simple, our evaluation shows that it is highly effective.

3. PREDICTING MOBILITY

If we can predict the AP that a vehicle will visit, we can start replicating the required content to the AP well before the vehicle arrives so that the vehicle can enjoy high wireless bandwidth during its download. Predicting mobility for vehicles is challenging because (i) vehicles often move at high speed, which implies that there can be many possible next states and it is difficult to accurately predict transitions to a large number of next states, (ii) the GPS updates often have relatively low frequency (e.g., once per minute) and tend to arrive at irregular intervals, and (iii) the road and traffic conditions are highly dynamic and difficult to predict.

To address the challenge, we develop a novel mobility prediction algorithm for vehicular networks: *K Nearest Trajectories (KNT)*. We also implement two existing algorithms based on Markov mobility models [42, 35]. In Section 5, we show that *KNT* achieves better accuracy on our dataset.

Algorithm: We observe that the mobility of vehicles exhibits unique structure – a vehicle follows the roads and only makes turns at the street corners or highway exits. This suggests that a good predictor should take into account the speed and direction in the previous interval as well as the underlying road structure. Our *KNT* algorithm is able to account for such information without requiring explicit knowledge about the detailed road map. Given a vehicle v_0 and current time t_0, the algorithm predicts the set of APs visited by v_0 in a future interval $[t_0+\Delta_1, t_0+\Delta_2]$ ($\Delta_2 \ge \Delta_1 \ge 0$) in two steps:

1. *Finding K nearest trajectories.* Our algorithm first finds K existing mobility trajectories in a GPS location database that best match the recent mobility history of the given vehicle. Specifically, we maintain a database of past GPS coordinate updates: $\mathcal{D} = \{(v,t,c)\}$, where v is a vehicle, t is the time for the update, and c is the GPS coordinate. For any vehicle v and current time t, we define its mobility history MH as the set of GPS coordinates reported by v in the past δ seconds: $MH_v^t = \{c | (v,s,c) \in \mathcal{D} \wedge s \in [t-\delta, t]\}$. We also define a distance function between two trajectories: $f(MH_{v_0}^{t_0}, MH_v^t) = \sum_{c\in MH_{v_0}^{t_0}} \min_{d\in MH_v^t} \|c-d\|_2$, where $\|c-d\|_2$ is the Euclidean distance between the two locations specified by GPS coordinates c and d. Essentially, this distance function reflects the total distance from each point on $MH_{v_0}^{t_0}$ to the closest point on MH_v^t. We then find K pairs of (v,t) that minimizes $f(MH_{v_0}^{t_0}, MH_v^t)$, i.e., the K nearest neighbors of (v_0,t_0).
2. *Voting.* For each of K nearest trajectories (v,t), we use linear interpolation (i.e., using a line to connect two adjacent points) to obtain its mobility trajectory in the future interval $[t+\Delta_1, t+\Delta_2]$. Based on this, we obtain the set of APs visited by v during that interval. We then report all the APs that are visited by at least T out of K nearest trajectories as the predicted set of APs that will be visited by v_0 during future interval $[t_0+\Delta_1, t_0+\Delta_2]$.

In step 1 above, to avoid computing $f(MH_{v_0}^{t_0}, MH_v^t)$ for all pairs of trajectories (which is expensive), we only compute for the trajectory pairs that are nearby. To quickly identify the trajectories that are close to the current one, we create an efficient index structure by (i) discretizing the GPS latitude-longitude coordinate space into $0.0001° \times 0.0001°$ grid squares, and (ii) storing all the (v,t) inside each grid square. Given (v_0,t_0), we start from its grid square and use expanded ring search to find C candidate points (v,t) residing in the same or nearby grid squares. We then find K nearest neighbors among these C candidate points.

To be general, our prediction algorithm intentionally does not exploit external knowledge (e.g., certain vehicles have similar trajectory on different days, which may hold for some personal vehicles). When such information is available, our prediction algorithm can potentially incorporate it when finding nearest trajectories to further improve the accuracy.

Parameter setting: Our algorithm has four control parameters: the number of nearest trajectories K, the number of candidate points C, the voting threshold T, and the mobility history duration H. In our evaluation, we keep $C = 32$, vary $T = 1,2$, vary K from 2 to 12, and vary H from 60 to 180 seconds. Our results show that $(K = 4, T = 2, C = 32, H = 60)$ consistently give the best performance. We thus only report the results under this parameter setting.

4. VCD IMPLEMENTATION

We implement VCD in both Emulab [20] and our real testbed with smartphone and laptop clients. VCD consists of a controller, APs, content servers, and clients in vehicles. Emulab and testbed use the same controller, AP, and content server implementation, all of which are implemented as multi-threaded C++/Linux programs. They differ in client implementation. In Emulab, we implement a virtual vehicle program, which can emulate multiple vehicles, allowing us to conduct a trace driven emulation of all the vehicles in our trace using a few virtual vehicles. The client in the real testbed is implemented on both smartphones and laptops, which is described in Section 4.2.

4.1 System Overview

Communication between APs and controller: The APs and controller communicate with each other using TCP. As noted in Section 2.1, at the beginning of every interval the controller collects inputs, computes the replication strategy, and instructs content servers or APs to perform wireline and mesh replication at the desirable rates.

Communication between AP and vehicle: The communication between APs and vehicles uses UDP that sends data at close to the PHY data rate. When a vehicle contacts an AP, it sends a HELLO message that includes (i) a list of files and their sizes that it already has, (ii) the files it is interested in during the current and next intervals. Upon receiving the first HELLO message from the vehicle, the AP initiates data download to the vehicle according to the four steps described in Section 2.1. Meanwhile, the vehicle also sends buffered GPS updates (generated every 20 seconds in the testbed and every 1 minute in Emulab). In step 4, the AP determines a list of files for the vehicle to upload sorted in increasing utility as described in Section 2.4. The AP sends this list in a REQ message. Upon receiving the first REQ message, the vehicle initiates data upload to the AP. Both HELLO and REQ messages use soft state and are sent periodically once every control interval (100ms in testbed and 1s in Emulab). These messages also serve as heartbeats to the other party. To achieve efficiency and reliability for data traffic, an AP applies network coding before sending the data it receives. In addition, we use multiple content servers and leverage a central dispatcher to distribute requests to an appropriate content server for load balancing.

