MobTorrent: A Framework for Mobile Internet Access from Vehicles

Bin Bin Chen
Department of Computer Science
National University of Singapore
Email: chenbinb@comp.nus.edu.sg

Mun Choon Chan
Department of Computer Science
National University of Singapore
Email: chanmc@comp.nus.edu.sg

***Abstract*—In this paper, we present MobTorrent, an on-demand, user-driven framework designed for vehicles which have intermittent high speed access to roadside WiFi access points (AP). Mobile nodes in MobTorrent use the WWAN network as a control channel. When a mobile client wants to initiate a download, instead of waiting for contact with the AP, it informs one (or multiple) selected AP(s) to prefetch the content. The scheduling algorithm in MobTorrent then replicates the prefetched data on the mobile helpers so that the total amount of data transferred and the average transfer rate to the mobile clients are maximized. Therefore, instead of limiting high speed data transfer to the short contact periods between APs and mobile clients, high speed transfers among vehicles are opportunistically exploited.**

Evaluation based on testbed measurement and trace-driven simulation shows that MobTorrent provides substantial improvement over existing architectures. For the case of a single AP, its performance approximates that of an off-line optimal scheduler. In case of multiple APs, our evaluation shows that MobTorrent's performance is robust in a variety of settings.

I. Introduction

For commuters and passengers on public buses, taxis or private vehicles, the most common and seamless way of getting Internet access is through the use of wireless wide-area network (WWAN), for example, GPRS, 3G or HSDPA. The WWAN radio can be plugged into the end host (e.g. a laptop) or mounted on the vehicle from which shared network access is provided to all passengers in the vehicle using IEEE 802.11. However, even though performance of WWAN has improved significantly over the years, in particular with deployment of HSDPA, the aggregate or per user data rate is still limited by the need to provide ubiquitous coverage to a large number of users. In a recent measurement, we observe around 300Kbps download speed from a vehicle (with a 1.5Mbps-limit subscription plan) using a local commercial HSDPA network.

Meanwhile, many cities around the world have witnessed large scale deployment of open IEEE 802.11 or WiFi hot spots. In Singapore, more than 7000 free WiFi access points (APs) have been deployed in the last few years in public open areas, shopping malls and commercial buildings. On a smaller scale, in a measurement on our 150 hectares campus in Kent Ridge, we can observe more than 2000 APs installed in 90 buildings. Strong WiFi signal can be received from about 25% of the 4km route traveled by the campus shuttle bus. Recent research works [1], [2] have also demonstrated the feasibility of providing network access via road-side APs.

On the other hand, Singapore being a city state, has a dense deployment of public buses. The largest public bus transportation company has a fleet of more than 2000 buses. Currently, almost all buses are equipped with GPS and GPRS device. While the bandwidth provided by GPRS is sufficient for its main application, an Automatic Vehicle Location (AVL) system, it is too low to support Internet access service for passengers. Upgrading the whole system to HSDPA is costly. With such a large number of open WiFi APs available already, providing network access to moving vehicles through roadside WiFi APs offers an *alternative* and *complementary* solution that can significantly increase the bandwidth available to the vehicles.

Heterogeneous mobile wireless broadband access architectures for commuters have been suggested in [3], [4], where multiple network interfaces (e.g. 3G and WiFi) are available and can be utilized simultaneously. The concept of Always Best Connection (ABC) is often adopted, where mobile nodes automatically start to use the WiFi network at the moment an AP is in range. While WiFi provides higher bandwidth at cheaper price, it is only usable when the vehicle is in range and the contact duration is often short. In comparison, while the speed of WWAN link is lower, it has higher availability.

In order to fully exploit the available high bandwidth but intermittent contacts, we propose MobTorrent, an on-demand, user-driven framework designed to optimize performance for vehicular network. The approach taken in MobTorrent is different in that we use WWAN mainly as a control channel and assume that mobility information can be predicted with high accuracy using AVL system and past history. In this framework, a mobile client, instead of waiting for contact with the AP, uses the WWAN radio (e.g. GPRS) to inform one (or multiple) selected AP(s) to prefetch the content. The prefetched data are then replicated on the mobile helpers, and further propagated by the latter in a store-carry-forward, i.e., epidemic Disruption Tolerant Network (DTN) routing fashion. As a result, instead of limiting high speed data transfer to a few short contact periods with the selected APs, high speed transfer among vehicles can be opportunistically exploited.

While MobTorrent exploits prefetching and replication, the key component is the scheduling algorithm, which replicates the prefetched data by taking into account locations of the mobile nodes and existing level of data replication. The objective is to maximize the total amount of data transferred and the average transfer rate to the mobile clients.

In this work, we first characterize the performance limits of opportunistic mobile forwarding through a simple scenario using only one AP. The insight gained is then used to design the scheduling scheme for inter-vehicle transmission. In the evaluation, we use testbed measurement to verify the benefit of prefetching and use trace-driven simulation to evaluate performance of MobTorrent. Simulation result shows that MobTorrent provides substantial improvement over existing architectures and often performs close to what can be achieved by an off-line optimal scheduler. In case of multiple APs, our evaluation shows that MobTorrent is robust in a variety of settings.

The rest of paper is organized as follows. In Section II, we present related works. In Section III, we describe the architecture of MobTorrent. In Section IV, we discuss scheduling issues and analyze the performance gain. In Section V, we evaluate the performance of MobTorrent. We conclude in Section VI.

II. Related works

Due to the complementary nature of WWAN and WiFi, there have been a number of research efforts that try to combine them. In many of these approaches, only the WWAN base stations are gateways to Internet and WiFi is used to improve the performance of the WWAN infrastructure, for example, for coverage expansion [5], load balancing [6], and better channel utilization [7]. A survey of these approaches can be found in [8]. In comparison, we use WWAN mainly as a control channel for the vehicle to send out request and acknowledge the data it has received.

