TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 2014; **25**:835–851
Published online 30 September 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/ett.2695

SPECIAL ISSUE PAPER

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

Li-Ping Tung[1], Yi-Chao Chen[2], Kun-chan Lan[3] and Ling-Jyh Chen[4]*

[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 utilises the network resource effectively and provides better quality of service for mobile users than the first-come-first-serve and least-first algorithms.

***Correspondence**
L.-J. Chen, Institute of Information Science, Academia Sinica, Taiwan.
E-mail: ccljj@iis.sinica.edu.tw

Received 15 March 2013; Revised 12 June 2013; Accepted 3 July 2013

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 Wi-Fi, 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 WiFi AP. They also proposed a TCP-based section protocol [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 and tried to improve 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 WiFi 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 (P-PER) and PC-based callback (P-CB), for downloading data in mobile environments [29]. Our approach differs from P-PER and P-CB in that it stores pre-fetched data for download at sub-

DOI: 10.1002/ett

sequent 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 prefetched 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

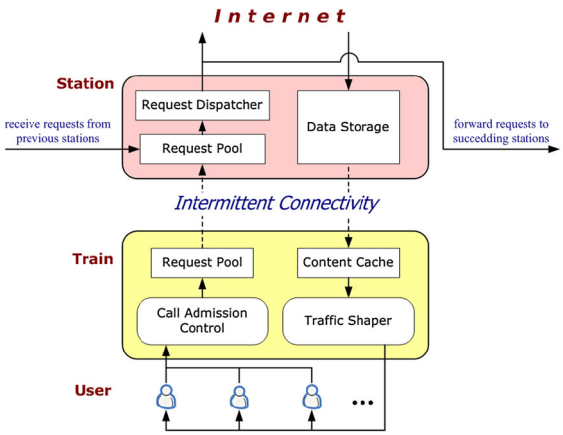


Figure 1. The system architecture of MetroNet.

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 Pool*s 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 *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 $MSG\_ARRIVE(k_j)$.
- $REQ\_DATA\_USER(u_i, k_j, d_n)$: sent by a user u_i to the train k_j to request the data stream d_n.
- $ACK\_USER\_REQ(u_i, k_j, d_n)$: sent by the train k_j to the user u_i to acknowledge receipt of the message $REQ\_DATA\_USER(u_i, k_j, d_n)$.
- $REQ\_DATA\_TRAIN(k_j, s_m, d_n, d_n^s)$: sent by the train k_j to the current connected station s_m to request the data stream d_n starting at position d_n^s.
- $ACK\_TRAIN\_REQ(k_j, s_m, d_n)$: sent by the station s_m to the train k_j to acknowledge receipt of the message $REQ\_DATA\_TRAIN(k_j, s_m, d_n, d_n^s)$.
- $REQ\_DATA\_STATION(s_m, s_{m'}, d_n, d_n^s)$: sent by station s_m to the station $s_{m'}$ to request the data stream d_n starting at position d_n^s.
- $ACK\_STATION\_REQ(s_m, s_{m'}, d_n)$: sent by the station $s_{m'}$ to the station s_m to acknowledge receipt of the message $REQ\_DATA\_STATION(s_m, s_{m'}, d_n, d_n^s)$.
- $DATA\_TRAIN(k_j, s_m, d_n, d_n^s, d_n^e, P(d_n^s, d_n^e))$: sent by the station s_m to the train k_j. It contains the payload $P(d_n^s, d_n^e)$ of the data stream d_n that starts at d_n^s and ends at d_n^e.
- $ACK\_DATA\_TRAIN(k_j, s_m, d_n, d_n^s, d_n^e)$: sent by the train k_j to the station s_m to acknowledge receipt of the message $DATA\_TRAIN(k_j, s_m, d_n, d_n^s, d_n^e, P(d_n^s, d_n^e))$.
- $DATA\_USER(u_i, k_j, d_n, d_n^s, d_n^e, P(d_n^s, d_n^e))$: sent by the train k_j to the user u_i. It contains the payload $P(d_n^s, d_n^e)$ of the data stream d_n that starts at d_n^s and ends at d_n^e.
- $ACK\_DATA\_USER(u_i, k_j, d_n, d_n^s, d_n^e)$: sent by the user u_i to the train k_j to acknowledge receipt of the message $DATA\_USER(u_i, k_j, d_n, d_n^s, d_n^e, P(d_n^s, d_n^e))$.

Figure 2 shows an example of message exchange in the MetroNet system. In this example, when the train k_j receives a request for the data stream d_n from the user u_i (i.e. REQ_DATA_USER), it acknowledges the request immediately with ACK_USER_REQ. The user u_i then waits for data stream d_n.

The train k_j broadcasts MSG_ARRIVE periodically until it receives ACK_ARRIVE from the station (say s_m) it is approaching. Then, k_j forwards the data requests from its request pool to s_m using REQ_DATA_TRAIN, which s_m will acknowledge immediately with an ACK_TRAIN_REQ. The station then checks its local data storage and responds to the request in one of the following ways: (i) if s_m has the amount of data required to serve the user in the travel time from s_m to $s_{m'}$, it transmits the data with $DATA\_TRAIN$ to k_j; or (ii) if s_m can only provide part of the data, it sends a $REQ\_DATA\_STATION(s_m, s_{m'}, d_n, d_n^{s'})$ to $s_{m'}$ with a new starting position $d_n^{s'}$ for the data d_n. On receipt of the data d_n, the train k_j transmits it to the user u_i (i.e. $DATA\_USER$), who then confirms the transmission has been successful by sending ACK_DATA_USER to the train. Once receiving the ACK_DATA_USER, the

*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.

Trans. Emerging Tel. Tech. **25**:835–851 (2014) © 2013 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

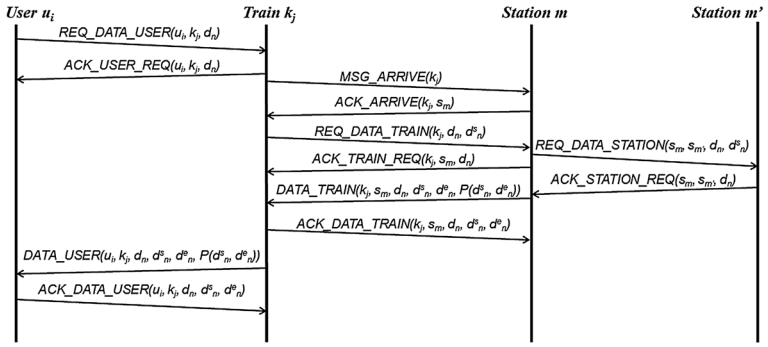


Figure 2. An example of message exchange in the MetroNet system.

Algorithm 1 The algorithm implemented by the train k_j to determine whether to accept a request for downloading a data stream d_n.