Batch size	110 packets		70 packets		35 packets	
Device	Phone	Desktop	Phone	Desktop	Phone	Desktop
Encoding	12.19s	0.0228s	4.79s	0.0088s	1.18s	0.0021s
Decoding	8.22s	0.017s	3.27s	0.0067s	0.809s	0.0012s

Table 1: Network coding benchmarks

4.2 Client Implementation

We implemented client on both Windows XP laptops and smartphones. We use HP Ipaq 910 Business Manager smartphones with Windows Mobile 6.1 Professional operating system, Marvell PXA270 416 MHz Processor, 128MB RAM, Marvell SDIO8661 802.11 b/g Wi-Fi card, and the .Net Compact Framework. Our implementation on smartphones uses OpenNet API, and that on Windows uses Managed Wi-Fi API. Implementing on smartphones introduces several challenges: (i) limited APIs and often inconsistent implementations, (ii) expensive I/O, (iii) limited system resources, and (iv) many existing wireless optimizations cannot be implemented due to lack of low level access, which we address.

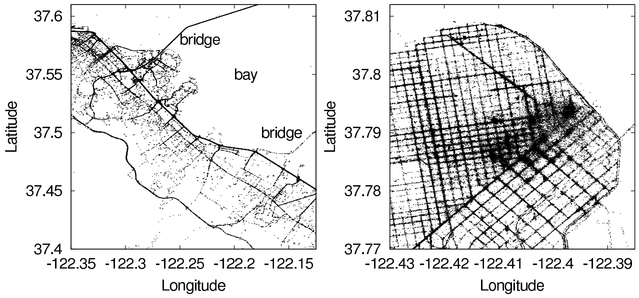
Handling expensive I/O: Since I/O on smartphones is around an order of magnitude slower than desktops, packets cannot be stored on the disk and read back on-demand for vehicular replication. For simplicity, we use an in-memory packet buffer with FIFO replacement policy. We further limit disk access during the contact with APs and push data to the disk only after the contact is over so that we can fully utilize the short contact time for data transfer.

Handling network coding cost: Due to the slow processor, thread scheduling and dynamic assignment of priorities are important. For example, network coding incurs much higher cost on the smartphone than on the desktop as shown in Table 1. We use packet size of 1230 bytes (i.e., the packet payload in our testbed implementation to ensure the maximum packet size is still within 1500 bytes (Ethernet MTU)). Our evaluation uses file sizes of 35, 70, and 110 packets, which correspond to minimum, median and maximum file sizes used in our experiments. To minimize the impact of decoding, we schedule the decoding thread at a low priority during a contact and increase its priority after the contact.

Connection setup: The ability to quickly establish connection to an AP is crucial. [10, 24] examine this problem in greater detail. In the context of smartphones, the problem becomes even harder since NDIS does not provide access to many low level parameters to implement the association optimizations proposed in the literature. Windows Mobile provides two ways to initiate connection to a Wi-Fi network programmatically, either through the wireless zero config (WZC) interface or by setting the appropriate NDIS OIDs. The association times using the WZC interfaces were around 3.0 sec, which is unacceptable in the vehicular network context. We therefore disable WZC and implement NDIS based association, which yields significantly lower association times. We also implement our own DHCP client and use the DHCP caching mechanism described in [10].

Our connection setup procedure is as follows. The smartphone scans for APs every 100 ms. When an AP is discovered, the smartphone waits for 3 RSSI readings greater than -91dB before trying to associate. We do not associate immediately because an association failure is expensive. The association procedure is retried up to 7 times with a short delay of 50ms between consecutive attempts. The various threshold values used in the scheme were chosen empirically. We report the association time and failures in Section 8.

5. MOBILITY PREDICTION ACCURACY



(a) Taxi locations (on highways) (b) Taxi locations (inside city)

Figure 3: Illustration of traces for mobility prediction.

Mobility traces: We obtain real vehicular mobility traces from Cabspotting [11] and Seattle [40]. The former contain over 10 million GPS longitude and latitude coordinates for approximately 500 taxis in the San Francisco Bay Area over the course of 30 days (December 13, 2008 – January 13, 2009). The latter contains several week-long traces of city buses in Seattle during 2001. The bus system consisted of over 1200 vehicles covering a 5100 square kilometer area. The GPS coordinates are updated approximately once per minute for both Cabspotting and Seattle traces. Figure 3 (a) and (b) illustrate the vehicle locations along the highway and inside San Francisco. One can clearly observe the underlying street structure from taxis' GPS. Similar pattern was observed in Seattle traces.

AP locations: We consider two sets of locations for placing APs: (i) gas stations and (ii) coffee shops. We use Yahoo's Local Search API (version 3) [49] to obtain the longitude and latitude coordinates of 1120 gas stations and 1620 coffee shops in San Francisco Bay Area, as well as 618 gas stations and 738 coffee shops in Seattle. The average distance between two closest APs in the traces ranges between $345-589$ *m* and the median distance is $157-433$ *m*. There are quite a few APs whose distance exceeds 3500 *m* in all the four traces. The communication range between an AP and a vehicle is set to either 100 or 200 meters. We use these values because they approximate the communication ranges we measured from our vehicular testbeds using 802.11b and 802.11g, respectively. To determine the contact period between a vehicle and an AP, we use linear interpolation to obtain the vehicle's mobility trajectory between two adjacent GPS location updates.