In the area of vehicle network access using WiFi, the authors of [1] propose an architecture to support so-called drive-through Internet. The key component is a session protocol (PCMP) that offers *persistent* end-to-end communication even though the vehicles on the road have only intermittent contact with road side APs. In their work, for vehicles with velocity from 40km/h to 180km/h, a few megabytes could be transferred to and from the mobile node using TCP and UDP. In [2], as part of the MIT Cartel project, the authors measure the upload bandwidth available to vehicles in the Boston metropolitan area using in-situ unplanned open APs. The result is also encouraging. The upload TCP bandwidth has a median of 30 KB/s, and median transfer size per contact duration is 216KB. Cabernet [9] further improves the performance by optimizing both the connection establishment procedure (QuickWiFi) and transport protocol (CTP). Another measurement of WiFi connectivity from moving vehicle can be found in [10]. In [11], the authors investigate scheduling issues for vehicle uploading or downloading from a road side unit. [12] proposes ViFi, a protocol that opportunistically exploits base station diversity to minimize disruptions and support interactive applications for mobile clients. In comparison, we focus on the setting where road-side WiFi only provides partial coverage (around 25% from our measurement or even lower), so that the main application of interest is delay tolerant bulk file transfer.

Another direction in the area of vehicular communication is from the angle of Disruption Tolerant Network (DTN). Epidemic routing [13] proposes the “store-carry-forward” paradigm for intermittently connected networks. They use the hop count of messages to regulate the resource usage. [14] shows that binary splitting is optimal under certain assumptions for spreading a given number of replicas into network. To improve over blind replication, [15] proposes PROPHET, which is shown to work better than epidemic routing, based on the observation that real users tend to move in a predictable fashion with repeating behavioral patterns. UMASS’s DieselNet project presents a study of vehicle (public buses) contact time in [16], and proposes a series of routing protocols in [17], [18] and [19]. There are several major differences between MobTorrent and existing DTN routing works, as most of the latter are designed for the general case where the mobility pattern is largely structure-less, and using historic meeting information is recognized as a good heuristic to estimate future contact probability. In above systems, the target application is communication among mobile nodes, the target message delivery delay is often in the scale of hours (e.g., the average delivery delay is 67.5 minutes using epidemic routing in DieselNet trace [16]), and the target delivery rate is dozens of packets per hour. Instead, our interest is to use the capacity of intermittently connected network to supplement the bandwidth of WWAN, and we focus on the data transmission between road side AP and mobile clients. The target delivery delay is in a few minutes, and the target delivery rate is hundreds of Kbps, which is comparable to that of HSDPA network. To achieve this, we make full use of the mobility information from in-situ AVL system.

Mobile ferry routing is proposed in [20], [21] for data delivery in a sparsely connected network. The main idea is to introduce some non-randomness in node movement or actively change trajectories to help deliver data. However, in the scenario we are interested in, it is not likely that nodes will move just to accommodate communication.

Prefetching has been used extensively to speedup web download, see [22]. In a mobile, vehicular environment, prefetching has also been proposed to speed up access to result of web queries [23]. We used prefetching in a different way, as the file required is known, while the uncertainty comes from the varying contact opportunities.

III. Architecture

A. Components

In order to deploy MobTorrent, we require wide adoption of GPS devices on vehicles (e.g., Japan’s vehicle navigation system installation rate is estimated to be as high as 59%, while Europe and the United States are around 25% [24]). In addition, each vehicle must be equipped with WiFi interface and WWAN interface. We use WWAN mainly as control channel, so the existing low bit rate GPRS suffices. Vehicles are expected to have an estimation of their travel route. This can be obtained from historical values or based on locations and digital street maps. We believe that all of these are reasonable requirements, in particular, for public buses and vehicles that travel on regular routes.

The components of MobTorrent are shown in Figure 1.

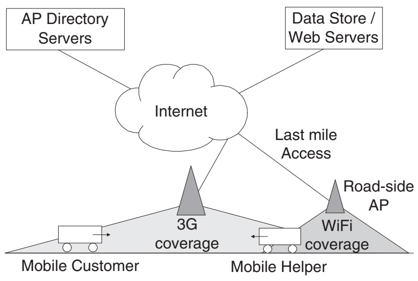


Fig. 1. Components of MobTorrent Architecture

Mobile Clients are vehicles that require help to download data from Data Store / Web Servers through Internet.

Road-side APs are static WiFi access points reachable from the road that have Internet backhaul, and offer their services to mobile clients. They can be residential gateways in apartments or installed as part of a vehicle network infrastructure that is placed along the road, say at bus stops, taxi stands, or traffic lights. Note that the backhaul downlink bandwidth to these APs can be lower than the wireless bandwidth available on the 802.11 link. For example, in a residential home, the downlink speed could be a few Mbps or less but the average WiFi bandwidth can be over 20Mbps for 802.11g and much higher for 802.11n.

AP Directory Servers provide location information on available road-side APs. There are a number of open WiFi AP locators available on the web already, including *http://www.openwifispots.com*, *http://www.fon.com*, and *http://www.whisher.com*. The location of participating APs need to be in the form of coordinates given in longitude and latitude, which can be easily found even without GPS by using digital street maps. Depending on the system requirements, these servers can also maintain information related to the AP's reputation and performance. For scalability purpose, it is likely that the servers are clustered into different geographical regions.

Mobile Helpers are idle vehicles willing to offer their bandwidth to help peer vehicles with downloading demand.

B. Control and Data Flow

In this section, we describe a typical operation in MobTorrent data transfer, as illustrated in Figure 2.

Initially, a mobile client wants to download a (sufficiently large) file. Note that small requests are assigned to the always-on WWAN link to minimize their delay. By downloading large files via WiFi, the WWAN link can be less congested, which benefits the small requests too. The mobile, with its location known through a GPS device, acquires the list of APs that are along its travel path. Based on a number of parameters (file size, location of AP, estimated travel time to the AP), the node selects a set of APs and contacts them through its WWAN interface. For example, as shown in Figure 2, 2 data blocks are needed, while AP 1 and AP 2 are selected.

