SPECIAL ISSUE

MetroNet: a disruption-tolerant approach for mobile downloads on metro systems

Li-Ping Tung1, Yi-Chao Chen2, Kun-chan Lan3 and Ling-Jyh Chen4*
1 Intelligent Information and Communications Research Center, National Chiao Tung University, Taiwan
2 Department of Computer Science, The University of Texas at Austin, USA
3 Department of Computer Science & Information Engineering, National Cheng Kung University, Taiwan
4 Institute of Information Science, Academia Sinica, Taiwan

ABSTRACT

In this study, we propose a disruption-tolerant network based system called MetroNet, which exploits the precise schedules of metro systems to pre-fetch data and facilitate mobile Internet downloads, even if an end-to-end path between the source (i.e. content sources) and the destination (i.e. users) does not exist contemporaneously. Using a comprehensive set of simulations, as well as real-world scenarios of the Taipei and Atlanta metro systems, we demonstrate that MetroNet can handle the intermittent Internet connectivity caused by mobility and therefore provide a better mobile download service. The simulation results demonstrate that the proposed call admission control algorithm utilizes the network resource effectively and provides better quality of service for mobile users than the first-come-first-serve and least-first algorithms.

1. INTRODUCTION

The last decade has witnessed an increasing demand for Internet services on mobile devices, and a variety of wireless access technologies, such as 3G/4G and WiFi, have been developed to meet the demand. People can now access the Internet with a high data rate on the move; however, a truly ubiquitous solution for mobile Internet access is still lacking because (i) WiFi-based solutions have limited the service coverage and (ii) the effective bandwidth of 3G/4G-based solutions is constrained, and their cost is prohibitive for mobile users. As a result, Internet connectivity for mobile devices is inevitably intermittent. Hence, a mobile Internet solution that can provide low-cost and ubiquitous Internet access on the move is highly desirable.

Rather than exploit high-cost cellular technology, a great deal of research effort has been devoted to providing ubiquitous Internet access via WiFi technology in vehicular networks [1, 2]. The advantages of the technology are its scalability with low entry cost and its support for a higher data rate; however, its short-range coverage is a major limitation. To cope with the problem of intermittent Internet connectivity, a number of solutions based on the concept of disruption-tolerant vehicular networks have been developed. For instance, Ott et al. proposed the drive-thru Internet [3, 4], which provides people in moving vehicles with wireless access via access points (APs) deployed along the roadside. Data communication in such vehicular networks is challenging because the data download performance depends to a large extent on the traffic density, the vehicle’s speed, an AP’s transmission range and other environmental parameters.

In this study, we focus on providing a reliable mobile Internet download service on metro systems. Unlike conventional vehicular networks, metro systems have permanent routes, usually operate on precise schedules and maintain good punctuality. To provide mobile users on metro systems with a good network experience, we propose a novel solution called MetroNet, which leverages the strengths of the mobile hotspot approach and the concept of disruption-tolerant networks (DTNs). More specifically, MetroNet exploits the precise schedules of metro systems to pre-fetch data and facilitate data transmission, even if an end-to-end path between the source and the destination does not exist contemporaneously. We also propose a call admission control (CAC) algorithm to improve the
quality of service (QoS) of Internet services provided by MetroNet.

To evaluate MetroNet, we conducted a comprehensive set of simulations with three scheduling algorithms, namely, the proposed CAC algorithm, the first-come-first-serve (FCFS) algorithm and the least-first (LF) algorithm, in two real-world scenarios (i.e. the Taipei mass rapid transit (MRT) system [5] and the Metropolitan Atlanta Rapid Transit Authority system [6]). The simulation results demonstrate that the CAC algorithm outperforms the FCFS and LF algorithms in terms of the system gain, the number of accepted requests and the session times in all test cases.

The contribution of this study is three-fold:

(1) We propose a novel solution, called MetroNet, to facilitate mobile Internet downloads on metro systems with intermittent Internet connectivity. The system combines the strengths of the mobile hotspot approach and the concept of disruption-tolerant networks. As a result, it is better able to cope with the intermittent Internet connectivity caused by mobility and therefore provides a more reliable mobile download service than existing solutions.

(2) Via simulations, we show that the CAC algorithm provides better QoS for mobile users than the FCFS and LF algorithms.

(3) The proposed MetroNet solution is simple and generalizable to other metro systems as long as the schedules are known and the services maintain good punctuality.

The remainder of this paper is organised as follows. Section 2 contains a review of disruption-tolerant vehicular networks. In Section 3, we describe the proposed MetroNet system in detail. Then, we evaluate MetroNet’s performance via simulations with the CAC, FCFS and LF algorithms in two real-world scenarios in Section 4. Section 5 contains some concluding remarks.