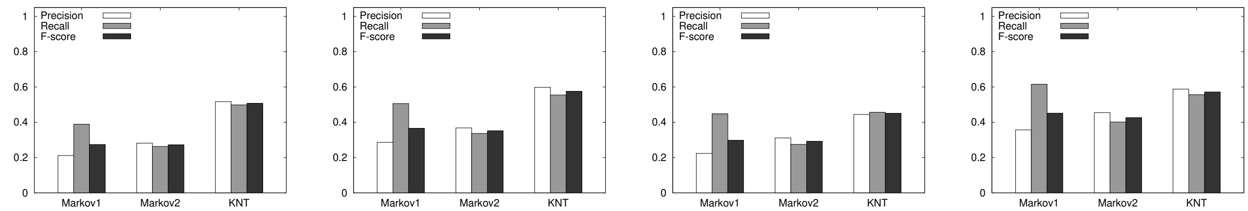
Trace statistics: We analyze the traces and find that 23% – 40% of time the vehicles were parked or moved within 1 mile/hour, 70% of time they moved less than 11 – 15 miles/hour, and 90% of time they moved less than 25-27 miles/hour. Since most of the cabs are in the downtown area, they are bounded by the speed limits of the downtown area. We further study the contact duration and observe 70% of the contacts between a vehicle and an AP last less than 39-51 seconds when the communication range is 100 meters, and less than 54-82 seconds when the range increases to 200 meters. Such short contacts highlight the importance of replicating data in advance.

Baseline algorithms: For baseline comparison, we implement a variant of the mobility prediction algorithm in [35]. The algorithm is based on a second-order Markov mobility model. Each state has two sets of coordinates: the vehicle's location at time τ ago, and its current location. In our evaluation, τ is either 1 or 2 or 3 minutes. We deal with irregular GPS update intervals through linear interpolation. To avoid state space explosion, the algorithm discretizes the longitude and latitude coordinates into $0.001^\circ \times 0.001^\circ$ grid squares. The algorithm uses past mobility traces to learn the probability for a vehicle to transition into any new grid square given its last and current grid squares. Based on the transition probabilities, the algorithm identifies the grid square that the vehicle is most likely to visit next, and uses the center of this grid square as the predicted new location for vehicle after time τ. This procedure is repeated to make predictions further into the future. Based on the predicted locations, the algorithm applies linear interpolation to obtain the entire mobility trajectory and then computes the set of APs the vehicle is predicted to visit during a future interval. As in [35, 42], the algorithm falls back to a first-order Markov model when the second-order Markov model fails to make a prediction. Finally, we also implement the first-order Markov model as another baseline algorithm.

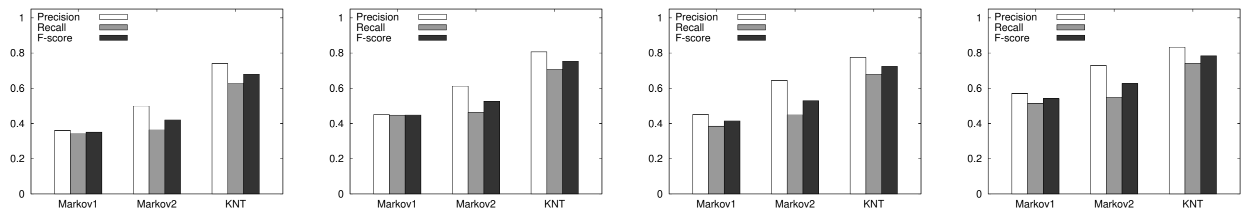
Metrics: We quantify the prediction accuracy using two metrics: (i) *precision*, i.e., the fraction of APs predicted by our algorithms are indeed visited by the vehicles in a future interval, and (ii) *recall*, i.e., the fraction of APs visited by the vehicles in a future interval are correctly predicted by our algorithms. In addition, we integrate precision and recall into a single metric called *F-score* [47], which is the harmonic mean of precision and recall: $\textit{F-score} = \frac{2}{1/\textit{precision}+1/\textit{recall}}$. For all three metrics, larger values indicate higher accuracy.

Evaluation results: We consider the following prediction scenario as required by our replication optimization algorithm: *per-interval prediction*, which divides time into fixed intervals and the goal is to predict the set of APs that will be visited by a vehicle in the next interval. The prediction interval is set to 3 minutes, which matches the interval for periodic replication optimization. For each prediction algorithm we evaluate, we consider multiple parameter configurations and choose the configuration that yields the best *F-score*. The results from Cabspotting traces use seven days of training data to predict the mobility on the eighth day, and results from Seattle bus traces use 5 days of training data to predict the sixth day as these traces have shorter duration.

Figure 4 shows the prediction accuracy when APs are placed at either gas stations or coffee shops and the communication range is either 100*m* or 200*m*. For the San Francisco taxi mobility trace (Figure 4 (a)–(d)), the *F-scores* of our algorithm (*KNT*) are 25-85% higher than those of the first-order Markov model (*Markov1*) and second-order Markov model (*Markov2*). For the Seattle bus mobility trace (Figure 4 (e)–(h)), *KNT* outperforms *Markov1* and *Markov2* by 25–94% in terms of *F-scores*. In general, the absolute prediction accuracy for all three algorithms is higher for the bus mobility trace, because buses tend to follow fixed routes and are thus more predictable.



(a) San Francisco, gas, range=100m (b) San Francisco, gas, range=200m (c) San Francisco, coffee, range=100m (d) San Francisco, coffee, range=200m



(e) Seattle, gas, range=100m (f) Seattle, gas, range=200m (g) Seattle, coffee, range=100m (h) Seattle, coffee, range=200m

Figure 4: Accuracy comparison of different mobility prediction algorithms.

Finally, it is worth noting that in contrast to findings in [35, 42], *Markov2* does not significantly outperform *Markov1* in our evaluation. This suggests that with higher speed and less frequent GPS location updates, mobility prediction is more challenging in vehicular networks. As a result, solutions that perform better in less mobile environment do not necessarily perform better in vehicular networks.