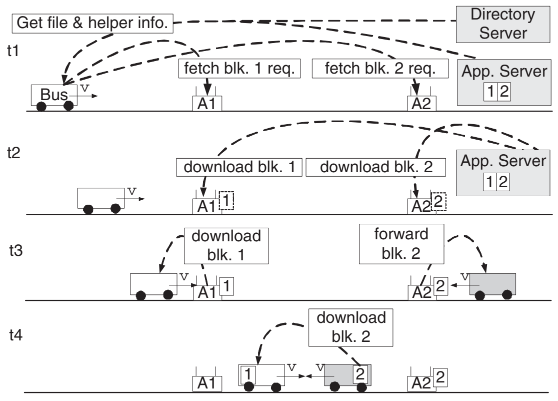


Fig. 2. Flow of Data Downloading

At time t_1, the mobile client requests AP 1 to prefetch block 1 and AP 2 to prefetch block 2, and the two APs begin to download the respective blocks.

At time t_2, the blocks are downloaded to the corresponding APs and cached locally.

At time t_3, the mobile client travels within range of AP 1 and downloads block 1 from AP 1. At the same time, the mobile helper, a second bus moving towards the mobile node, enters the coverage of AP 2. AP 2 sends block 2 to the mobile helper.

At time t_4, the mobile client and the mobile helper meet at some point between AP 1 and AP 2. The mobile helper transfers block 2 to the mobile client, thus completing the transfer even before the mobile client reaches AP 2.

In order to efficiently orchestrate the whole downloading process, two questions need to be answered: (1) How much data should an AP prefetch? (2) How to relay the data to a client via mobile helpers, so that the amount is maximized and the delay is minimized. We address these two issues in the next section.

IV. Scheduling in MobTorrent

A. Roles and Functions of Different Mobile Helpers

Before presenting the scheduling algorithm in MobTorrent, we first draw insight from how opportunistic relay should work in a simple mobility model on a 2-way street. This model abstracts the major properties for some typical settings, such as commuters on highway, and public buses with fixed routes within a city.

We consider a relatively sparse vehicle network, where the probability of forming a contemporaneous multi-hop path is negligible, so single hop forward in each contact opportunity is the main form of data transfer. Vehicles are assumed to move on a long, two way street without divergence in the path. A vehicle moves in one of the two directions (LEFT or RIGHT) on the road and it never changes its direction. In addition,

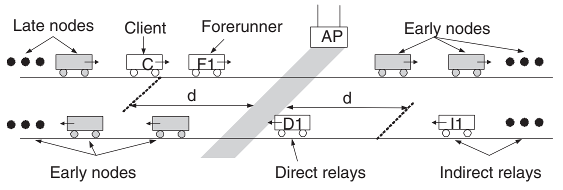


Fig. 3. Classes of helpers

there is no overtaking among the vehicles moving in the same direction. We focus on the case of a single AP in the model.

We define an *opportunistic contact* as the time period that two peers get in communication range, and can exchange data with each other directly. We define the *contact capacity* as the amount of data that can be exchanged in a contact. Note that the two directions of transfer compete for the same contact duration, so the contact capacity limits the sum of amount that can flow in each direction.

Given the time and client's location when the request is generated, we can categorize all mobile helpers into the following classes based on their moving directions and positions (relative to the client and AP) as shown in Figure 3:

- **direct relay:** mobile nodes which move towards client and meet AP after the request is generated but before they meet client. As the name suggests, direct relay can get data from AP, carry it, and *directly* send to the client.
- **forerunner:** mobile nodes which move in the same direction as client and meet AP after the request is generated but before client meets AP.
- **indirect relay:** mobile nodes which move towards client and meet client before they meet AP. Note that when an indirect relay meets AP, there is no need for AP to send data to it, as it will not meet client or any node that will meet client from then on.

The set of direct relays, forerunners, and indirect relays are denoted as D, F, I respectively. All other mobile nodes cannot participate in the transfer for the following reasons:

- **early node:** mobile nodes that have moved past the AP before the request is generated. They cannot help the data delivery because they cannot receive data directly from the AP, or from any vehicle carrying the desired data.
- **late node:** mobile nodes that are always behind the client. They cannot help the data delivery because they cannot send data directly to client, or to any mobile node that will meet the client after meeting them.

Figure 4 (a) shows the trace of vehicles and their contacts along the time axis. We denote a contact using 4 parameters, namely $\{$*node1, node2, contact time, contact capacity*$\}$. As shown in Figure 4 (a), at time $t1$, AP (A) meets the first direct relay (D_1) (as their trace intersects), with contact capacity 2, which is the number marked at the intersection. This contact can be represented as: $\{A, D_1, t1, 2\}$. Similarly, we can write down the rest of contacts from the figure in their sequence as: $\{D_1, F_1, t2, 1\}$; $\{A, F_1, t3, 2\}$; $\{D_1, C, t4, 1\}$; $\{F_1, I_1, t5, 3\}$; $\{C, A, t6, 2\}$; $\{F_1, C, t7, 3\}$.

B. Performance Limits

The two performance metrics of interest are (1) the maximum amount of data sent by the AP that reaches the client eventually, and (2) the minimum delay to deliver a given amount of data. We assume that there is sufficient buffer on all nodes to accommodate packets in transfer, and the only bottleneck is the contact constraint between nodes.

Let the time and capacity of contact between node i and node j be denoted by t^i_j and $\mathcal{C}^i_j$ respectively. Denote AP and client as node A and C.