1: **Function Accept_REQ**(d_n)
2: $type \Leftarrow Get\_REQ\_Type(d_n)$
3: $n_{video} \Leftarrow Get\_Active\_Video\_Users()$
4: $abw \Leftarrow Get\_Available\_Bandwidth()$
5: **return** $CAC(type, n_{video}, abw)$

train k_j updates the data request of the user u_i with a new starting position $d_n^{s'}$ 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 REQ_DATA_TRAIN could be forwarded again. If the train could not receive the 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 $CLR\_REQ\_DATA\_TRAIN$ 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 $Accept\_REQ$ algorithm (Algorithm 1), which implemented when a data stream is requested. The algorithm

Algorithm 2 The algorithm implemented by station s_m when it receives a REQ_DATA_TRAIN message for a delayed-live stream from a train k_j.

1: **Function Rcv_REQ_DATA_TRAIN**(k_j, s_m, d_n, d_n^s)
2: Send $ACK\_TRAIN\_REQ(k_j, s_m, d_n)$
3: **if** there is a piece of d_n starts at d_n^s and ends at d_n^e **then**
4: Send $DATA\_TRAIN(k_j, s_m, d_n, d_n^s, d_n^e, P(d_n^s, d_n^e))$
5: **end if**
6: **for** $x = m + 1$ **to** S **do**
7: **if** there are no pieces of d_n that start at d_n^s & $T(S_{m+1}, S_x) < T_{delay}$ **then**
8: $d_n^s = A(k_j, s_x) - T_{delay}$
9: Send $REQ\_DATA\_STATION(s_m, s_x, d_n, d_n^s)$
10: **else if** $T(s_m, s_x) \geqslant T_{delay}$ & $T(s_{m+1}, s_x) < T_{delay}$ **then**
11: $d_n^s = A(k_j, s_m) + T_{stay}(s_m) + T(s_m, s_x) - T_{delay}$
12: Send $REQ\_DATA\_STATION(s_m, s_x, d_n, d_n^s)$
13: **end if**
14: **end for**
15: **return**

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 *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_{delay}, instead of live broadcasting. The length of T_{delay} is based on the maximum time required for a train to

[†]The behaviour of getting off the train without finishing the data request does impact the available space on the data storage of the station and on the content cache of the train. However, because the price and size of storage devices have been both decreasing, we posit that this is not a big issue at all in the proposed system.

Algorithm 3 The algorithm implemented by station s_m when it receives a $REQ\_DATA\_STATION$ message for a delayed-live stream from station s_x.

Function Rcv_REQ_DATA_STATION(s_x, s_m, d_n, d_n^s)
Send $ACK\_STATION\_REQ(s_x, s_m, d_n)$
$d_n^e = d_n^s + T_{stay}(s_{m+1}) + T(s_m, s_{m+1})$
Download d_n from the position d_n^s to d_n^e
return

Algorithm 4 The algorithm implemented by station s_m when it receives a REQ_DATA_TRAIN message for a video clip stream from a train k_j.

Function Rcv_REQ_DATA_TRAIN(k_j, s_m, d_n, d_n^s)
Send $ACK\_TRAIN\_REQ(k_j, s_m, d_n)$
if there are no pieces of d_n that start at d_n^s **then**
  Send $REQ\_DATA\_STATION(s_m, s_{m+1}, d_n, d_n^s)$
else {there is a piece of d_n starts at d_n^s and ends at d_n^e}
  Send $DATA\_TRAIN(k_j, s_m, d_n, d_n^s, d_n^e, P(d_n^s, d_n^e))$
  $vlen \Leftarrow Get\_Video\_Length(d_n)$
  if $d_n^e < vlen$ **then**
    Send $REQ\_DATA\_STATION(s_m, s_{m+1}, d_n, d_n^e)$
  end if
end if
return

Algorithm 5 The algorithm implemented by station s_m when it receives a $REQ\_DATA\_STATION$ message for a video clip stream from station s_{m-1}.

Function Rcv_REQ_DATA_STATION(s_{m-1}, s_m, d_n, d_n^s)
Send $ACK\_STATION\_REQ(s_{m-1}, s_m, d_n)$
$vbr \Leftarrow Get\_Video\_Bitrate(d_n)$
$vlen \Leftarrow Get\_Video\_Length(d_n)$
if $vlen > d_n^s + vbr \times T(s_m, s_{m+1})$ **then**
  $d_n^e \Leftarrow d_n^s + vbr \times T(s_m, s_{m+1})$
  Send $REQ\_DATA\_STATION(s_m, s_{m+1}, d_n, d_n^e)$
else
  $d_n^e \Leftarrow vlen$
end if
Download d_n from position d_n^s to position d_n^e
return

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:

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

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 2 and 3. 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.

Algorithm 6 The algorithm implemented by station s_m when it receives a REQ_DATA_TRAIN message for a data file from a train k_j.

Function Rcv_REQ_DATA_TRAIN(k_j, s_m, d_n, d_n^s)
Send $ACK\_TRAIN\_REQ(k_j, s_m, d_n)$
if there are no pieces of d_n that start at d_n^s **then**
  Send $REQ\_DATA\_STATION(s_m, s_{m+1}, d_n, d_n^s)$
else {there is a piece of d_n starts at d_n^s and ends at d_n^e}
  Send $DATA\_TRAIN(k_j, s_m, d_n, d_n^s, d_n^e, P(d_n^s, d_n^e))$
  $flen \Leftarrow Get\_File\_Length(d_n)$
  if $d_n^e < flen$ **then**
    Send $REQ\_DATA\_STATION(s_m, s_{m+1}, d_n, d_n^e)$
  end if
end if
return

Algorithm 7 The algorithm implemented by station s_m when it receives a $REQ\_DATA\_STATION$ message for a data file from station s_{m-1}.

Function Rcv_REQ_DATA_STATION(s_{m-1}, s_m, d_n, d_n^s)
Send $ACK\_STATION\_REQ(s_{m-1}, s_m, d_n)$
$d_n^e \Leftarrow Get\_File\_Length(d_n)$
Download d_n from the position d_n^s to d_n^e
return

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 4 and 5.

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

$$abw(s_m) = B_{TS} - \frac{(R_{delayed-live} + n_{video} \times R_{video}) \times (T(s_m, s_{m+1}) + T_{stay}(s_{m+1}))}{T_{stay}(s_m)}. \quad (2)$$

Trans. Emerging Tel. Tech. **25**:835–851 (2014) © 2013 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

Algorithm 8 The algorithm executed by a train k_j to determine whether a request for downloading data stream d_n should be accepted.

Function CAC($type, n_{video}, abw$)
if $type$ = delayed live stream **then**
  return ACCEPT
else if $type$ = video stream **then**
  for $x = m + 1$ **to** S **do**
    calculate $abw(s_x)$
    if $abw(s_x) - \frac{R_{video} \times (T(s_x, s_{x+1}) + T_{stay}(s_{x+1}))}{T_{stay}(s_x)} < 0$ **then**
      return REJECT
    end if
  end for
  return ACCEPT
else {ftp downloading}
  for $x = m + 1$ **to** S **do**
    calculate $abw(s_x)$
    if $abw(s_x) < 0$ **then**
      return REJECT
    end if
  end for
  return ACCEPT
end if

The objective of the *CAC* function is to maintain good 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 B_{TS} 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 can not 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 Danshui 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 used. 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.