Summary: The above results clearly show that our *KNT* mobility prediction algorithm consistently achieves good accuracy in vehicular networks. Later in Section 6, we further show that optimization based on our prediction results yields good performance in practice.

6. TRACE-DRIVEN SIMULATION

6.1 Simulation Methodology

We develop a trace-driven simulator for evaluation as follows. We first generate the contact traces based on the mobility traces, AP locations, and wireless communication range. When multiple vehicles contact an AP at the same time, we divide the original contacts into non-overlapping contacts, each of which has only one vehicle in contact with an AP. Such contact partitions can be easily realized in practice by letting the AP serve the new vehicle only after it finishes serving the previous one. Similarly, when a vehicle is within the communication range of multiple APs, we also partition the contact into multiple non-overlapping intervals, each of which involves one AP. Another way to partition a contact between multiple vehicles and an AP or between multiple APs and a vehicle is to equally divide the contact time among multiple vehicles or multiple APs that are involved in the contact to mimic round-robin scheduling. The performance of these two types of partitions is similar, and we use the first partition in our evaluation.

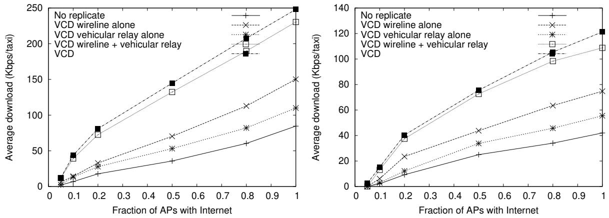
We then feed the actual contact traces (after the above post processing), predicted contacts, and traffic demands to the simulator. The simulator updates the content at APs and vehicles based on the actual contacts, traffic demands, replication schemes, and wireless and wireline capacity at APs. We implement network coding for all data transfer to ensure only innovative packets (i.e., whose coding coefficients are linearly independent) are exchanged between APs and vehicles or among APs. We have a content server on the Internet, which has all the content, whereas all APs and vehicles are initialized with no content.

We compare (i) no replication, (ii) wireline replication alone, (iii) vehicular replication alone, (iv) both wireline and vehicular replication, (v) wireline, vehicular, and mesh replication (VCD). In all the schemes, a vehicle downloads content remotely from the Internet whenever the AP has Internet connectivity and the content is not available locally at the AP or mesh network.

To study the impact of traffic demands, we generate traffic demands following either uniform or Zipf-like distribution. In both cases, for every interval, a vehicle randomly selects a specified number of files to request. In the uniform distribution, a file is uniformly drawn from the pool of the files that the vehicle has not requested previously. In Zipf-like distribution, the probability of requesting the ith file is proportional to $\frac{1}{i^{\alpha}}$, where i is the popularity ranking of the file and $i = 1$ indicates the most popular file. We set $\alpha = 0.4$ so that we can generate similar traffic load using both Zipf-like and uniform distributions and the performance difference is solely due to the difference in the distribution.

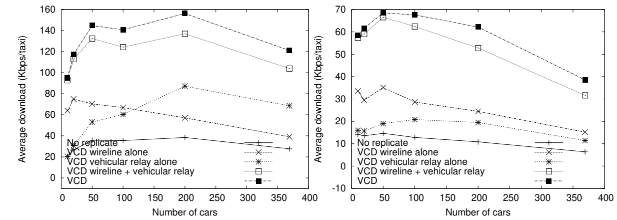
For delay sensitive applications, such as video, their performance depends on the amount of data received before the deadline. Therefore, we use average throughput per vehicle as our performance metric, which denotes the total demand that is satisfied before the deadline divided by the product of the number of vehicles and the entire trace duration (including the time without contacts with APs). The deadline is set to the end of the interval in which the demand is generated.

Our evaluation uses 2-hour trace, which exhibits similar contact characteristics as in the 1-day trace, shown in Section 5. Other default settings used in our evaluation include: 100-meter communication range between APs and vehicles, 500-meter communication range among APs (well within reach by many mesh routers [4, 32]), Zipf-like traffic demands, placing APs at coffee shops, all APs having 22 Mbps wireless link, half of the APs having Internet links with 2Mbps while the other half have no Internet connection. The content server has a 1 Gbps Internet link and zero wireless capacity to indicate that it is not directly reachable by vehicles. There are 1200 files in total. Each user requests 20 files every 3-minute interval, each file has 2K packets,



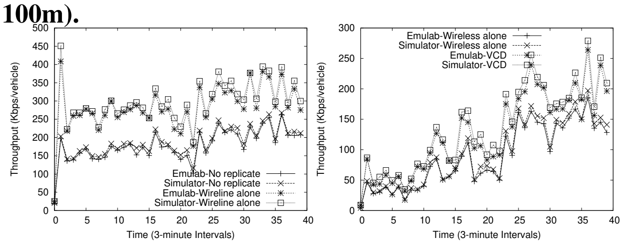
(a) San Francisco (b) Seattle

Figure 6: Average throughput under varying fraction of APs with Internet (Zipf-like traffic, APs at coffee shops, range=100, 50 vehicles).



(a) coffee shops (b) gas stations

Figure 7: Average throughput under a varying number of vehicles (San Francisco, Zipf-like traffic, range = 100m).



(a) All APs have Internet (b) 10% APs have Internet

Figure 9: Cross validation: comparing performance in Emulab and simulation

which contains 1000 bytes. Every file represents either a video clip or one chunk in a larger video file (e.g., We divide a large video file into smaller chunks and generate random linear combinations of packets within each chunk for efficient replication). We further evaluate the effects of changing these parameters.