1) Maximum Data Transfer from AP to Client: The maximum amount of data $\mathcal{C}$ that the AP can push to network and stand a chance to reach client is:

$$\mathcal{C} = \mathcal{C}^A_C + \sum_{i \in D} \mathcal{C}^A_i + \sum_{j \in F} \mathcal{C}^A_j \tag{1}$$

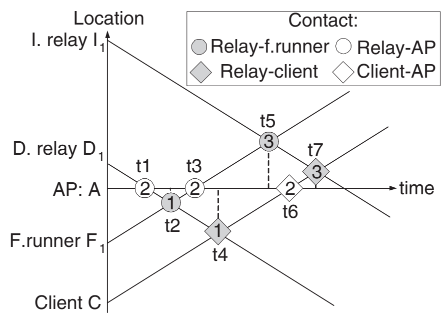
Note that sending data to the rest of nodes (the indirect relay and late nodes) is useless. However, the capacity estimated in Equation (1) often can not be achieved because loss of data at direct relays is unavoidable if replication (on the forerunners and client) cannot be completed before the direct relay goes past the client. Loss is minimized if a direct relay uses all possible contact capacity with forerunners and client to copy the data that it gets directly from AP. So a tight bound for the capacity achievable by AP is:

$$\mathcal{C}^A_C + \sum_{i \in D} min(\mathcal{C}^A_i, \mathcal{C}^i_C + \sum_{t^j_i > t^A_i, j \in F} \mathcal{C}^i_j) + \sum_{j \in F} \mathcal{C}^A_j \tag{2}$$

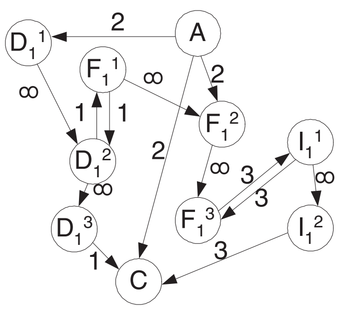
2) Minimum delay from AP to Client: In terms of the minimum delay to deliver a set of n blocks, a lower bound can be obtained by assuming that all contacts between client and a direct or indirect relay are fully utilized. However, this bound is also loose because it is possible that some contact capacity between client and a relay cannot be fully utilized as the relay does not carry enough new data.

In order to obtain a tight bound, we observe that it is possible to find the maximum amount of data that can reach client given a contact trace, by modelling it as a maximum network flow problem. Hence, we can perform an off-line computation to characterize the performance. Given a sequence of contacts $\{c_1, c_2, ...c_n\}$ between the different nodes, the graph $G = (V, E)$ for the network flow problem is constructed in the following way.

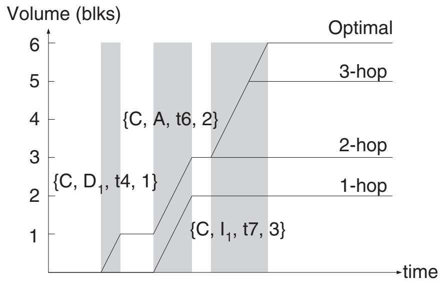
- **Vertices** There is one vertex A representing AP, and another vertex C representing client. They are the source and destination of the network flow problem. For each of the non-client vehicle v, if it experiences n contacts, we split it into n vertices, v^1 to v^n. These constitute all vertices in graph G.
- **Edges** For each contact, we use one (or two) directed edge(s) to represent it in the graph. For the contact between AP and client, we add a directed edge from A to C. For the contact between AP and a mobile helper v, if this contact is the i^{th} contact of the helper, we add a



(a) Time series trace



(b) Network flow formulation



(c) Performance of schemes

Fig. 4. A simple two-way street example

directed edge from A to v^i. For the contact between client and a mobile helper v, if this contact is the i^{th} contact of the helper, we add a directed edge from v^i to C. A single directed edge suffices because only the specified direction is useful to maximize the network flow from A to C. During the contact, the data flow should always follow the direction of the edge. For each edge added, the edge capacity is set to the corresponding contact capacity. For a contact between two mobile helpers u and v, if this contact is the i^{th} contact for vehicle v, and j^{th} contact for vehicle u, we add a pair of directed edges between v^i and u^j, as both of the directions may help to maximize the network flow. These two edges should share the same contact capacity (denoted as setting S1). However, as detailed below, we can set the capacity of both (instead of the sum of both) to the contact capacity (denoted as setting S2) without affecting the value of maximum flow. Finally, we add directed edge with unlimited capacity from v^i to v^{i+1}, for all v and valid i. These directed edges represent the fact that a vehicle can carry the data it received from a previous contact to the next contact. A finite edge capacity can be used to model buffer limit if required.

We can show that S1 and S2 have identical maximum flow solution in the following way. First, the optimal solution of S2 is no worse than S1. Second, given an optimal solution in S2, if there are flows over a pair of edges, it can be reduced to a solution with the same maximum flow using at most one of the edges, by offsetting the flow in the opposite direction until one of them becomes 0. It is easy to see that, by repeating above process for all pairs of edges, we can get a solution valid in S1. So S2's solution is no better than S1.

For example, the contact trace as depicted in Figure 4 (a) will result in the graph as shown in Figure 4 (b). Given the above formulation, the minimum delay for delivering a given amount of data can be calculated efficiently.

With the performance limits known, we next examine several typical schemes starting from the simplest.

C. Comparison of scheduling schemes

Figure 4 (c) shows the volume of data that can reach client using different schemes, under Figure 4 (a)'s setting.

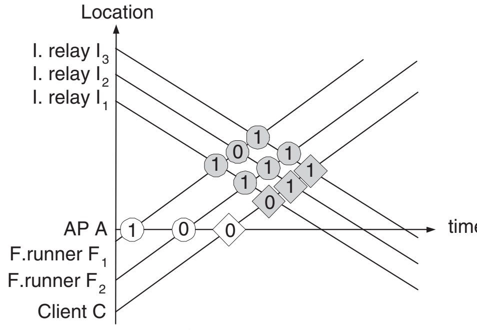


Fig. 5. Minimize delay

1) 1-hop scheme: AP only transfers directly to client during their contact. Volume of data delivered is $\mathcal{C}_{1-hop} = \mathcal{C}_C^A$. In the example, client can get 2 blocks of data from AP directly.

2) 2-hop scheme: In this scheme, AP sends data to client and direct relays. A direct relay keeps the data until it meets client, where it sends the data to client. The amount of data transferred by a direct relay (D_i) from AP to client is the minimum of two contact capacities, $\mathcal{C}_{D_i}^A$ and $\mathcal{C}_C^{D_i}$. Therefore, $\mathcal{C}_{2-hop} = \mathcal{C}_C^A + \sum_i min(\mathcal{C}_{D_i}^A, \mathcal{C}_C^{D_i})$. In the example, client can get 1 additional block of data from direct relay D_1. Note that another block sent to D_1 by AP is lost due to the low contact capacity between D_1 and client.