2. RELATED WORK

Disruption-tolerant networks [7] transmit data by exploiting intermittent network connectivity, instead of utilising an end-to-end network connection between the source and the destination. Because the network topology may change dynamically in a random manner, conventional routing protocols cannot be applied in a DTN because they assume the existence of end-to-end connectivity. Thus, routing and data forwarding in DTNs are major challenges, and a great deal of research [8–13] has focused on finding an effective routing protocol for data dissemination. In our research, we assume that trains only interact with one-hop stations, and the metro systems have regular operating schedules and fixed routes. Hence, routing is not an issue in the proposed solution.

With the rapid development of DTNs, vehicular communication is becoming an increasingly popular application [1, 2]. Ott and Kutscher developed the drive-thru Internet system [3, 4] to provide users in moving vehicles with wireless access via APs deployed along the roadside. The system reports real-world measurements recorded between a moving car with an external antenna and a roadside wireless local area network AP. They also proposed a transmission control protocol-based session [14] to provide end-to-end connections that allow large volumes of data to be downloaded under conditions of intermittent connectivity. A number of studies have analysed the performance of the drive-thru Internet system [15–18]. For example, Tan et al. [15, 16] developed an analytical model to characterise the average number of bytes downloaded by a vehicle as it passes through an AP’s coverage range. In [17], Xie et al. studied the association control problem of vehicular wireless access in the drive-thru Internet system and tried to improve the efficiency and fairness for all users. Kim et al. [19] also proposed an association control algorithm to minimize the frequency of handoffs in mobile devices. Moreover, Pogel [18] exploited the knowledge about system characteristics and node mobility to improve the DTN routing performance over urban public transport systems.

Several other measurement and feasibility studies of disruption-tolerant vehicular networks have been conducted [20–27], and various solutions have been proposed. In [27], Zhang et al. analysed the bus-to-bus contact traces to characterise the contact process between the buses and its impact on DTN routing performance, as well as to model bus-to-bus connectivity. The bus-to-bus contact trace is obtained from the UMass DieselNet, which is a DTN that runs on 40 buses in Amherst, Massachusetts. In [22], the authors deployed the Thedu system on DieselNet to enable web searches from a bus. Thedu is a proxy-based solution that collects search engine responses to queries submitted by users and pre-fetches pages that correspond to each response. The user then downloads the pages from the proxy when Internet connectivity is available. The Massachusetts Institute of Technology CarTel project [20, 25] investigated general architectures for vehicular sensor networks, and a subsequent project called Cabernet [23] studied the scenario in which vehicular users access the Internet via roadside wireless local area network APs. Hadaller et al. [24] conducted a comprehensive set of experiments and showed that vehicular opportunistic access can be improved significantly if environmental information is made available to the media access control layer and the transportation layer (e.g. transmission control protocol).

A number of studies focus on the data delivery scheme in delay-tolerant vehicular networks [22, 28, 29]. Balasubramanian et al. [22] proposed a web search service based on the Thedu system. They also present a web page priority algorithm to rank pre-fetched web pages so that users can download the most useful response first. Similar to [22], Pack et al. proposed two proxy-based data access algorithms, called PC-based poll-each-read and PC-based callback, for downloading data in mobile environments.
Our approach differs from PC-based poll-each-read and PC-based callback in that it stores pre-fetched data for download at subsequent stations on the train’s route rather than use a single proxy server because the latter may cause a bottleneck problem. In contrast to the proxy-based approach, Jeong et al. [28] proposed a trajectory-based statistical forwarding scheme for data delivery from WiFi APs to moving vehicles. Instead of utilising a proxy, the scheme forwards data to a target point that the vehicle is expected to pass. Unlike vehicles travelling on roads, trains in metro systems have regular routes, so the data is forwarded to the next station on the route; hence, data delivery is more straightforward.

As vehicular networks become more widespread, an increasing number of users will want to access data from vehicles, such as cars, buses and trains. However, if a large number of users or vehicles attempt to access data through roadside devices (e.g., APs), service scheduling becomes an important issue. Several studies have investigated the problem [30–32] and proposed scheduling schemes that consider both the service deadline and the amount (size) of the data when making scheduling decisions. To control the QoS provided to vehicles, Luan et al. [33] developed an analytical model that considers the features of 802.11p enhanced distributed channel access (EDCA) and vehicle mobility and then adjusted the parameters of EDCA based on the model. The proposed MetroNet system differs from existing schemes as it provides three types of network service. In addition, to guarantee the QoS, it utilises a CAC mechanism and different scheduling priorities.