6.2 Simulation Results

Varying wireless bandwidth: In Figure 5, we plot the total downloaded content as we vary wireless bandwidth from 5, 11, 22, 54, 120, and 150 Mbps. We make the following observations. First, in all cases VCD significantly out-performs the other schemes and its benefit increases rapidly with wireless capacity. Second, as we would expect, no replication performs the worst. Interestingly, its performance remains the same as we increase wireless capacity. This is because without replication APs often do not have content locally and the wireless download is bottlenecked by slow Internet access capacity. This further demonstrates the need of replication. Third, the performance of both wireline and vehicular replication alone initially improves with increasing wireless capacity and then tapers off. This is because limited Internet capacity prevents fully taking advantage of large wireless capacity. In comparison, harnessing both wireline and vehicular replication opportunities can effectively utilize the large wireless capacity when available. Adding mesh replication further increases average throughput by 14-20% under high AP density (Figure 5(c)), and by 3-13% in low AP density. The benefit of mesh replication can be increased further if APs use high gain antennas or MIMO. Overall, at 22Mbps Wi-Fi capacity, VCD achieves 70 – 300 Kbps average throughput per vehicle depending on the AP density, which can support video streaming applications.

Varying fraction of APs with Internet connectivity: Next we vary the fraction of APs with Internet connectivity. Figure 6(a) and (b) plot the average downloaded traffic in San Francisco and Seattle traces, respectively. As we can see, VCD continues to significantly out-perform the other schemes. In addition, the benefits of all types of replication increase with the fraction of APs that have Internet connectivity. The rate of such increase is faster for the replication schemes that involve wireline replication, since they explicitly take advantage of the new wireline capacity to push data.

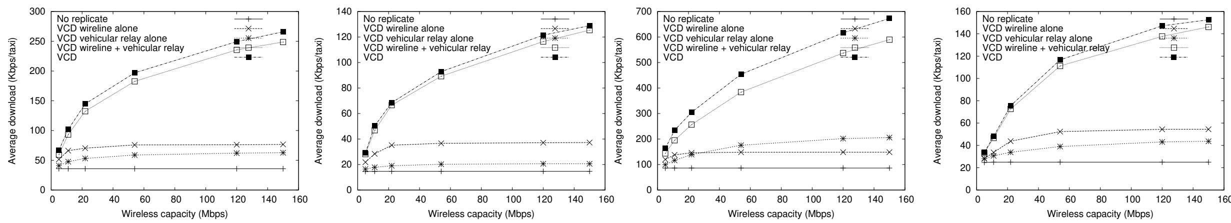
Varying number of vehicles: To further evaluate the impact of degree of deployment, we vary the number of vehicles by randomly selecting a subset of vehicles from the traces. Figure 7 summarizes the performance results. We make the following observations. First, VCD continues to perform the best in all cases. Second, increasing the number of vehicles initially improves the average throughput because more content are available locally at APs due to previous requests coming from other users. In addition, increasing the number of vehicles also creates more wireless relay opportunities. However, a further increase degrades performance due to increased contention for limited wireline and wireless resources. Third, the benefit of mesh replication increases with the number of vehicles. When we use all the vehicles in the two-hour traces, we find that the mesh replication helps to increase throughput by 17-22%. This is because increasing the number of vehicles increases vehicular relay opportunities and makes it more likely to have content available at nearby mesh nodes.

Varying traffic demands: Figure 8 shows the performance for uniformly and Zipf-like distributed traffic demand, respectively. As before, VCD performs the best in all cases. The performance of uniform and Zipf-like distributed traffic receives similar performance. Moreover, decreasing the total number of files tends to improve performance as demands are more concentrated and less replication is required to satisfy them. Finally, the replication benefit tends to increase with an increasing number of files requested by each user. This is because when a user is interested in more content, it is more likely to have some locally available content that satisfies the user.

7. TRACE-DRIVEN EMULATION

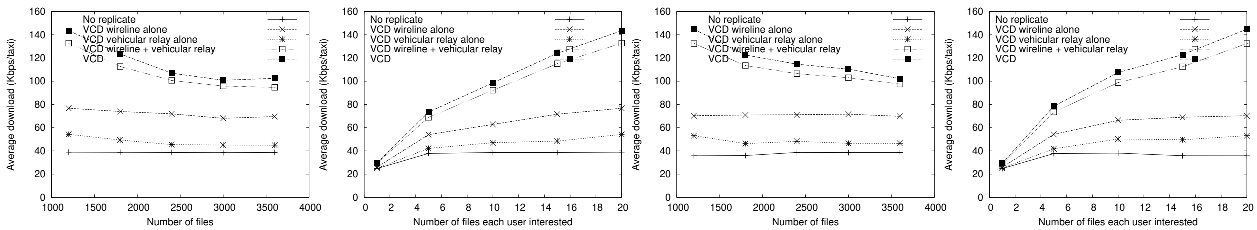
The goal of our Emulab implementation is twofold: (1) validate simulation results, and (2) evaluate the performance of VCD at scale, which is hard to do in testbed experiments.

Validation: To validate the simulation results, we compare them against those obtained from Emulab under identical settings. We consider the 30 most interactive APs from the trace contacting 100 vehicles. The radio range is 200m. Given limited machine availability on Emulab, we emulate multiple APs and vehicles on each machine. This limits the link capacity we can select per AP or per vehicle. Hence, our evaluation uses 1Mbps and 6Mbps as the Internet and



(a) San Francisco, coffee shops, range=100m (b) San Francisco, **gas station**, range=100m (c) San Francisco, coffee shops, **range=200m** (d) **Seattle**, coffee shops, range=100m

Figure 5: Average throughput of 50 vehicles under varying wireless capacity and Zipf-like traffic demands. The difference from the base configuration is in bold.



(a) Uniform: vary # files (b) Uniform: vary # files per user (c) Zipf-like: vary # files (d) Zipf-like: vary # files per user

Figure 8: Average throughput under varying traffic demands (San Francisco, vehicle=50, range=100m, coffee shops).

Packet type	Avg KB	% of total traffic
Controller to APs	192	0.006
APs to controller	1483	0.048
Content server to AP data	3078200	99.946
Vehicles to APs	49122	1.599
APs to vehicles data	3023100	98.401

Table 2: Average control message overhead per interval.

wireless link capacities, respectively.