3) 3-hop scheme: In this scheme, AP sends data to client, direct relays and forerunners. Although forerunners cannot send directly to client, they can send their data via direct relay or indirect relay (thus 3 hops). Under our assumption, forerunners can meet enough relays to dump its data to client, thus all data sent to forerunners can reach client eventually. Therefore, $\mathcal{C}_{3-hop} = \mathcal{C}_C^A + \sum_i min(\mathcal{C}_{D_i}^A, \mathcal{C}_C^{D_i}) + \sum_j \mathcal{C}_{F_j}^A$. In the example, client can get 2 additional data blocks from indirect relay I_1, which itself gets the data from F_1. However, the 2^{nd} block carried by D_1 is still lost. Minimizing the loss of directed relay requires at least a 4-hop delivery.

4) 4-hop scheme: In this scheme, a direct relay saves its data as soon (and as much) as possible to forerunners before meeting client. This feature minimizes loss since forerunners can always transfer its data via indirect relay later. 4-hop delivery achieves the capacity as characterized in equation (2). In the example, the missing block from 3-hop scheme reaches client via four hops: $\{A, D_1\}$, $\{D_1, F_1\}$, $\{F_1, I_1\}$, $\{I_1, C\}$.

However, 4-hop is not yet optimal to minimize the delivery delay. Consider the situation as shown in Figure 5. The minimum delay to deliver the 1 data is through 5 hops, i.e., the contact of $\{A, F_1\}$, $\{F_1, I_1\}$, $\{I_1, F_2\}$, $\{F_2, I_2\}$, $\{I_2, C\}$. Replication among forerunners (F_1 and F_2) is necessary to minimize the delay. Note that, the replication can only be carried out from upstream forerunner (F_1) to downstream forerunner (F_2).

D. MobTorrent scheduling

Based on observations summarized in previous section, the scheduling algorithm used by MobTorrent is as follow:

- **Meta data:** MobTorrent keeps the following meta information with each data block k at each relevant node:
  1) Req_k: request id, b_k: block id;
  2) Ack_k: whether this block has reached client;
  3) r_k: a (local) estimation of the *persistent* replication level of this block in the whole system;
  4) ID_k: the ID of the most upstream forerunner possessing this block.
- **First hop:** Assuming AP always has new data to send to client. When a data block k is sent to a forerunner F_j, the latter marks $r_k = 1$ and $ID_k = F_j$. When a data block k is sent to a direct relay, the latter marks $r_k = 0$, and $ID_k = 0$. Replication on direct relay is not counted as a persistent replication.
- **Meta data reconciliation:** When two vehicles A and B meet, they will first exchange their meta data. Suppose the replication level estimation of block k is r_k^A at A, and r_k^B at B. After the exchange, both of them will set their estimation to $max(r_k^A, r_k^B)$. For all the following transmissions within this contact, the replication level of transmitted block will be updated to the same new value at both sides. $Ack_k^A = Ack_k^B = Ack_k^A \vee Ack_k^B$.
- **Priority calculation:** Suppose that the set of undelivered blocks (with $Ack = false$) at A is S_A, and the set of undelivered blocks at B is S_B. After the exchange, they calculate $S_A - S_B$ and $S_B - S_A$, which are the candidate set to be sent to each other. A node sorts its candidate blocks according to replication levels, *giving highest priority to block with the lowest level of replication*. In case of ties, the block is sorted by ID with downstream forerunner first. Therefore, block stored by a downstream forerunner has higher priority. This is because replication can happen only from upstream forerunner to downstream forerunner. Thus, in a long term, block from upstream forerunner has more opportunity to be replicated. Given a selected direction, data transmission is performed according to the priority calculated. Between two mobile helpers, the following three rules, i.e. *minimize loss*, *maximize rate*, and *increase replication level*, take action in order to specify the scheduling of transmission directions.
- **Minimize loss:** If one party is a direct relay, it may carry blocks with $r_i = 0$, i.e., the data blocks that the relay received directly from AP and have not yet been sent to any other node. Whenever such blocks exist, transfer opportunity is given to direct relay to minimize loss. After the transmission, both parties set the replication level of the block to 1, and set the ID to the ID of the forerunner.
- **Maximize rate:** After the loss-minimizing step is done, this rule ensures that direct or indirect relay have enough *new* data so that its contact capacity with the client can be fully used. Based on the contact statistic, a threshold γ is selected. We set γ as two times the expected transmission capacity between vehicles. While this threshold is not reached yet, transfer opportunity is given to forerunner. After the transmission, the replication level of the block is increased by $\alpha < 1$. This value is less than one because unlike replication on the forerunner, the relay can go past the client without delivery or replicating this data block and this particular copy is lost. As it is difficult to determine the "best" value, we simply set it to 0.5. Our simulation results show that performance does not change much when this value is varied.
- **Increase replication level:** Once the threshold for new data block is reached, data exchange happens in both directions. Remained candidate blocks from both sides are merged into a single priority queue sorted by replication level and ID. In case of ties, data stored on direct or indirect relay is given higher priority, as transferring it will increase the replication level by 1, while the transfer at the other direction will only increase replication level by α.
- **Last hop:** When the direct or indirect relay meets the client, blocks are transferred from the relay to the client according to their priorities. When the contact finishes, the client uses WWAN to update APs and all forerunners about new blocks that it has just received. Note that the client does not need to update direct and indirect relays, as they will be updated when they get contact with forerunners. Once passing by the client, a relay removes all of its data for the client (even those that have not been delivered yet), as there is no more opportunity to deliver them.