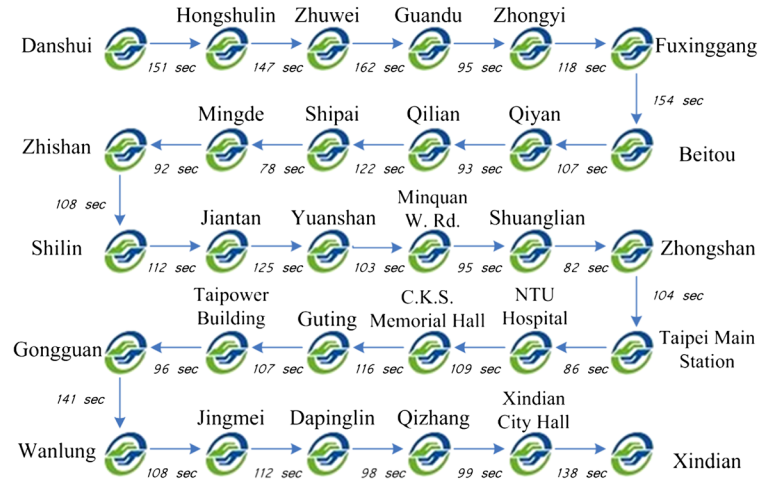
In addition, we assume that the bandwidth between trains and users is 10 Mbps, and that between trains and stations, B_{TS} is also 10 Mbps[‡]. A network connection is available at each station, and the bandwidth between stations and content sources is infinite. The bit rates of delayed-live video broadcasting 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 λ 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

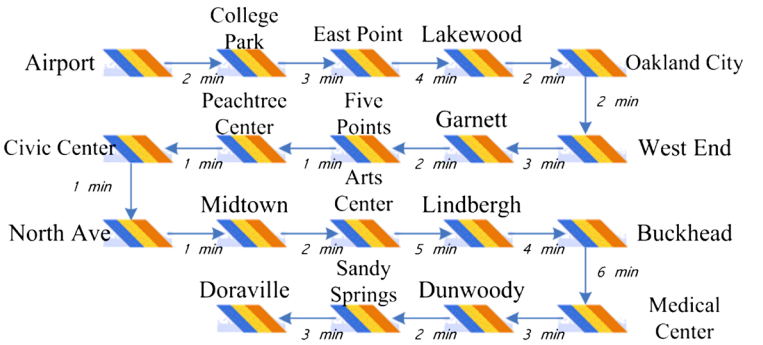
[‡]The fast development of wireless communication technologies enables high-speed data transmission between trains and stations. Thus, the dwell time of a train at a station would be long enough to transmit the content in the data storage of the station to the content cache of the train. The simple disk 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.



(a) Taipei Scenario



(b) Atlanta Scenario

Figure 3. The network scenarios used in the evaluation: (a) Taipei Scenario and (b) Atlanta Scenario.

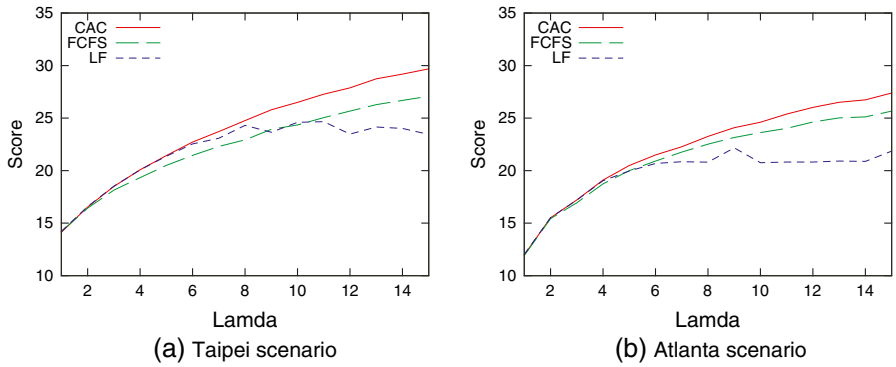


Figure 4. The system gain in the two metro scenarios when Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1.

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.

$$G = \ln(N) \times \ln(T). \tag{3}$$

Trans. Emerging Tel. Tech. **25**:835–851 (2014) © 2013 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

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. When λ 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 λ 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

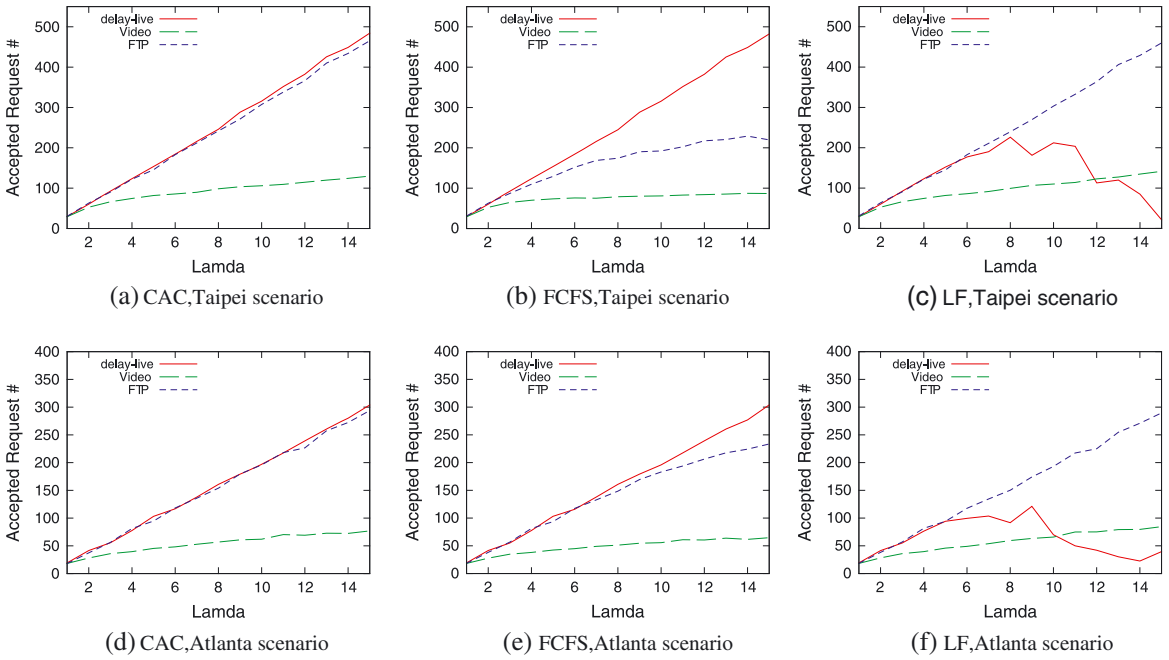


Figure 5. The number of accepted requests under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1.