Figure 9 shows the average throughput for each interval in Emulab and simulator. In Figure 9(a), we consider that all APs have Internet connectivity and compare the simulation and emulation performance under no replication and wireline replication alone. We observe that the simulation results closely follow that of Emulab and the discrepancy between them is below 10%. Next we consider only 10% of the APs have Internet connectivity and compare the performance for vehicular replication alone and VCD in both simulator and Emulab. In this case, since most APs are not connected to the Internet and there is no mesh connectivity, most content is replicated via vehicles. Figure 9 (b) shows that the simulation results match well with Emulab results: within 10% difference for both vehicular replication and VCD.

Micro-benchmarks: The following micro-benchmark results show that our implementation is efficient and lightweight even when operating at scale. We emulate the 120 most interactive APs and 317 vehicles from the trace.

Table 2 shows the per-interval control message overhead. We observe that control messages constitute only 0.054% of the total wireline traffic exchanged amongst APs and between APs and the controller, and constitute only 1.6% of the total wireless traffic between APs and vehicles.

Next we evaluate the efficiency of the controller. On a 2.133GHz Xeon machine with 3GB RAM, average CPU and memory utilization of the controller is 2% and 38 MB respectively. The average latency at the controller is 7.8s, which is a small fraction of the 3-minute interval. Out of 7.8s, the LP computation takes 6.5s. It is performed on Emulab using *lp_solve* [30] due to licensing issues with *cplex* [17], and the time can be further reduced if cplex is used instead.

Finally we evaluate the scalability of APs by running 120 instances of the AP on 2.133GHz Xeon machines with 3GB RAM. We find that all APs have roughly the same usage, with each AP instance consuming only 0.01% CPU load and 33 MB of memory. Therefore it is light-weight.

8. TESTBED EXPERIMENTS

We evaluate our approach using two testbeds to understand its feasibility and effectiveness under realistic wireless conditions. The first testbed consists of 9 APs deployed in office buildings near the road. The APs are Linux desktops equipped with 802.11b radios, which are set to a fixed data rate of 11Mbps. The second testbed consists of 4 APs deployed outdoor equipped with 802.11n radios that use auto-rate. 802.11n radios use 2.4GHz frequency with a 20MHz band. In both testbeds, the APs have 1Mbps wireline access link connecting to the back-end content server. In the 802.11b testbed, 3 out of the 9 APs forms a mesh network as a 2-hop linear chain, whereas the 4 APs in the 802.11n testbed forms a mesh network with pairwise connectivity. In both testbeds, mesh communication takes place using additional 802.11b radios. We implement clients on both Windows Mobile Smartphones and Windows XP Laptops. Smartphone clients are used in 802.11b experiments and laptop clients are used in 802.11n experiments. Both clients ran a video streaming application during the car ride. The cars travelled around the testbed at 15 mph (speed limit). We expect that the driving speed does not significantly affect the performance when association time is small, because increasing speed reduces both on-time (i.e., contact time) and off-time (i.e., the time between two consecutive contacts).

Connection setup: Due to deployment constraints, the placement of our 802.11b APs is not ideal: 4 of our APs were placed on the 3rd floor of buildings, limiting their range; and 3 APs were placed in high AP density areas, with 50-70 APs within their range, causing heavy interference. This deployment stress-tests our system. In our experiments during car rides, we were able to associate successfully for 65.2% of all attempts. Most of the failures came from the 3 APs deployed

	Download (kB)	Play time (sec)
No replication	29297	3662
Wireline	71930	8991
Wireline + Mesh	79440	9930
Full replication	92493	11562

Table 3: Throughput of wireline and mesh replication in the 802.11b testbed

	Download (kB)	Play time (sec)
No replication	16857	2107
Wireline	123175	15387
Wireline + Mesh	130827	16353
Full replication	136479	17060

Table 4: Throughput of wireline and mesh replication in the 802.11n testbed

in the high AP density area: association success percentage was only 33.3% for these APs. In fact, even the Windows Mobile Wi-Fi manager utility experienced problems such as very long connection time and adapter freezing near these APs even without any movement. The other access points can successfully associate for 85.7% of the time. The association time in our experiments has minimum, median and maximum of 36ms, 844ms, and 14867ms, respectively. 70% of the associations finish within 2 seconds. We retry association up to 7 times and the median retry count is 1.

In our 802.11n outdoor testbed, association success rate was 89.58% out of 48 attempts. The minimum, median and maximum association times were 48 ms, 162 ms, and 4086 ms, respectively. 80% of the associations finish within 246 ms and the median retry count was 1. The better results for 802.11n testbed were because (i) we used laptops as clients, (ii) APs were placed outdoor closer to vehicles, and (iii) MIMO in 802.11n improves received signal strength.

Wireline and mesh replication: We implemented a video streaming application that can play H.264 videos (downloaded from APs) encoded at 64Kbps. We divide every video into multiple files and use network coding to generate random linear combination of packets within a file. Once enough packets are received for the file, the file is decoded and passed to the video player on the smartphone/laptop to play in proper order using the Windows Mobile media player plugin.

Tables 3 and 4 compare the performance of our optimized wireline and mesh replication with no replication and full replication at all the APs in 802.11b and 802.11n testbeds, respectively. We consider two performance metrics: total download size and total amount of time the video can play (which is proportional to the download size). We report the averages over 3 runs. The full replication assumes every AP has all the files and serves as an upper bound. In both experiments, we follow the planned trajectory, which was fed as input to the controller. In 802.11b testbed, wireline replication alone and wireline plus mesh replication performs 2.45x and 2.7x that of no replication, respectively. In 802.11n testbed, the throughput of wireline and wireline plus mesh replication is 7.3x and 7.8x that of no replication, respectively. This demonstrates the effectiveness of replication. Moreover, the benefit increases with wireless capacity. There is a gap between the performance of VCD and full replication, since the Internet bottleneck prevents complete replication of all the required files.