Several studies have focused on aerial-carrier-based or car-based or bus-based vehicular networks in which the vehicles act as data carriers [34–37]. For example, in [36], a train is regarded as a backhaul for transporting data. In terms of vehicular networks, the advantage of trains over cars or buses is that they travel on fixed tracks according to predefined timetables; hence, their movements are deterministic rather than random. In addition, the aerial carriers are also used for message delivery between airports based upon scheduled flight connections in [34]. In [36], the authors proposed the TrainNet architecture, a delay tolerant vehicular network, in which trains and stations have mass storage devices, and the trains are used to transfer data between stations. The authors investigated the resource scheduling problem because the capacity of the hard disks is finite, and only a fixed number of hard disks can be loaded and unloaded at each station. The goal is to ensure that the space on the hard disks is divided fairly among competing stations. In [37], the authors designed a DTN-based network that utilises trains as routers to connect a remote village to Internet network services, such as news portal and email services. Thus, users at the station can access news portal or email service through a workstation provided at the station. In this paper, we consider metro systems, which also have regular operating schedules. More specifically, we focus on providing a mobile download service that enables users to access pre-fetched video data from stations and watch videos without interruption as they travel between stations. The reason why we refer to the DTN concept is that the Internet connections between trains and stations are intermittent. Thus, the MetroNet system should pre-fetch enough data for uses in trains. When a train arrives at a station, the station transmits the related data to the train. Then, the data is forwarded to the end users as if the train is a hotspot. This is quite a different idea from the aforementioned work [34–37].

3. THE PROPOSED APPROACH

In this section, we introduce the proposed MetroNet system, which facilitates mobile downloads on metro systems affected by intermittent Internet connectivity. The system combines the strengths of the mobile hotspot approach with the concept of disruption-tolerant networks. It provides a more reliable mobile download service than existing solutions because it is better able to cope with the intermittent Internet connectivity caused by mobility. We describe the MetroNet system in detail in the following subsection.

3.1. MetroNet architecture

The system architecture of the MetroNet system is comprised of three components: the user, the train and the station, as shown in Figure 1. The user component represents passengers who require network connectivity for Internet applications (e.g., delayed-live video broadcasting, streaming and file downloads). The train and station components represent the core of the MetroNet system because they must provide passengers with network connectivity transparently by collaborating closely on a number of functionalities, such as CAC, traffic shaping, data caching, relaying requests and predicting mobility.

The train component contains four modules, namely, the CAC module, the request pool, the traffic shaper and the content cache. The CAC module determines whether a user’s request (REQ) for data download should be
accepted on the basis of the following factors: the bandwidth requirement, the available network bandwidth and the number of active users in the system. If a train accepts a REQ, it adds the REQ to its Request Pool and forwards it to the next station on the schedule. When a station receives a REQ, it stores the REQ in its Request Pool immediately and uses its Request Dispatcher to calculate the number of data bytes that it has to download to satisfy the REQ. Note that the calculation process relies to a large extent on the CAC algorithm and the different types of network traffic, which we will discuss in detail in the following subsections. If the Request Dispatcher only downloads a portion of the data bytes for the received REQ, it will initiate new REqs for the remaining data bytes and forward them to the subsequent stations immediately via the Internet. The forwarded REqs are then added to the Request Pools of the respective stations so that the requested data bytes can be pre-fetched before the train arrives (i.e. the pre-fetched data is stored in each station’s data storage).

When a train arrives at a station that has pre-fetched a requested data stream, the station transmits the stream to the train’s content cache* immediately (e.g. via wireless communication or by simply exchanging the disks). Then, the data stream is forwarded to the end user via the train’s traffic shaper module, which enforces the bandwidth usage policy in cooperation with the CAC module to provide good QoS for the end user.

### 3.2. MetroNet system messages

The MetroNet system relies a great deal on frequent exchanges of system messages. Suppose, there are $U$ users, $K$ trains and $S$ metro stations in the system. We denote the $i$-th user as $u_i$, the $j$-th train as $k_j$, the $m$-th station as $s_m$ and the $n$-th requested data stream as $d_n$. The stream can be a data file, a video clip or a delayed-live video broadcasting. Because the proposed system allows a relay download (i.e. the station can download a certain number of contiguous data bytes from any byte position of a data stream), we denote the start and end positions (e.g. time or byte positions) of the requested portion of a data stream as $d_n^s$ and $d_n^e$, respectively. The payload of the requested portion that starts at $d_n^s$ and ends at $d_n^e$ is denoted as $P(d_n^s, d_n^e)$. Next, we define the system messages utilised in MetroNet:

- **MSG ARRIVE**($k_j$), sent periodically in a broadcast fashion by a train $k_j$ to find a station that is within the wireless transmission range and that $k_j$ will visit. The message is only sent when $k_j$ is moving, and it does not have an active connection with any stations.
- **ACK ARRIVE**($k_j$, $s_m$), sent by the station $s_m$ to acknowledge receipt of the message **MSG ARRIVE**($k_j$).