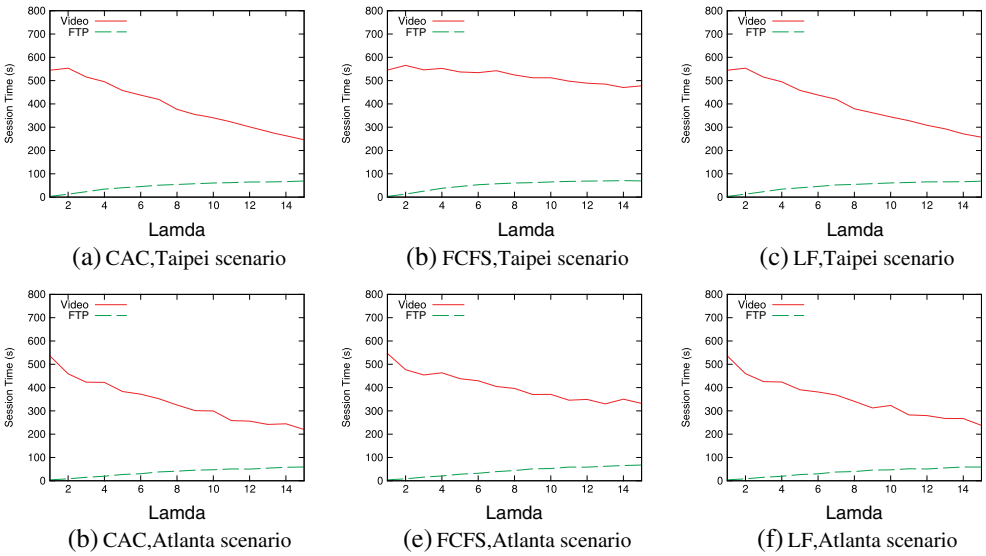


Figure 6. The session times under the three algorithms in the two metro scenarios when the Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1.

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 λ increases, the average session time of video clip streaming decreases, whereas that of FTP downloading increases. The reason is that the available bandwidth 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

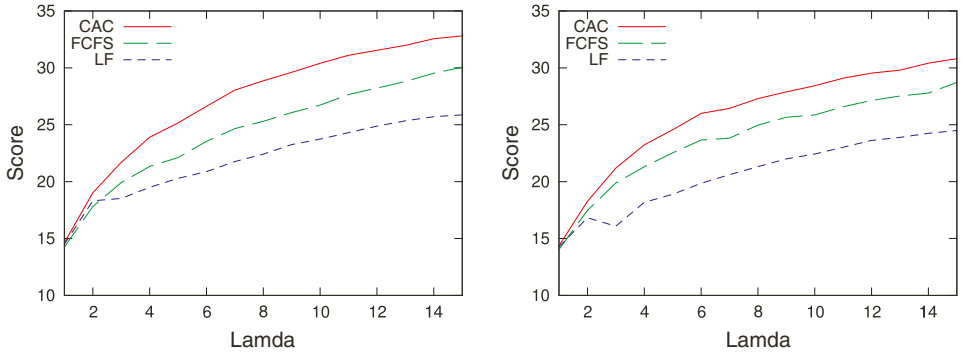


Figure 7. The system gain in the two metro scenarios when Internet connectivity is only available at every other station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1.

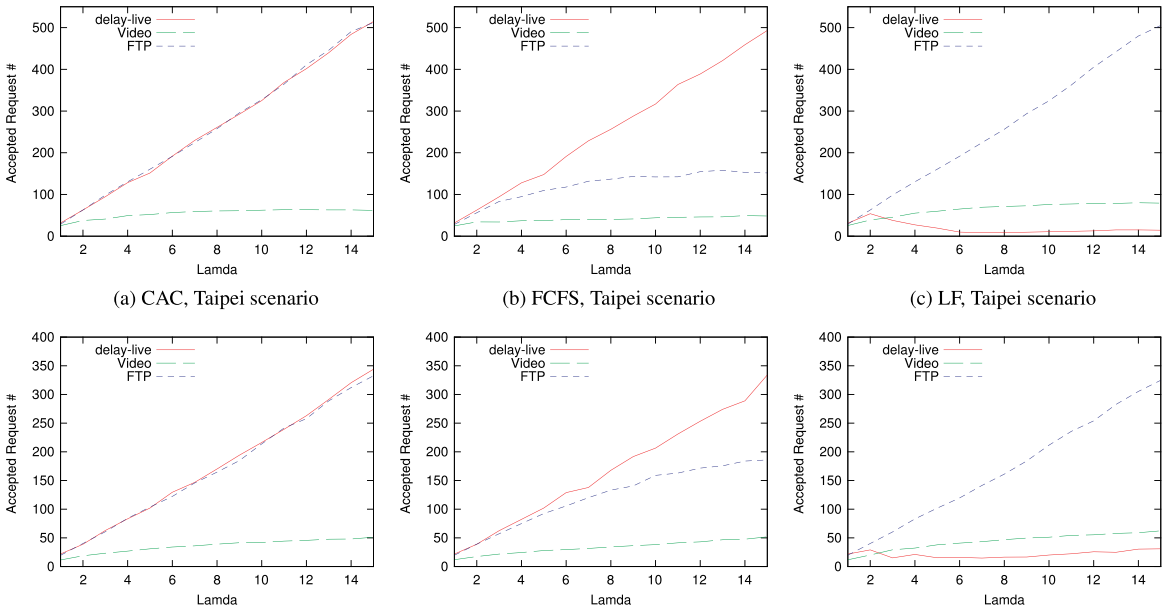


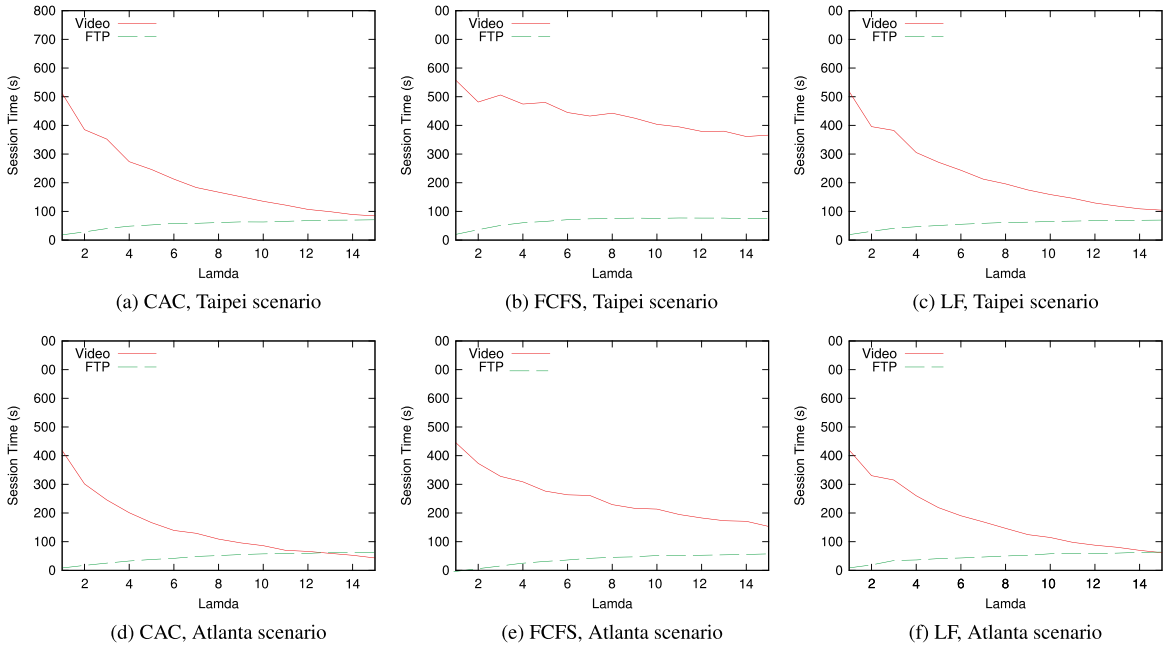
Figure 8. The number of accepted requests under the three algorithms in the two metro scenarios. Internet connectivity is only available at every other station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:1:1.