Vehicular replication: To show the benefit of vehicular replication, we use the following setup. Car 1 follows the route $AP1 - AP2$, and Car 2 follows the route $AP2 - AP1$. Car 1 possesses files 1-20 and is interested in files 21-40, while car 2 has files 21-40 and is interested in files 1-20. Both $AP1$ and $AP2$ lack Internet and mesh connectivity. Therefore, without vehicular replication, neither car can get the content it is interested in and the total throughput is 0 under no replication, wireline replication alone, and mesh replication alone.

	No replication		Wireless replication	
	Car 1	Car 2	Car 1	Car 2
AP1	0	0	Upload 780 pkts	Download 780 pkts,20 files
AP2	0	0	Download 1159 pkts, 20 files	Upload 1159 pkts

Table 5: Comparison between performance with and without vehicular replication.

In comparison, VCD exploits the vehicular replication opportunity. When car 1 meets $AP1$, VCD finds that files 1-20 have highest utility because it predicts car 2 will visit $AP1$ soon and need these files. So $AP1$ instructs the car to upload them first. Similarly, car 2 uploads file 21-40 at $AP2$. When car 1 reaches $AP2$ it can download these files. Similarly, car 2 can download files 1-20 from $AP1$, leading to much higher throughput. Table 5 shows that both cars download their interested files in the actual road experiments.

9. RELATED WORK

We classify related works into three areas: (i) vehicular networks, (ii) disruption tolerant networks (DTNs), and (iii) mobility and demand prediction.

Vehicular networks: A variety of novel techniques have been proposed to optimize various aspects of communications in vehicular networks. One class of works focuses on techniques for optimizing one-hop communication between a vehicle and nearby APs. For example, CarTel project [14] proposes architectures for vehicular sensor networks, and develops a series of techniques to optimize association, scanning, data transport protocols, and rate selection. ViFi [7] proposes to take advantage of multiple nearby APs to improve communication with passing vehicles. [12] conducts in-depth study of various rate adaptation schemes in vehicular networks and proposes to select data rate based on a combination of RSSI and channel coherence time. [34] uses directional antennas to maximize the transfer opportunity between the vehicle and the AP. These works are complementary to our work, which focuses on end-to-end performance of content distribution. We can potentially leverage these approaches to improve the performance of the last hop. With these enhancements, the gap between Internet and wireless capacity will further increase and make replication even more important. Another class of works consider changes to applications to support vehicular networks. For example, Thedu [6] transforms interactive Web search into one-shot request and response process to reduce access delay. While Thedu still requires connecting with the remote server, we replicate content to APs to eliminate the Internet bottleneck. The third class of work studies protocol issues. [19] proposes fast connection establishment, scripted handoffs, and prefetching at APs using HTTP range requests. Finally, there are a few works on vehicle-to-vehicle communication. For example, SPAWN [18] uses gossip for file transfer and CarTorrent [27] extends SPAWN and is implemented in a testbed. [15] treats vehicular networks as a special type of DTNs and focuses on leveraging vehicle to vehicle (V2V) communication to deliver content. As mentioned earlier, inspired by the analysis in [8], we leverage APs as the rendezvous points for replicating content among vehicles. We focus on optimizing content replication given limited wireline and wireless resources, which has not been studied earlier.

Disruption tolerant networks: Vehicular networks can be considered as a special type of disruption tolerant networks (DTNs) and benefit from advances in this area. Different from traditional DTNs, which focuses on communicating with a specific node, we focus on content delivery. Epidemic routing [45] was initially proposed for DTNs, where any two nodes exchange messages whenever they meet. Recently, utility-based replication was proposed, where nodes replicate data over the best contacts according to some utility (e.g., mobility history [26] or delay [5]). We leverage both utility-based optimization for wireline/mesh replication and target wireless replication to maximize effectiveness.

Mobility and demand prediction: There is a large body of literature on mobility prediction, ranging from coarse-grained prediction in cellular networks (e.g., [1, 2, 28, 29, 37]) to more fine-grained prediction in Wi-Fi networks (e.g., [35, 41]). In particular, [42] compares various predictors in literature and suggests that 2nd order Markov with a simple fallback mechanism (when there is no prediction) performs well. [22] builds mobility profiles for users and statistically predicts the next social hub the user will visit. [35] builds the user's customized mobility models on the devices themselves, and uses a second order Markov model to predict the connection opportunity and its quality of the device with an AP. [31] uses the past history to identify opportunities for media sharing in ad hoc DTNs. These works focus on low speed (e.g., personal mobility). Vehicles travel much faster and make mobility prediction more challenging.

In this paper, we do not study demand prediction, since it is a well-researched topic (e.g., [36, 6]). We can leverage the existing work to enhance the effectiveness of VCD.

10. CONCLUSION

We present the VCD system that provides high-bandwidth content access to vehicular passengers by utilizing opportunistic connections to Wi-Fi access points along the road. VCD predicts which APs a vehicle will encounter in the future and proactively pushes content to these APs by leveraging both wireline and wireless connectivity. Using trace-driven simulation and Emulab-based emulation, we show that VCD is capable of downloading 3-6X as much content as no replication and 2-4X as much content as wireline or vehicular replication alone. The benefit further increases as the ratio between wireless and wireline capacity increases. We further develop a full-fledged prototype of VCD using two testbeds: a 9-AP 802.11b testbed and a 4-AP 802.11n testbed. Our experience suggests that VCD is an effective approach for vehicular content distribution.

Acknowledgments: This research is supported in part by NSF Grants CNS-0916106 and CNS-0546755. We are grateful to Marcelo Dias de Amorim and anonymous reviewers for their valuable comments.