---

*The size of train’s content cache as well as the size of station’s data storage is not a big issue because the storage is cheap. Furthermore, the traffic is hardly infinite large because it depends on the number of passengers and the travel time between stations.
train \( k_j \) updates the data request of the user \( u_i \) with a new starting position \( d_{s_0} \) for the data \( d_n \) in its request pool if the whole requested data has not been downloaded completely. Then, when the train approaches to the next station, the \( \text{REQ\_DATA\_TRAIN} \) could be forwarded again. If the train could not receive the \( \text{ACK\_DATA\_USER} \) within a period of time, it implies that the users have got off the train and does not need the requested data anymore. Thus, the downloaded data in the train’s content cache could be deleted immediately or be cached for a period of time depending on the type of requested data stream. In the meantime, the train could send an explicit \( \text{CLR\_REQ\_DATA\_TRAIN} \) message to the next station to clear the unnecessary pre-fetched data in station’s data storage.†

3.3. MetroNet services

MetroNet provides three types of network service, namely, delayed-live video broadcasting, video clip streaming and data file downloading. To exploit the network resource fully and maintain a good network experience, we designed the \( \text{Accept\_REQ} \) algorithm (Algorithm 1), which implemented when a data stream is requested. The algorithm utilises the CAC module to decide whether to accept a request for a data stream \( d_n \). When making the decision, the CAC function considers several network factors, such as the type of the requested data stream, the available bandwidth and the number of active users in the system. We discuss the CAC function further in the next section.

If a request is accepted, the system applies another algorithm called \( \text{request dispatcher} \), which is designed to implement the three network services, that is, delayed-live video broadcasting, video clip streaming and data file downloading. We describe the three variations of the algorithm in the following subsections.

3.3.1. Delayed-live video broadcasting.

Delayed-live video broadcasting means that MetroNet streams the requested video in a delayed-live fashion, that is, the video is streamed in real-time with a constant delay, \( T_{\text{delay}} \), instead of live broadcasting. The length of \( T_{\text{delay}} \) is based on the maximum time required for a train to...
move between two adjacent stations. More precisely, let $T(s_i, s_j)$ denote the time required for a train to move from station $s_i$ to station $s_j$. Then, $T_{delay}$ is derived as follows:

$$T_{delay} = \max(T(s_i, s_{i+1})); \text{ for } 1 \leq i < S - 1. \quad (1)$$

Let $A(k_j, s_m)$ denote the arrival time of the train $k_j$ at the station $s_m$, and $T_{stay}(s_m)$ denote the length of time the train $k_j$ stays at $s_m$. The relay algorithms for the delayed-live video broadcasting service are detailed in Algorithms 1 and 2. They calculate the number of data bytes that they have to download to satisfy the REQ and then download the requested data to their data storage units.

### 3.3.2. Video clip streaming.

For video clip streaming (e.g., from YouTube), MetroNet pre-fetches the minimum length of the requested video clip, which is identical to travel time to the next station. Note that there is no need to pre-fetch more than this volume because the user will not have time to play it before the train reaches the next station. To determine the pre-fetched length, the algorithm uses $vbr$ and $vlen$ to derive the bit rate and total length of a video clip respectively. The relay algorithms used to stream video clip traffic are detailed in Algorithms 6 and 7.

### 3.3.3. FTP traffic downloads.

To download FTP traffic, which has the lowest priority, MetroNet only needs to transmit some bytes of traffic to end users, as shown in Algorithms 6 and 7.

### 3.4. Call admission control for MetroNet
Algorithm 8 The algorithm executed by a train $k_j$ to determine whether a request for downloading data stream $d_s$ should be accepted.

1: Function CAC(type, $n_{video}$, abw)
2: 
3: if type = delayed live stream then
4: 
5: return ACCEPT
6: 
7: else if type = video stream then
8: 
9: for $x = m + 1$ to $S$ do
10: 
11: calculate $abw(s_j)$
12: 
13: if $abw(s_j) < 0$ then
14: 
15: return REJECT
16: 
17: end if
18: 
19: end for
20: 
21: return ACCEPT
22: 
23: end if

The objective of the CAC function is to maintain QoS for end users. To this end, the CAC algorithm determines whether a user’s request for data download should be accepted on the basis of the following factors: the bandwidth requirement, the available network bandwidth and the number of active sessions in the system. Delayed-live video broadcasting traffic has the highest priority traffic, so all requests for the service are accepted by default. For video clip streaming traffic, the available bandwidth between the train and the stations must be calculated first. Let $R_{delayed-live}$ and $R_{video}$ denote the bit rate of a delayed-live stream and a video clip stream, respectively, and let $n_{video}$ be the number of requests accepted for the video clip stream. Then, the available bandwidth $abw(s_m)$ at the metro station $s_m$ is derived by Equation 2, where $BTS$ denotes the bandwidth between the train and the station, and $T_{stay}(s_m)$ denote the length of time the train stays at $s_m$. When the train $k_j$ receives a REQ, the CAC algorithm checks all the subsequent stations to determine if the available bandwidth is sufficient to transmit the video clip stream. If one of the stations cannot satisfy the bandwidth requirement, the REQ will be rejected. Note that FTP download traffic will only be served if there is some bandwidth left. The steps of the CAC algorithm are detailed in Algorithm 8.