Trans. Emerging Tel. Tech. **25**:835–851 (2014) © 2013 John Wiley & Sons, Ltd.
DOI: 10.1002/ett

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

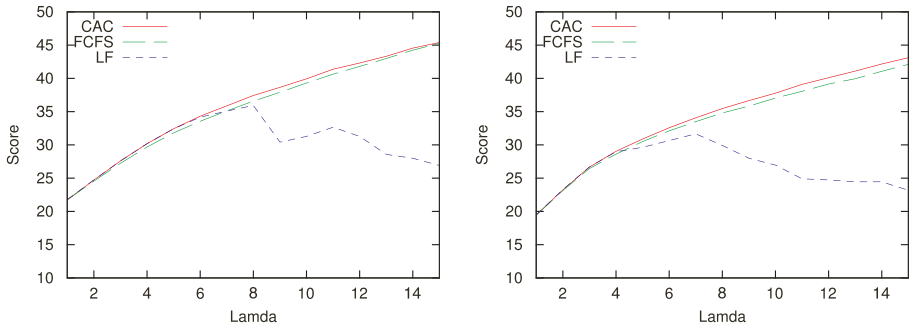
In this evaluation, a network connection is only available at every other station. The other parameter settings are the same as those used in Figure 4. Figure 7 shows that even if the network availability deteriorates, the proposed CAC algorithm still outperforms the FCFS and LF algorithms in both network 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).



(a) CAC, Taipei scenario (b) FCFS, Taipei scenario (c) LF, Taipei scenario

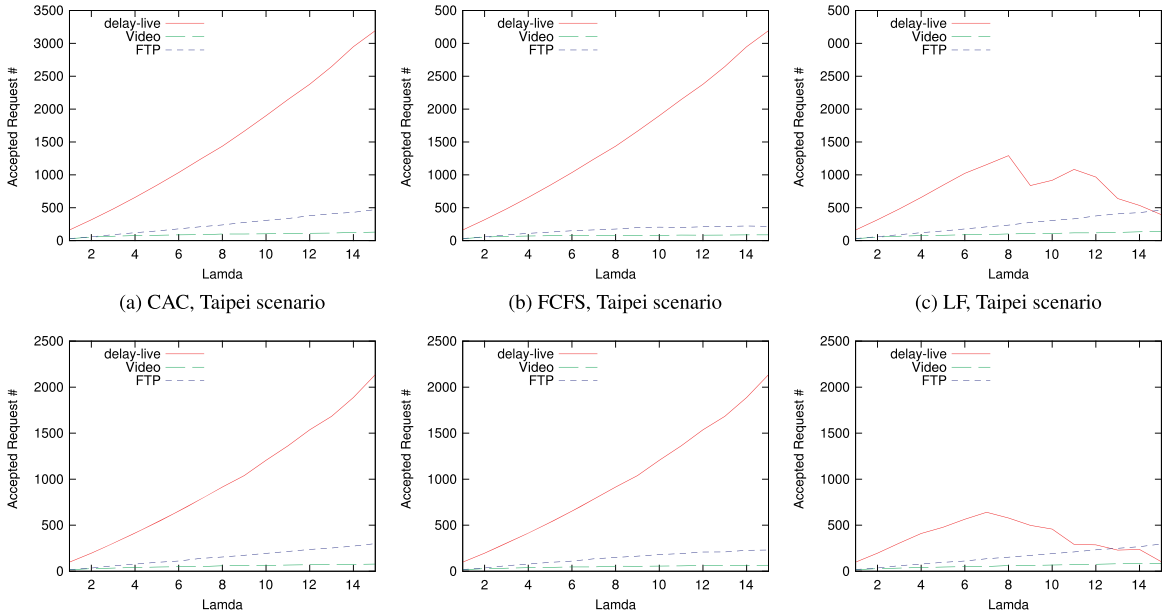
(d) CAC, Atlanta scenario (e) FCFS, Atlanta scenario (f) LF, Atlanta scenario

Figure 9. The session times 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) Taipei scenario (b) Atlanta scenario

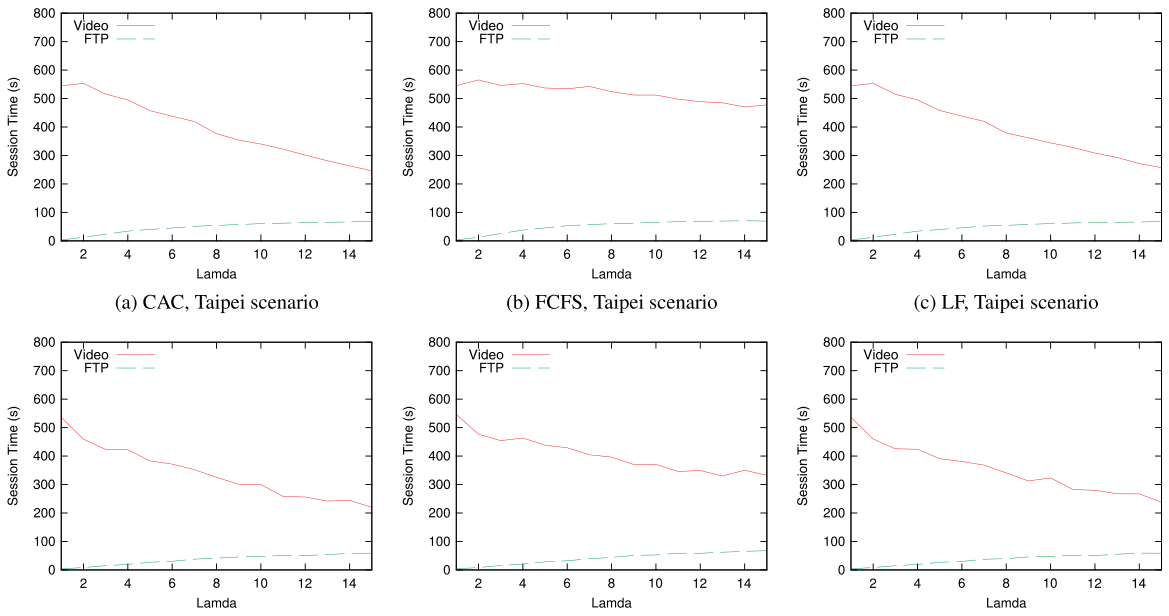
Figure 10. The system gain in the two metro scenarios when internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 5:1:1.