11. REFERENCES

[1] I. F. Akyildiz and W. Wang. The predictive user mobility profile framework for wireless multimedia networks. *IEEE/ACM Trans. Netw.*, 12(6), 2004.
[2] A. R. Aljadhai and T. Znati. Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications*, 19(10):1915–1930, 2001.
[3] AT&T DSL. http://www.att.com/gen/general?pid=6431.
[4] AWE Mesh Router. http://www.nomadio.net/AWE_overview.pdf.
[5] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *Proc. of SIGCOMM*, 2007.
[6] A. Balasubramanian, B. Levine, and A. Venkataramani. Enhancing interactive web applications in hybrid networks. In *Proc. of MobiCom*, Sept. 2008.
[7] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. Levine, and J. Zahorjan. Interactive WiFi connectivity for moving vehicles. In *Proc. of SIGCOMM*, 2008.
[8] N. Banerjee, M. Corner, D. Towsley, and B. Levine. Relays, base stations, and meshes: Enhancing mobile networks with infrastructure. In *Proc. of MobiCom*, Sept. 2008.
[9] BMW car2car communication development. http://www.motorauthority.com/bmw–enlists-in-car-2-car-communications-development.html.
[10] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. A measurement study of vehicular Internet access using in situ Wi-Fi networks. In *Proc. of MobiCom*, 2006.
[11] Cabspotting. http://www.cabspotting.com.
[12] J. Camp and E. Knightly. Modulation rate adaptation in urban and vehicular environments: Cross-layer implementation and experimental evaluation. In *Proc. of MobiCom*, Sept. 2008.
[13] Car2Car communication consortium. http://www.car-to-car.org.
[14] Cartel. http://cartel.csail.mit.edu/doku.php.
[15] B. B. Chen and M. C. Chan. MobTorrent: a framework for mobile internet access from vehicles. In *Proc. of IEEE INFOCOM*, 2009.
[16] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, Sept. 2003.
[17] CPLEX. http://www.ilog.com/products/cplex/.
[18] S. Das, A. Nandan, and G. Pau. Spawn: a swarming protocol for vehicular ad-hoc wireless networks. In *Proc. of VANET*, 2004.
[19] P. Deshpande, A. Kashyap, C. Sung, and S. Das. Predictive methods for improved vehicular wifi access. In *Proc. of ACM MobiSys*, 2009.
[20] Emulab. http://www.emulab.net.
[21] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular content delivery using WiFi. In *Proc. of MobiCom*, Sept. 2008.
[22] J. Ghosh, M. J. Beal, H. Q. Ngo, and C. Qiao. On profiling mobility and predicting locations of wireless users. In *Proc. of REALMAN*, 2006.
[23] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. In *Proc. of INFOCOM*, Apr. 2001.
[24] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular opportunistic communication under the microscope. In *Proc. of MobiSys*, 2007.
[25] T. Ho, M. Medard, J. Shi, M. Eros, and D. R. Karger. On randomized network coding, Oct. 06 2003.
[26] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. of ACM ASPLOS*, Oct. 2002.
[27] K. C. Lee, S.-H. Lee, R. Cheung, U. Lee, and M. Gerla. First Experience with CarTorrent in a Real Vehicular Ad Hoc Network Testbed. In *MOVE*, 2007.
[28] G. Liu and G. Maguire, Jr. A class of mobile motion prediction algorithms for wireless mobile computing and communication. *Mob. Netw. Appl.*, 1(2), 1996.
[29] T. Liu, P. Bahl, S. Member, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks, Mar. 1998.
[30] LP_SOLVE: Linear programming code. http://www.cs.sunysb.edu/~algorith/implement/lpsolve/implement.shtml.
[31] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *Proc. of MobiCom*, pages 58–69, 2008.
[32] Meraki MR58. http://meraki.com/products_services/access_points/MR58/.
[33] Mobile Broadband Review 2010. http://mobile-broadband-services-review.toptenreviews.com/.
[34] V. Navda, A. P. Subramanian, K. Dhanasekaran, A. Timm-Giel, and S. Das. Mobisteer: using steerable beam directional antenna for vehicular network access. In *Proc. of MobiSys*, 2007.
[35] A. J. Nicholson and B. D. Noble. Breadcrumbs: Forecasting mobile connectivity. In *Proc. of MobiCom*, Sept. 2008.
[36] V. N. Padmanabhan and J. C. Mogul. Using predictive prefetching to improve world wide web latency. *SIGCOMM Comput. Commun. Rev.*, 26(3), 1996.
[37] P. N. Pathirana, A. V. Savkin, and S. Jha. Mobility modeling and trajectory prediction for cellular networks with mobile base stations. In *MobiHoc*, 2003.
[38] Laptop & smartphone users prefer Wi-Fi to 3G; willing to pay for citywide Wi-Fi. http://www.teleclick.ca/2009/02/laptop-willing-to-pay-for-citywide-wi-fi/.
[39] B. Randunovi and J. Y. L. Boudec. Rate performance objectives of multihop wireless networks. In *Proc. of INFOCOM*, Apr. 2004.
[40] Seattle Bus Traces. http://crawdad.cs.dartmouth.edu/meta.php?name=rice/ad_hoc_city.
[41] L. Song, U. Deshpande, U. C. Kozat, D. Kotz, and R. Jain. Predictability of WLAN mobility and its effects on bandwidth provisioning. In *Proc. of INFOCOM*, Mar. 2006.
[42] L. Song, D. Kotz, R. Jain, and X. He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proc. of INFOCOM*, Mar. 2004.
[43] Memory cards. http://en.wikipedia.org/wiki/Comparison_of_memory_cards.
[44] Toyota and Honda vehicle-to-vehicle communication systems. http://www.motorauthority.com/toyota-and-honda-start-testing-vehicle-to-vehicle-communications-systems.html.
[45] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.
[46] Wifi-taxi. http://www.wifi-taxi.com/cgv_en.htm.
[47] Wikipedia. F1 score. http://en.wikipedia.org/wiki/F1_score.
[48] Worldwide pricelist for iPhone 3G Plans. http://www.unwiredview.com/2008/07/16/worldwide-pricelist-for-iphone-3g-plans/.
[49] Yahoo! Local Search Web Services. http://developer.yahoo.com/search/local/V3/localSearch.html.