4. EVALUATIONS

We evaluate the system performance in terms of the number of accepted requests, the session time and the system gain. Specifically, we implement the CAC algorithm to compare its performance with the other two classic scheduling algorithms, namely, the FCFS and the LF algorithms and perform emulations on Kasuari [38]. Kasuari is a Xen-based emulation framework that uses NS-2 [39] to simulate a network environment. In addition, we consider two real-world metro systems: the MRT system in Taipei [5] and the Metropolitan Atlanta Rapid Transit Authority system in Atlanta [6]. Figure 3(a) shows the route map of Taipei’s southbound Danshui-Xindian Line from Danshuei station to Xindian station; whereas Figure 3(b) shows the map of Atlanta’s northbound Gold Line from Airport station to Doraville station. The travel times between stations are also marked, and they are obtained by the time difference between the two departure times using the timetables of the two metro systems. Thus, each travel time value comprises of both the running time (i.e., the duration that the train is moving) and the dwell time (i.e., the duration that the train stays at a station). For simplicity, we assume the latter, that is, $T_{stay}$, is fixed at 30 s in this study.

In addition, we assume that the bandwidth between trains and users is 10 Mbps, and that between trains and stations, $B_{TS}$ is also 10 Mbps. A network connection is available at each station, and the bandwidth between stations and content sources is infinite. The bit rates of delayed-live video broadcasting and video clip streaming traffic are 350 and 160 Kbps, respectively, and the length of a video varies from 1 to 20 min. The size of FTP downloads varies from 1 to 5 MB. The occurrence of requests follows a Poisson distribution; and $\lambda$ represents the number of requests for delayed-live video broadcasting, video clip streaming and FTP downloading as a train travels between two neighboring stations. For example, $\lambda = 1$ means there is one delayed-live video broadcasting request, one video clip streaming request and one FTP download request on average.

Let $N$ be the average number of accepted requests, and let $T$ be the average session time. The system gain, $G$, is defined in Equation 3. The metric $G$ is designed to evaluate how well the proposed MetroNet system accommodates both performance parameters simultaneously. Clearly, the fast development of wireless communication technologies enables high-speed data transmission between trains and stations. Thus, the dwell time of a train at a station would be long enough to transmit the content in the data storage of the station to the content cache of the train. The simple disk exchanges is also a possible solution for data transfer.

In real-world scenarios, the lambda value may change dynamically following an exponential distribution or a power-law with long tail distribution. However, because it is impossible to enumerate all possible configurations of the lambda values, we decide to use fixed settings in this study because it allows us to observe the system performance in detail under different lambda values. Note that we use the natural logarithmic scale for both factors in $G$ because it has been shown that the logarithmic scale is more intuitive and appropriate in number-space mapping [40]. Moreover, the natural logarithmic scale has several properties (e.g., derivatives and Taylor series) that could be useful for further analysis.
Figure 3. The network scenarios used in the evaluation: (a) Taipei Scenario and (b) Atlanta Scenario.

Figure 4. The system gain in the two metro scenarios when Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1. (a) Taipei scenario and (b) Atlanta scenario.

The system gain increases as both the number of accepted requests and the session time increase. However, with a limited network resource, we may accept more requests but with few session time. On the other hand, we may
accept less requests but with longer session time. There is a tradeoff between the number of accepted requests and the session time.

\[ G = \ln(N) \times \ln(T). \] (3)

### 4.1. Evaluation I: Baseline

Firstly, we consider the baseline results of the system performance in the two network scenarios, where the ratio of delayed-live video broadcasting, video clip streaming and FTP downloading requests is 1:1:1, as shown in Figure 4.

![Graphs showing the number of accepted requests under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1.](a) CAC, Taipei scenario, (b) FCFS, Taipei scenario, (c) LF, Taipei scenario, (d) CAC, Atlanta scenario, (e) FCFS, Atlanta scenario, (f) LF, Atlanta scenario.

Figure 5. The number of accepted requests under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1. (a) call admission control, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (c) least-first, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario, (f) least-first, Atlanta scenario.

![Graphs showing the session times under the three algorithms in the two metro scenarios when the Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1.](a) CAC, Taipei scenario, (b) FCFS, Taipei scenario, (c) LF, Taipei scenario, (d) CAC, Atlanta scenario, (e) FCFS, Atlanta scenario, (f) LF, Atlanta scenario.

Figure 6. The session times under the three algorithms in the two metro scenarios when the Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1. (a) call admission control, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (c) least-first, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario, (f) least-first, Atlanta scenario.
When $\lambda$ is less than 4, the performance of the three algorithms is similar because the bandwidth between trains and stations is sufficient to handle all requests; that is, no requests are rejected. However, as $\lambda$ increases, the FCFS algorithm outperforms the LF algorithm. This is because LF tends to accept requests for smaller amounts of content, so it rejects delayed-live video broadcasting requests of indefinite length when the bandwidth is insufficient, as shown in Figures 5(c) and 5(f). In addition, we assume that only one delayed-live video broadcasting stream is provided, which means no additional bandwidth is required after one delayed-live video broadcasting request has been accepted. Because the number of requests for the three services is the same in this simulation, FCFS accepts more streaming requests than LF; thus, it achieves a higher system gain. The performance of the proposed CAC algorithm is always equal to, or better than, that of the FCFS and LF algorithms.