(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 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) CAC, Taipei scenario (b) FCFS, Taipei scenario (c) LF, Taipei scenario

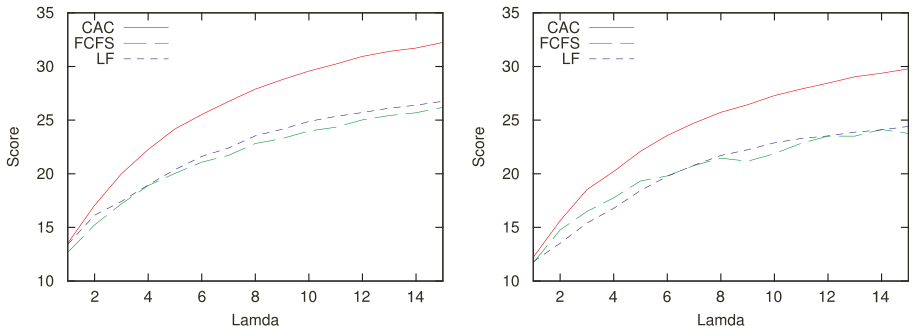
(d) CAC, Atlanta scenario (e) FCFS, Atlanta scenario (f) LF, Atlanta scenario

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.

It is unlikely that requests for longer video clips would be accepted because the amount of available bandwidth decreases as λ 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 prefetched data, as shown in Figure 8(c)). The results of the number of accepted requests and the session times for the Atlanta scenario are similar, as shown in Figures 8(d)–(f) and Figures 9(d)–(f).

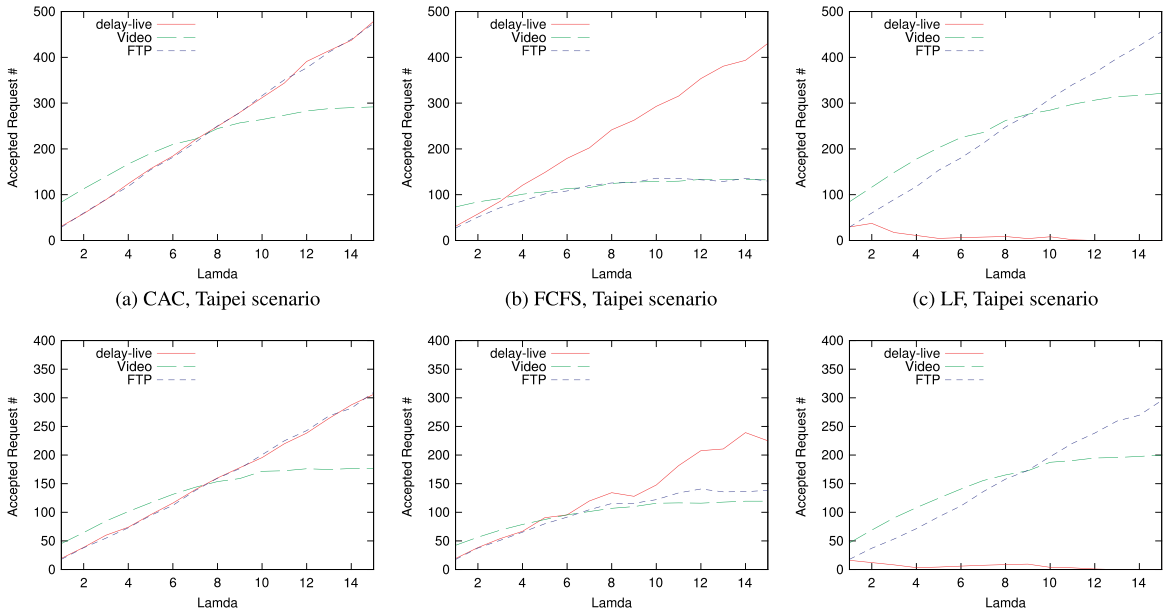
4.3. Evaluation III: Diversity of the types of requests

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.



(a) Taipei scenario (b) Atlanta scenario

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.



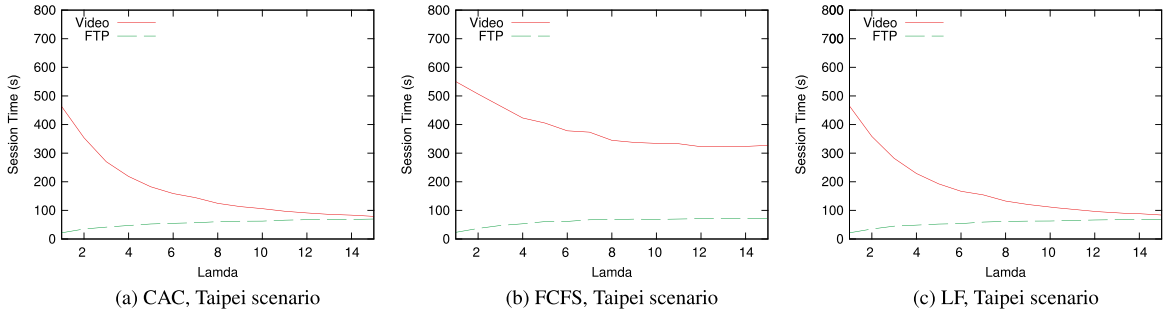
(a) CAC, Taipei scenario (b) FCFS, Taipei scenario (c) LF, Taipei scenario

(d) CAC, Atlanta scenario (e) FCFS, Atlanta scenario (f) LF, 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.

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 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.

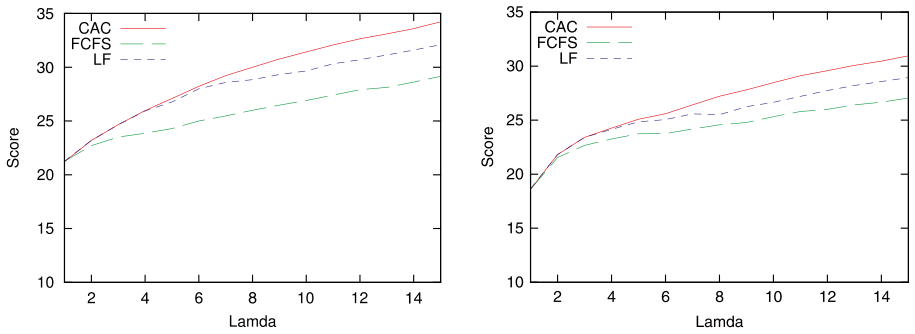
The simulation results for delayed-live video broadcasting, video clip streaming and FTP downloading requests when the ratio is 1:5:1 are shown in Figures 13, 14 and 15, respectively. Under the CAC algorithm, the number of requests accepted for video clip streaming is much higher than that in the baseline 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-