The intuition behind the scheduling scheme can be explained as follow. When few data blocks have been delivered, the relays often have sufficient "new" data to fully utilize the contact capacity with the client. However, as more blocks are delivered, it becomes harder for the relay to transfer sufficient undelivered data to the client if its contact capacity with forerunners does not match up well with the amount of undelivered data on them. As a result, data delivery rate to the client decreases as more blocks are delivered. MobTorrent scheduling scheme is designed such that as more blocks are delivered, the replication level of the undelivered data increases. In this way, data delivery rate can be maintained at a high level till all blocks are delivered.

Scheduling decisions in MobTorrent have some similarity in the spirit to the scheduling decision made in existing DTN routing, for example, MaxProp [18] and RAPID [19]. However, in MobTorrent, the information of direction and relative position are fully exploited to optimize performance.

In the model presented, we have assumed that there is no overtaking and nodes do not leave the system. The MobTorrent system will work in the presence of overtaking and path divergence. The impact of these factors will be evaluated using simulation in the next section.

V. Performance Evaluation

A. Testbed Configuration

We build a simple MobTorrent prototype to evaluate its performance in a real environment. We equipped 16 campus buses with an on-board LinkSys WRT54GL router as mobile clients. These clients run on the OpenWRT operating system. Each client is fixed at a bracket in front of the driver, and draws power from the bus. When the bus is moving, the client scans and attempts to associate to the campus WiFi network. Once it successfully associates with an AP, the client uses a pre-stored map to figure out the valid IP address that it can use (the school APs along the route belong to several different IP subnets). We pre-load the mapping between AP and IP subnet on all clients to reduce the overhead of IP address acquisition. Similar optimization can be done on MobTorrent, as the AP and client can exchange IP and authentication information via WWAN before they meet. Live bus tracking is available at http://mobtorrent.ddns.comp.nus.edu.sg/. Figure 6 gives a snapshot of the system. The campus shuttle buses run in a circle from both directions around the Kent Ridge campus. The average time for a bus to complete the 4km route is about 20 minutes. Over 120,000 contact statistics are collected from more than 1,300 driving hours over a 2 month period. The mean bus-AP contact duration is around 15s with mean contact capacity around 4.5MB, while the mean bus-bus contact duration is around 11s with the mean contact capacity around 3.2MB.

The evaluation has two parts. First, we evaluate the benefits of prefetching on the testbed. Next, we evaluate the benefits of using mobile helpers.

B. Benefits of pre-fetching

When a client sends its request to an AP through WWAN before contact, the AP prefetches the data and stores it locally. In order to evaluate the potential gain, the client is programmed to download several selected files via the WiFi network. In the measurement, there were 100 attempts to download each of these files over the WiFi network when the clients were within range of a campus AP. Three of the selected files are shown in Table I together with the average downloading duration, downloading speed, and the downloading completion ratio. The completion ratio is computed as the number of times the files were completely transferred in a single AP-bus contact duration over the total number of attempts. Note that, as the file size increases, it becomes less likely that the file can be downloaded successfully in a single attempt. For the largest file of 6.5MB, complete downloading from Internet is possible only 1 in 6 attempts.

For comparison, the files are stored on the APs in advance and downloaded to the mobile node on request. As shown in Table I, there is a significant improvement of downloading performance for files of all sizes. Note that such improvement is possible for all downloads made using advance requests, independent of the scheduling algorithm.

TABLE I
Download performance

File	Source	Time	Speed	Ratio
A pdf file	Internet	3.2s	25KBps	68%
(80KB)	local AP	0.33s	240KBps	98%
OpenWRT firmware	Internet	12s	111KBps	45%
(1513KB)	local AP	1.77s	853KBps	78%
A 3 minute video clip	Internet	33s	201.5KBps	16%
(6.5MB)	local AP	6.7s	993KBps	59%

TABLE II
RTT measurement (ms)

Link	25%	50%	75%	95%
end-to-end wired Internet links	37.6	81.0	160.5	330.6
One hop WiFi Link	2.347	2.742	4.668	26.129

A closer look at the source of performance gain reveals that, for the first file with a small size (80KB), the downloading duration difference is mainly due to decrease in RTT (round trip time). Table II shows the measured RTT distribution for end-to-end wired Internet Links and the local single WiFi link. The end-to-end wired Internet Link RTT is obtained from the Internet End-to-end performance Measurement (IEPM) with 413 different pairs of nodes across several continents. The local RTT is measured using ARP packets sent from the client box to the AP. ARP is used because many APs on campus do not response to ping but to ARP request. The measurement shows that the RTT of a local WiFi link is about one magnitude shorter than a typical RTT over Internet. Shorter RTT allows TCP to increase its congestion window at a faster rate, which helps to shorten the downloading time for short files. For the larger files, the speed difference is also due to the avoidance of bottlenecks in the Internet. Though the download rate constraint from the server can be alleviated by using parallel downloads, the constraint in the local backhaul link (such as ADSL, cable, or wireless mesh) cannot be bypassed.

C. Benefits of scheduling

We compare performance of the following schemes.

1) *1-hop:* AP only sends data directly to client. This serves as the baseline of performance.
2) *Random:* A scheduling scheme which employs random replications between peers. When both sides have innovative data for each other, one side is randomly selected to transfer a randomly selected innovative block.
3) *Greedy:* A scheduling scheme which greedily replicates data in the order of their expected deduction in delay to reach client, following the design of RAPID [19].
4) *MobTorrent:* as described in Section IV-D.
5) *Ideal:* A perfect download mechanisms that have perfect off-line knowledge of contact time and achieve the maximum delivery rate to the client by solving the network flow problem.

Note that, because of the significant difference in settings and assumptions, we can not directly compare MobTorrent with existing related works, including PROPHET [15], Spray and Wait [14], MV routing [17], MaxProp [18], and RAPID

[19]. For example, in our testbed, if two nodes just meet and pass by each other, the probability that they will meet again in recent future is almost 0. However, all existing schemes tends to assign a higher meeting probability to this pair of nodes. For a fair comparison, we implement the Random and Greedy scheme such that, block is never replicated to a node which has no chance to deliver it to the client, while still keeping their original salient features.