The average session times are shown in Figure 6. As the value of $\lambda$ increases, the average session time of video clip streaming decreases, whereas that of FTP downloading increases. The reason is that the available bandwidth

![Figure 7](image-url)

Figure 7. The system gain in the two metro scenarios when Internet connectivity is only available at every other station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1. (a) Taipei scenario and (b) Atlanta scenario.

![Figure 8](image-url)

Figure 8. The number of accepted requests under the three algorithms in the two metro scenarios. Internet connectivity is only available at every other station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1. (a) call admission control, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (c) least-first, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario and (f) least-first, Atlanta scenario.
decreases as the number of accepted requests increases. Thus, requests for shorter video clips have a higher probability of being accepted. On the other hand, for FTP traffic, the smaller the amount of available bandwidth, the longer it will take to complete the download; thus, the session time will be longer.

The results of the Atlanta scenario are similar to those of the Taipei MRT, as shown in Figures 5(d)–(f) and Figures 6(d)–(f). A slight difference is that the number of accepted requests is about 2/3 that in the Taipei scenario because the Atlanta line (i.e. 19 stations) is only 2/3 the size of the Taipei line (i.e. 30 stations). Furthermore, the average travel time between stations on the Atlanta line is longer than that on the Taipei line. Consequently, when a train stops at a station, more bandwidth is needed to download the pre-fetched data, so the number of accepted requests would be limited.

### 4.2. Evaluation II: Availability of network connections

In this evaluation, a network connection is only available at every other station. The other parameter settings are the same as those used in Figure 4. Figure 7 shows that even if the network availability deteriorates, the proposed
Figure 11. The number of accepted requests under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 5:1:1. (a) call admission control, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (c) least-first, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario and (f) least-first, Atlanta scenario.

Figure 12. The session times under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 5:1:1. (a) call admission control, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (c) least-first, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario and (f) least-first, Atlanta scenario.
CAC algorithm still outperforms the FCFS and LF algorithms in both network scenarios. However, as shown in Figure 8(a), in the Taipei scenario, the CAC algorithm accepts fewer video clip streaming requests than in the scenario where Internet connectivity is available at each station (Figure 5(a)). This is because the algorithm needs more bandwidth to download delayed-live video broadcasting traffic to trains when the network connection is only available at every other station. Moreover, because the bandwidth between trains and stations is limited, there may not be enough bandwidth to transmit video clip streaming traffic. The session times of accepted video clip streaming requests also tend to be shorter, as shown in Figure 9(a).

It is unlikely that requests for longer video clips would be accepted because the amount of available bandwidth decreases as \( \lambda \) increases. Under the FCFS algorithm, the number of accepted video clip streaming and FTP download requests is lower than under the baseline scenario because there is less bandwidth overall. Meanwhile, under the LF algorithm, requests for delayed-live video broadcasting are rarely accepted because more bandwidth is needed between trains and stations to transmit pre-fetched data, as shown in Figure 8(c). The results of the number of accepted requests and the session times for the

Figure 13. The system gain in the two metro scenarios when Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:5:1. (a) Taipei scenario and (b) Atlanta scenario.

Figure 14. The number of accepted requests under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:5:1. (a) call admission control, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (c) least-first, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario and (f) least-first, Atlanta scenario.
Atlanta scenario are similar, as shown in Figures 8(d)–(f) and Figures 9(d)–(f).

4.3. Evaluation III: Diversity of the types of requests

In the previous two evaluations, all requests for the different services had the same distribution because the ratio of delayed-live video broadcasting, video clip streaming and FTP downloading requests was 1:1:1. However, in a real-world scenario, the ratio depends on the type of service. In this simulation, we consider three cases where the ratios of delayed-live video broadcasting, video clip streaming and FTP downloading requests are 5:1:1, 1:5:1 and 1:1:5, respectively. The other parameter settings are the same as those in the baseline scenario.

Figures 10, 11 and 12 show the simulation results when the ratio of the three services is 5:1:1. The performance results for video clip streaming, and FTP downloading are similar to results in Figures 5 and 6. The only difference is that the number of accepted requests for delayed-live video

Figure 15. The session times under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:5:1. (a) call admission control, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (c) least-first, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario and (f) least-first, Atlanta scenario.

Figure 16. The system gain in the two metro scenarios when Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:5. (a) Taipei scenario and (b) Atlanta scenario.
Figure 17. The number of accepted requests under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:5. (a) call admission control, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (c) least-first, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario and (f) least-first, Atlanta scenario.