(a) CAC, Taipei scenario (b) FCFS, Taipei scenario (c) LF, Taipei scenario

(d) CAC, Atlanta scenario (e) FCFS, Atlanta scenario (f) LF, Atlanta scenario

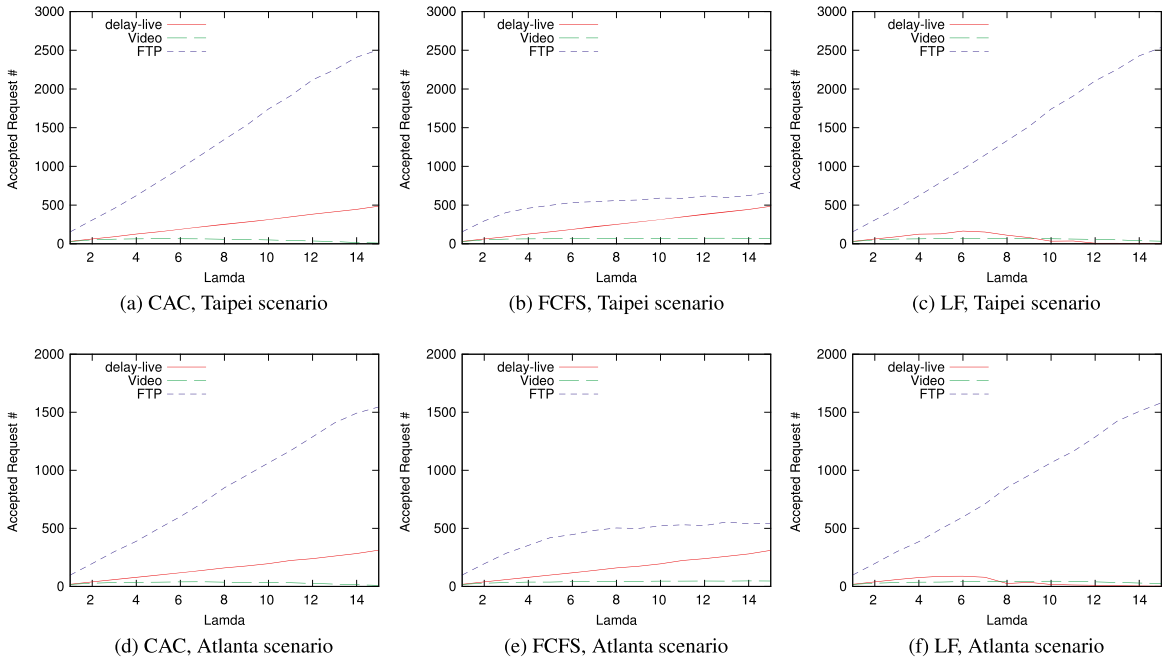
Figure 15. The session times under the three algorithms in the two metro scenarios. Internet connectivity is available at each station, and the ratio of delayed-live video broadcasting, video clip streaming and file transfer protocol downloading requests is 1:5:1.



(a) Taipei scenario (b) 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.

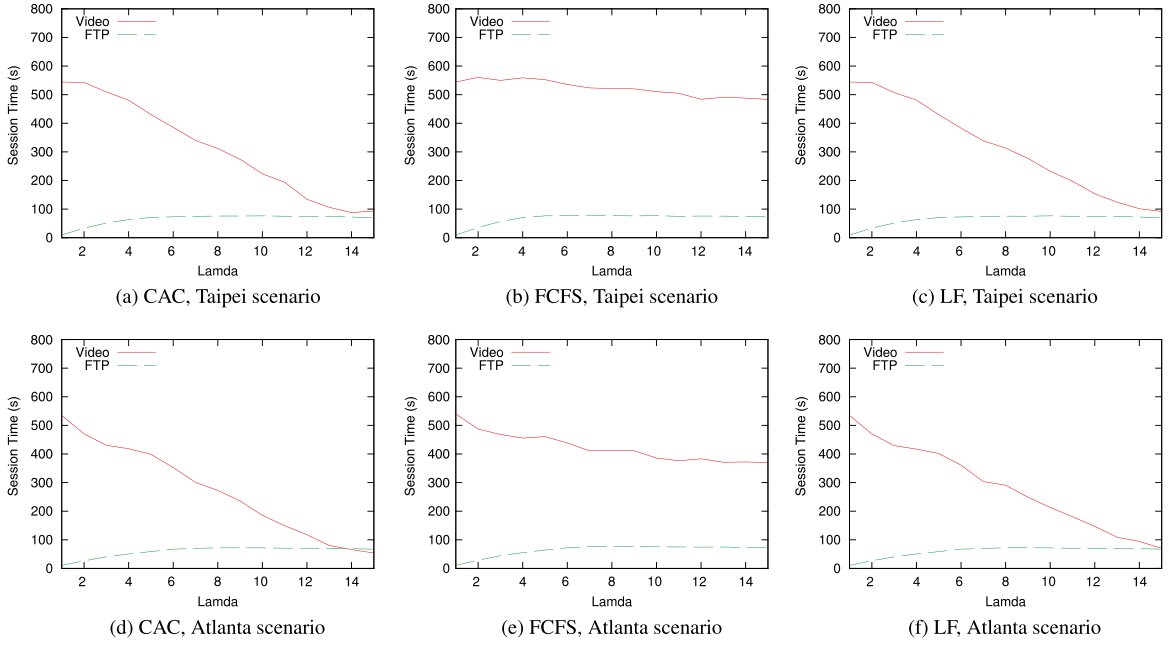
Trans. Emerging Tel. Tech. **25**:835–851 (2014) © 2013 John Wiley & Sons, Ltd.
DOI: 10.1002/ett



(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 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) CAC, Taipei scenario (b) FCFS, Taipei scenario (c) LF, Taipei scenario

(d) CAC, Atlanta scenario (e) FCFS, Atlanta scenario (f) LF, 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.

live video broadcasting and FTP downloading. The results in Figures 14(c) and 14(f) 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