1) Performance with single AP and single client: In this scenario, there is only one AP located on a 2-way street with an infinite flow of vehicles moving in both directions. The contact capacity is generated using the trace collected from testbed. 100 rounds of simulation are run, and the average is presented in Figure 7 (a). To make the averaging meaningful, for each run, the time and volume is normalized so that the optimal network flow scheme reaches an optimal volume capacity of 100 at time clock 100. In the simulation, the average number of forerunners is 5.

As shown in the figure, MobTorrent is close to optimal in terms of both the volume of data delivered and the delay to deliver data. At time clock 100, MobTorrent, random, greedy and 1-hop schemes deliver 91%, 78%, 71% and 11% of all data respectively.

The random scheme delivers most of the data eventually, but takes much longer than MobTorrent, due to the coupon collection phenomenon where new data is difficult to locate towards the end. MobTorrent alleviates such effect by giving priority to deliver downstream forerunner's data from the very beginning. The greedy scheme does not reach the volume capacity (with a 20% gap) because the greedy transfer of data from forerunner to direct relay prevents the latter from replicating its data into network. As expected, the 1-hop scheme only achieves a small fraction of the available capacity.

Next, we evaluate the impact of varying number of forerunners. The average number of direct relays is the same as the number of forerunners. We defined average data rate as the ratio of total data volume delivered and the time taken to deliver the last packet (lost packets are ignored). Figures 7 (b) and (c) show that when the number of forerunners (and hence helpers) increases, both the average data rate and the total volume of data delivered increase. In terms of average data rate, the rate of increase for MobTorrent tracks the optimal algorithm fairly well, while the random and greedy schemes improve at a slower rate. 1-hop scheme does not benefit from mobile helpers at all.

2) Performance with multiple APs and multiple clients: We use the mobility and contact trace collected from the testbed to drive the simulation. In the simulation, the request arrival to the whole system follows a Poisson process with average rate of one request per $20s$. A running bus is randomly selected as the source of the request, whose size follows an exponential distribution with mean $5MB$ (MobTorrent scheduling scheme's advantage over other schemes increases with the file size. The figure is omitted due to lack of space). We fix the number of APs in the network to 5, and varies the number of buses in the network. Note that the average load in

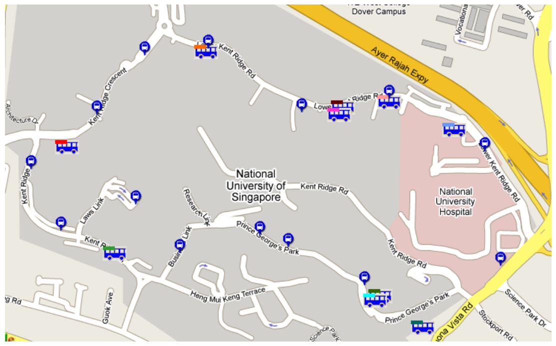


Fig. 6. A snapshot of NUS bus monitoring system.

the system does not change with the number of buses.

As shown in Figure 8(a), schemes using mobile helpers improve data download rate. As the number of buses grows, more mobile forwarding opportunity can be exploited.

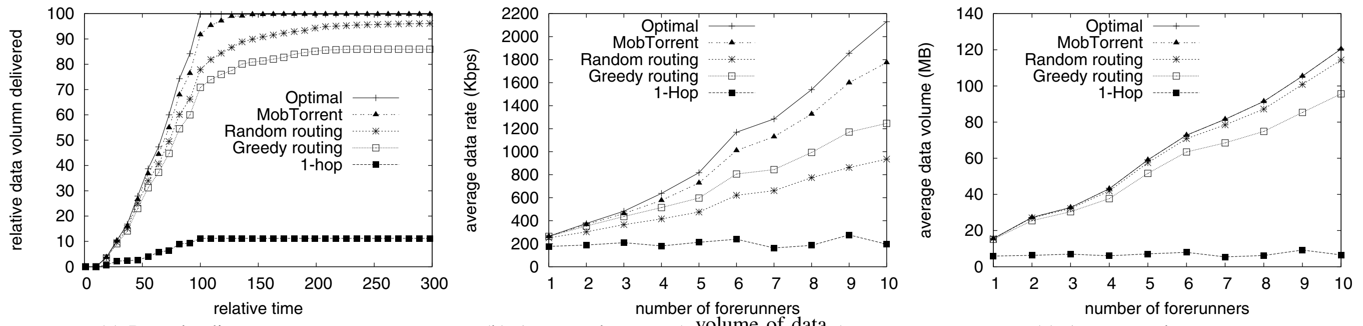
We also investigate the impact of vehicle overtaking by varying the vehicle velocity. We achieve this by sampling the trace of bus from both peak hour when bus tends to be slower and off hour. As the variation of vehicle velocity increases, the overtaking probability increases. For example, when the variation increases to 0.6, the ratio of contacts due to overtaking is 25%. Figure 8(b) shows that the average data rate remains fairly constant with respect to overtaking, with the MobTorrent scheduling scheme constantly outperforms other schemes in all velocity variation settings. Since location information is not utilized by the other 3 schemes, it is not surprising that their performances remaining fairly stable. For MobTorrent, when the relative node locations are not static any more, the performance is fairly robust for the following reasons. First, blocks at downstream forerunner are given higher priority, reducing the impact of overtaking since these blocks may have already been delivered when overtaking occurs. Second, when a forerunner is overtaken by the client or another forerunner, a new opportunistic contact between the two nodes is created, which would not have occurred without overtaking. The overall impact of overtaking on MobTorrent's performance is not significant.

Finally, we investigate the impact of path divergence by letting vehicles disappears from the system suddenly. As shown in Figure 8(c), all forwarding schemes' performance degrades as the disappearance rate increases. While MobTorrent's performance remains the best among all schemes evaluated, overall performance is similar among MobTorrent, random and greedy schemes when path divergence occurs with a probability higher than 20% per node per minute.