Figure 18. The session times under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming, and FTP downloading requests is 1:1:5. (a) call admission control, Taipei scenario, (c) least-first, Taipei scenario, (b) first-come-first-serve, Taipei scenario, (d) call admission control, Atlanta scenario, (e) first-come-first-serve, Atlanta scenario and (f) least-first, Atlanta scenario.
broadcasting is five times greater than that in the baseline scenario. This is because, once a delayed-live video broadcasting request has been accepted, it can be used to serve multiple users’ requests without allocating additional bandwidth. In terms of system gain, the performance of the LF algorithm is not as good as that of the CAC and FCFS algorithms because the number of requests that can be accepted for delayed-live video broadcasting is limited.

The simulation results for delayed-live video broadcasting, video clip streaming and FTP downloading requests when the ratio is 1:5:1 are shown in Figures 13, 14 and 15, respectively. Under the CAC algorithm, the number of requests accepted for video clip streaming is much higher than that in the baseline scenario (i.e. where the ratio is 1:1:1), but the average session time is shorter because of the limited bandwidth. Compared with the results in Figure 5(b), the FCFS algorithm accepts more video clip streaming requests because the number of requests is much higher than for delayed-live video broadcasting and FTP downloading requests. However, as the algorithm requires more bandwidth, fewer requests are accepted for delayed-live video broadcasting and FTP downloading. The results in Figures 14(c) and 15(c) show that the LF algorithm tends to accept requests for video clips that vary in size from 1.2 MB (i.e. 160 Kbps × 1 min) to 24 MB (i.e. 160 Kbps × 20 min), rather than requests for delayed-live video broadcasting. Thus, if the number of accepted requests for video clip streaming increases, the available bandwidth may not be enough to accept any requests for delayed-live video broadcasting of indefinite length. As a result, the system gain of the CAC algorithm is much better than that of the LF algorithm.

Figures 16, 17 and 18 show the performance results when the ratio of delayed-live video broadcasting, video clip streaming and FTP downloading requests is 1:1:5. The number of accepted requests for FTP downloading is higher than in the baseline scenario (Figures 5(b) and 17(b)). This is because, under the FCFS algorithm, the larger the number of requests, the higher will be the acceptance rate. However, when delayed-live video broadcasting and video clip streaming requests are accepted, there may not be any bandwidth left for FTP downloading requests; therefore, many FTP requests may be rejected. Under the LF algorithm, the number of accepted FTP download requests is much higher than that for delayed-live video broadcasting and video clip streaming. The reason is that the LF algorithm favours FTP download files that are smaller (in terms of length) than delayed-live video broadcasting and video clip streaming files.

5. CONCLUSION

We have proposed a system called MetroNet, which combines the strengths of the mobile hotspot approach and the concept of disruption-tolerant networks, to facilitate mobile downloads on metro systems affected by intermittent Internet connectivity. MetroNet exploits the reliability of metro system schedules to pre-fetch data and facilitate data transmission, even if an end-to-end path between the source and the destination does not exist contemporaneously. We evaluate MetroNet via a comprehensive set of simulations and compare the performance of the proposed CAC algorithm with that of the FCFS and LF algorithms in two real-world scenarios. The results show that the CAC algorithm outperforms the FCFS and LF algorithms in all test cases. Moreover, the results demonstrate that the CAC algorithm provides a better network experience for mobile users when Internet connectivity is poor under different ratios of the three services. The proposed system is simple and can be deployed easily in other metro systems, as long as the schedules are known and punctuality is maintained.

ACKNOWLEDGEMENTS

This research was supported by the National Science Council of Taiwan and Academia Sinica under grant numbers: NSC 102-2219-E-001-002, NSC 101-2628-E-001-004-MY3 and AS-102-TP-A06.

REFERENCES


Dear Author,

During the copyediting of your paper, the following queries arose. Please respond to these by annotating your proofs with the necessary changes/additions.

- If you intend to annotate your proof electronically, please refer to the E-annotation guidelines.
- If you intend to annotate your proof by means of hard-copy mark-up, please refer to the proof mark-up symbols guidelines. If manually writing corrections on your proof and returning it by fax, do not write too close to the edge of the paper. Please remember that illegible mark-ups may delay publication.

Whether you opt for hard-copy or electronic annotation of your proofs, we recommend that you provide additional clarification of answers to queries by entering your answers on the query sheet, in addition to the text mark-up.