1. Hossain E, Chow G, Leung VC, *et al.* Vehicular telematics over heterogeneous wireless networks: a survey. *Computer Communications* 2010; **33**(7): 775–793.
2. Sichitiu M, Kihl M. Inter-vehicle communication systems: a survey. *IEEE Communications Surveys Tutorials* 2008; **10**(2): 88–105.
3. Ott J, Kutscher D. Drive-thru internet: Ieee 802.11b for "automobile" users, In *INFOCOM 2004, vol.1*, Hong Kong, 2004; 362–373.
4. Ott J, Kutscher D. A modular access gateway for managing intermittent connectivity in vehicular communications. *European Transactions on Telecommunications* 2006; **17**(2): 159–174.
5. MRT system (Taipei). http://english.trtc.com.tw/ [25 January 2013].
6. MARTA system (Atlanta). http://www.itsmarta.com/ [25 January 2013].
7. Voyiatzis A. A survey of delay- and disruption-tolerant networking applications. *Journal of Internet Engineering* 2012; **5**: 331–344.
8. D'Souza RJ, Jose J. Routing approaches in delay tolerant networks: a survey. *International Journal of Computer Applications* 2010; **1**(17): 8–14.
9. Jones EP, Li L, Schmidtke JK, Ward PA. Practical routing in delay-tolerant networks. *IEEE Transactions on Mobile Computing* 2007; **6**: 943–959.
10. Mundur P, Seligman M. Delay tolerant network routing: beyond epidemic routing, In *3rd International Symposium on Wireless Pervasive Computing (ISWPC)*, Santorini, Greece, 2008; 550–553.
11. Doering M, Pögel T, Wolf L. Dtn routing in urban public transport systems, In *Proceedings of the 5th ACM Workshop on Challenged Networks (CHANTS)*, Chicago, IL, USA, 2010; 55–62.
12. Cao Y, Sun Z. Routing in delay/disruption tolerant networks: a taxonomy, survey and challenges. *IEEE Communications Surveys Tutorials* 2013; **15**(2): 654–677.
13. Luo G, Zhang J, Huang H, Qin K, Sun H. Exploiting intercontact time for routing in delay tolerant networks. *Transactions on Emerging Telecommunications Technologies* 2012. DOI: 10.1002/ett.2553.
14. Ott J, Kutscher D. A disconnection-tolerant transport for drive-thru internet environments, In *INFOCOM 2005, vol.3*, Miami, FL, USA, 2005; 1849–1862.
15. Tan WL, Lau WC, Yue O. Modeling resource sharing for a road-side access point supporting drive-thru internet, In *Proceedings of the Sixth ACM International Workshop on VehiculAr Inter-NETworking (VANET)*, Beijing, China, 2009; 33–42.
16. Tan WL, Lau WC, Yue O, Hui TH. Analytical models and performance evaluation of drive-thru internet systems. *IEEE Journal on Selected Areas in Communications* 2011; **29**(1): 207–222.
17. Xie L, Li Q, Mao W, Wu J, Chen D. Association control for vehicular wifi access: pursuing efficiency and

fairness. *IEEE Transactions on Parallel and Distributed Systems* 2011; **22**(8): 1323–1331.
18. Pogel T. Optimized DTN-routing for urban public transport systems, In *17th GI/ITG Conference on Communication in Distributed Systems*, Kiel, Germany, 2011; 227–232.
19. Kim M, Liu Z, Parthasarathy S, Pendarakis D, Yang H. Association control in mobile wireless networks, In *INFOCOM 2008*, Phoenix, AZ, USA, 2008; 1256–1264, DOI: 10.1109/INFOCOM.2008.182.
20. Bychkovsky V, Hull B, Miu A, Balakrishnan H, Madden S. A measurement study of vehicular internet access using in situ wi-fi networks, In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom)*, Los Angeles, CA, USA, 2006; 50–61.
21. Balasubramanian A, Mahajan R, Venkataramani A, Levine BN, Zahorjan J. Interactive wifi connectivity for moving vehicles, In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, Seattle, WA, USA, 2008; 427–438, DOI: 10.1145/1402958.1403006.
22. Balasubramanian A, Zhou Y, Croft WB, Levine BN, Venkataramani A. Web search from a bus, In *Proceedings of the second ACM workshop on challenged networks (CHANTS)*, Montréal, QC, Canada, 2007; 59–66, DOI: 10.1145/1287791.1287803.
23. Eriksson J, Balakrishnan H, Madden S. Cabernet: vehicular content delivery using wifi, In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom)*, San Francisco, CA, USA, 2008; 199–210, DOI: 10.1145/1409944.1409968.
24. Hadaller D, Keshav S, Brecht T, Agarwal S. Vehicular opportunistic communication under the microscope, In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys)*, Puerto Rico, 2007; 206–219, DOI: 10.1145/1247660.1247685.
25. Hull B, Bychkovsky V, Zhang Y, Chen K, *et al.* Cartel: a distributed mobile sensor computing system, In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, Boulder, Colorado, USA, 2006; 125–138, DOI: 10.1145/1182807.1182821.
26. Mahajan R, Zahorjan J, Zill B. Understanding wifi-based connectivity from moving vehicles, In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, San Diego, CA, USA, 2007; 321–326.
27. Zhang X, Kurose J, Levine BN, Towsley D, Zhang H. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing, In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, Montréal, QC, Canada, 2007; 195–206, DOI: 10.1145/1287853.1287876.
28. Jeong J, Guo S, Gu Y, He T, Du D. Tsf: trajectory-based statistical forwarding for infrastructure-to-vehicle data delivery in vehicular networks, In *IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, Genoa, Italy, 2010; 557–566, DOI: 10.1109/ICDCS.2010.24.
29. Pack S, Rutagemwa H, Shen X, Mark J, Park K. Proxy-based wireless data access algorithms in mobile hotspots. *IEEE Transactions on Vehicular Technology* 2008; **57**(5): 3165–3177.
30. Shahverdy M, Fathy M, Yousefi S. Scheduling algorithm for vehicle to road-side data distribution. *High Performance Networking, Computing, Communication Systems, and Mathematical Foundations* 2010; **66**: 22–30.
31. Zhang Y, Zhao J, Cao G. On scheduling vehicle-roadside data access, In *Proceedings of the Fourth ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, Montréal, QC, Canada, 2007; 9–18, DOI: 10.1145/1287748.1287751.
32. Zhang Y, Zhao J, Cao G. Service scheduling of vehicle-roadside data access. *Mobile Network Applications* 2010; **15**: 83–96. DOI: 10.1007/s11036-009-0170-9.
33. Luan TH, Ling X, Shen XS. Provisioning qos controlled media access in vehicular to infrastructure communications. *Ad Hoc Networks* 2012; **10**(2): 231–242.
34. Keränen A, Ott J. Dtn over aerial carriers, In *Proceedings of the 4th ACM workshop on challenged networks (CHANTS)*, Beijing, China, 2009; 67–76, DOI: 10.1145/1614222.1614234.
35. Gorcitz R, Jarma Y, Spathis P, *et al.* Vehicular carriers for big data transfers, In *Vehicular Networking Conference (VNC), 2012 IEEE*, Seoul, Republic of Korea, 2012; 109–114, DOI: 10.1109/VNC.2012.6407418.
36. Zarafshan-Araki M, Chin KW. Trainnet: a transport system for delivering non real-time data. *Computer Communications* 2010; **33**(15): 1850–1863.
37. Husni E, Rinaldi Sumarmo A. Delay tolerant network utilizing train for news portal and email services, In *2010 International Conference on Information and Communication Technology for the Muslim World (ICT4M)*, Jakarta, 2010; 6–10, DOI: 10.1109/ICT4M.2010.5971931.
38. Kasuari: emulation framework for agile network protocol development. https://prj.tzi.org/cgi-bin/trac.cgi/wiki/Kasuari [25 January 2013].
39. The network simulator - ns-2. http://www.isi.edu/nsnam/ns/ [15 October 2012].
40. Dehaene S, Izard V, Spelke E, Pica P. Log or linear? Distinct intuitions of the number scale in western and amazonian indigene cultures. *Science* 2008; **320**(5880): 1217–1220.