VI. Conclusion

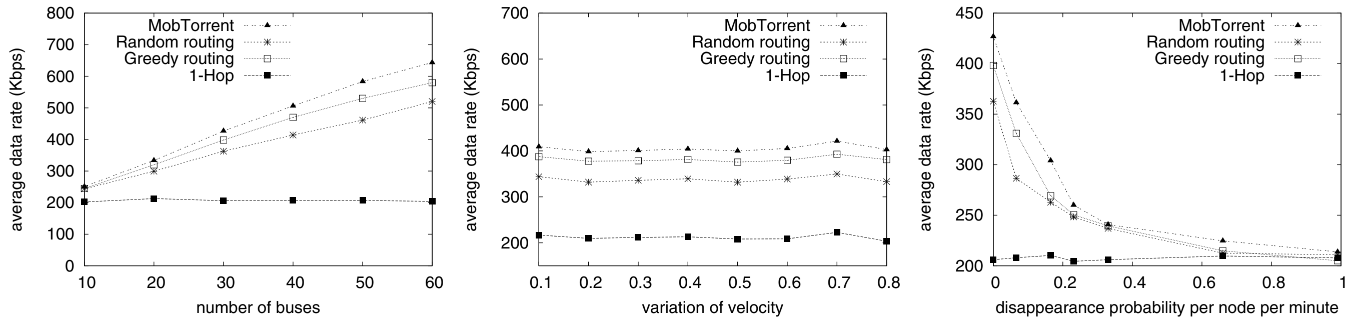
In this paper, we present MobTorrent, an on-demand, user-driven framework for vehicles to access Internet via road side static access points and other mobile vehicles on the road.

MobTorrent has the following components. In order to improve network throughput performance, prefetching and caching is used to better exploit the short contact time between AP and client by having the data locally available for transfer. In addition, to address the issue of low coverage, data can



(a) Downloading progress (b) Average data rate ($\frac{\text{volume of data}}{\text{delay of last data}}$) (c) Average volume

Fig. 7. Single-AP, single-client, ideal two-way street



(a) Average data rate (b) Impact of overtaking (c) Impact of path divergence

Fig. 8. Multi-AP, multi-client, testbed trace

be pushed to mobile helpers so that areas where WiFi can be used for data transfer is not limited to coverage of static road side APs but expands to include areas covered by mobile nodes. Evaluation based on trace-driven simulation shows that MobTorrent provides substantial improvement over other existing architectures.

VII. Acknowledgment

This research is supported by the National University of Singapore under grant R-252-000-359-112.

References

[1] J. Ott and D. Kutscher. A disconnection-tolerant transport for drive-thru Internet environments. In *INFOCOM*, Miami, FL, USA, March 2005.

[2] V. Bychkovsky, B. Hull, A. Miu, H. Balalkrishnan, and S. Madden. A measurement study of vehicular Internet access using in situ Wi-Fi networks. In *MobiCom*, Los Angeles, CA, USA, September 2006.

[3] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banjeree. MAR: A commuter router infrastructure for the mobile Internet. In *MobiSys*, Boston, MA, USA, June 2004.

[4] M. Buddhikot, G. Chandranmenon, S. Han, Y.W. Lee, S. Miller, and L. Salgarelli. Integration of 802.11 and third generation wireless data networks. In *INFOCOM*, San Francisco, CA, USA, March 2003.

[5] G. Aggelou and R. Tafazolli. On the relaying capacity of next generation GSM cellular networks. *IEEE Personal Communication Magazine*, 8(1):40–47, Feb 2001.

[6] H. Wu, C. Qiao, S. De, and O. Tonguz. Integrated cellular and ad hoc relaying systems: iCAR. *IEEE Selected Areas in Communications*, 19(10):2105–2115, October 2001.

[7] H. Luo, R. Ramjee, P. Sinha, and S. Lu. UCAN: a unified cellular and ad-hoc network architecture. In *MobiCom*, San Diego, CA, USA, 2003.

[8] H.Y. Hsieh and R. Sivakumar. On using peer-to-peer communication in cellular wireless data networks. *IEEE Transactions on Mobile Computing*, 3(1):57–72, January 2004.

[9] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular content delivery using WiFi. In *MobiCom*, San Francisco, CA, USA, September 2008.

[10] R. Mahajan, J. Zahorjan, and B. Zill. Understanding WiFi-based connectivity from moving vehicles. In *IMC*, San Diego, CA, USA, October 2007.

[11] Y. Zhang, J. Zhao, and G. Cao. On scheduling vehicle-roadside data access. In *VANET*, Montreal, QC, Canada, September 2007.

[12] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. Levine, and J. Zahorjan. Interactive WiFi connectivity for moving vehicles. In *SIGCOMM*, Seattle, WA, USA, August 2008.

[13] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, CS-200006, Duke University, April 2000.

[14] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *WDTN*, Philadelphia, PA, USA, August 2005.

[15] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *SAPIR Workshop*, Fortaleza, Brazil, August 2004.

[16] X. Zhang, J. Kurose, B. Levine, D. Towsley, and H. Zhang. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In *MobiCom*, Montreal, Qubec, Canada, September 2007.

[17] B. Burns, O. Brock, and B. Levine. MV routing and capacity building in disruption tolerant networks. In *INFOCOM*, Miami, FL, USA, March 2005.

[18] J. Burgess, B. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. In *INFOCOM*, Barcelona, Spain, April 2006.

[19] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *SIGCOMM*, Kyoto, Japan, August 2007.

[20] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc*, Roppongi, Japan, May 2004.

[21] Q. Li and D. Rus. Communication in disconnected ad-hoc networks using message relay. *Journal of Parallel and Distributed Computing*, 63(1):75–86, January 2003.

[22] V. Padmanabhan and J. Mogul. Using predictive prefetching to improve world wide web latency. *Computer Comm. Rev.*, 26(3):22–36, July 1996.

[23] A. Balasubramanian, Y Zhou, W. Croft, B. Levine, and A. Venkataramani. Web search from a bus. In *CHANTS*, Montreal, Canada, September 2007.

[24] RNCOS: World GPS market forecast (2006-2008).