---

### Query Form

**Journal:** Transactions on Emerging Telecommunications Technologies  
**Article:** ett_2695

<table>
<thead>
<tr>
<th>Query No.</th>
<th>Query</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>AUTHOR: Please provide a suitable figure (abstract diagram or illustration selected from the manuscript or an additional eye-catching figure) and a short ‘GTOC’ abstract (maximum 80 words or 3 sentences) summarising the key findings presented in the paper for Table of Content (TOC) entry.</td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td>AUTHOR: Is this the correct definition for APs? Please change if this is incorrect.</td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>AUTHOR: Are these the correct definitions of MRT and MARTA? Please change if this is incorrect.</td>
<td></td>
</tr>
<tr>
<td>Q4</td>
<td>AUTHOR: Is this the correct definition of WLAN? Please change if this is incorrect.</td>
<td></td>
</tr>
<tr>
<td>Q5</td>
<td>AUTHOR: Is this the correct definition of TCP? Please change if this is incorrect.</td>
<td></td>
</tr>
<tr>
<td>Q6</td>
<td>AUTHOR: Is this the correct definition of MIT? Please change if this is incorrect.</td>
<td></td>
</tr>
<tr>
<td>Q7</td>
<td>AUTHOR: Is this the correct definition of MAC? Please change if this is incorrect.</td>
<td></td>
</tr>
<tr>
<td>Q8</td>
<td>AUTHOR: Are these the correct definitions of P-PER and P-CB? Please change if this is incorrect.</td>
<td></td>
</tr>
<tr>
<td>Q9</td>
<td>AUTHOR: You have provided all figures in colour. Because of the high cost of colour printing we can only print the figures in colour if authors cover the expense. However, the figures will be published online in colour free of cost. Please confirm if you would like the figure(s) to be printed in colour.</td>
<td></td>
</tr>
<tr>
<td>Q10</td>
<td>AUTHOR: Please provide location and page number in Reference 3.</td>
<td></td>
</tr>
<tr>
<td>Q11</td>
<td>AUTHOR: Please provide accessed date for Reference 5, 6, 38, 39.</td>
<td></td>
</tr>
<tr>
<td>Q12</td>
<td>AUTHOR: Please provide location of the publisher for Reference 10, 14, 18-28, 31, 35, 37.</td>
<td></td>
</tr>
<tr>
<td>Q13</td>
<td>AUTHOR: If this reference has now been published in print, please add relevant year information.</td>
<td></td>
</tr>
<tr>
<td>Q14</td>
<td>AUTHOR: If this reference has now been published in print, please add relevant volume/issue/page information.</td>
<td></td>
</tr>
</tbody>
</table>
USING e-ANNOTATION TOOLS FOR ELECTRONIC PROOF CORRECTION

Required software to e-Annotate PDFs: Adobe Acrobat Professional or Adobe Reader (version 7.0 or above). (Note that this document uses screenshots from Adobe Reader X)

The latest version of Acrobat Reader can be downloaded for free at: http://get.adobe.com/uk/reader/

Once you have Acrobat Reader open on your computer, click on the Comment tab at the right of the toolbar:

This will open up a panel down the right side of the document. The majority of tools you will use for annotating your proof will be in the Annotations section, pictured opposite. We’ve picked out some of these tools below:

1. Replace (Ins) Tool – for replacing text.
   
   Strikes a line through text and opens up a text box where replacement text can be entered.

   How to use it
   - Highlight a word or sentence.
   - Click on the Replace (Ins) icon in the Annotations section.
   - Type the replacement text into the blue box that appears.

2. Strikethrough (Del) Tool – for deleting text.

   Strikes a red line through text that is to be deleted.

   How to use it
   - Highlight a word or sentence.
   - Click on the Strikethrough (Del) icon in the Annotations section.

3. Add note to text Tool – for highlighting a section to be changed to bold or italic.

   Highlights text in yellow and opens up a text box where comments can be entered.

   How to use it
   - Highlight the relevant section of text.
   - Click on the Add note to text icon in the Annotations section.
   - Type instruction on what should be changed regarding the text into the yellow box that appears.

4. Add sticky note Tool – for making notes at specific points in the text.

   Marks a point in the proof where a comment needs to be highlighted.

   How to use it
   - Click on the Add sticky note icon in the Annotations section.
   - Click at the point in the proof where the comment should be inserted.
   - Type the comment into the yellow box that appears.
5. **Attach File Tool** – for inserting large amounts of text or replacement figures.

Inserts an icon linking to the attached file in the appropriate pace in the text.

**How to use it**
- Click on the **Attach File** icon in the Annotations section.
- Click on the proof to where you’d like the attached file to be linked.
- Select the file to be attached from your computer or network.
- Select the colour and type of icon that will appear in the proof. Click OK.

6. **Add stamp Tool** – for approving a proof if no corrections are required.

Inserts a selected stamp onto an appropriate place in the proof.

**How to use it**
- Click on the **Add stamp** icon in the Annotations section.
- Select the stamp you want to use. (The **Approved** stamp is usually available directly in the menu that appears).
- Click on the proof where you’d like the stamp to appear. (Where a proof is to be approved as it is, this would normally be on the first page).

7. **Drawing Markups Tools** – for drawing shapes, lines and freeform annotations on proofs and commenting on these marks.

Allows shapes, lines and freeform annotations to be drawn on proofs and for comment to be made on these marks.

**How to use it**
- Click on one of the shapes in the **Drawing Markups** section.
- Click on the proof at the relevant point and draw the selected shape with the cursor.
- To add a comment to the drawn shape, move the cursor over the shape until an arrowhead appears.
- Double click on the shape and type any text in the red box that appears.

For further information on how to annotate proofs, click on the **Help** menu to reveal a list of further options: