Media Sharing based on Colocation Prediction in Urban Transport

Liam McNamara
Dept. of Computer Science
University College London
London, WC1E 6BT, UK
l.mcnamara@cs.ucl.ac.uk

Cecilia Mascolo
Computer Laboratory
University of Cambridge
Cambridge, CB3 0FD, UK
cm542@cl.cam.ac.uk

Licia Capra
Dept. of Computer Science
University College London
London, WC1E 6BT, UK
l.capra@cs.ucl.ac.uk

ABSTRACT

People living in urban areas spend a considerable amount of time on public transport, for example, commuting to/from work. During these periods, opportunities for inter-personal networking present themselves, as many members of the public now carry electronic devices equipped with Bluetooth or other wireless technology. Using these devices, individuals can share content (e.g., music, news and video clips) with fellow travellers that are on the same train or bus. Transferring media content takes time; in order to maximise the chances of successful downloads, users should identify neighbours that possess desirable content and who will travel with them for *long-enough* periods. In this paper, we propose a user-centric prediction scheme that collects *historical* colocation information to determine the best content sources. The scheme works on the assumption that people have a high degree of regularity in their movements. We first validate this assumption on a real dataset, that consists of traces of people moving in a large city's mass transit system. We then demonstrate experimentally on these traces that our prediction scheme significantly improves communication efficiency, when compared to a memory(history)-less source selection scheme.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communications

General Terms

Algorithms, Design, Experimentation, Human Factors.

Keywords

Bluetooth, P2P, Transport, Media

1. INTRODUCTION

The amount of time that people living in urban areas spend travelling to/from work is significant. A study from

the University of the West of England has reported that, in 2006, the average commuter living in big cities in the United Kingdom (e.g., Liverpool, London and Manchester) spent 139 hours a year travelling to and from work, with this figure increasing to a whole month per year for Londoners [15]. The prohibitive cost of personal transport, and the increasing length of distances being travelled, has made public transport (e.g., bus, train, subway) the preferred means of travel by many commuters, with the London tube alone carrying an average of 3.4 million people every weekday. This kind of travel often exhibits routine patterns and the expression *familiar strangers* [19] has been coined to designate fellow passengers that are periodically met.

During these journeys, most commuters carry electronic devices, often equipped with short-range wireless network interfaces (e.g., WiFi and Bluetooth), which are increasingly being left switched on [7]. The presence of these networked devices, routinely in contact, sometimes for prolonged periods, offers new possibilities for research in automated content (e.g., news, video clips, music files) sharing and distribution. Short-range device-to-device transfers have many advantages over infrastructure based solutions. They do not require network coverage, so can be performed anywhere, even underground; the avoidance of a middle-man also precludes censorship and snooping. Use of a cellular network ISP for filesharing may be costly and could lead to similar problems that large, wired ISPs have with network capacity, possibly even worse due to the use of a shared medium. However, when using wireless neighbours, a challenge arises in terms of *whom to select as a content source*, given that a local source may not be colocated with us long enough for downloads to complete.

In this paper, we propose a new content source selection scheme for *single-hop*, peer-to-peer based content sharing on public transport. Our aim is to identify, among colocated peers that have relevant content, the one that has the highest chance to remain colocated long enough for data transfers to complete, thus minimising resource waste due to incomplete transfers and interference. We do so by making each device *record colocation information* about fellow commuters met during their journeys, *infer colocation patterns* from the historical data, then *automatically* and *dynamically* select the best content source, based on these patterns.

More precisely, let us take a peer A's perspective. Initially, every peer is a *complete stranger* to A; our scheme records, in a single *anonymous profile* on A's device, the aggregate mean colocation length of all encounters occurred thus far in a given *time slot* (e.g., for each hour of the day).

Once that same peer has been encountered often enough, it is promoted to *familiar stranger* status: for each familiar stranger B, A maintains a *personalised record*, recording the mean colocation length with B only, again for each time slot. This personalised information is used to infer colocation patterns and make more accurate predictions of B's colocation duration. The length of a time slot can either be fixed (e.g., one hour) or customised to cover logically meaningful time periods (e.g., *morning rush hour* or *lunchtime*); customised slot periods could be dynamically learnt by a device simply observing its colocation patterns.

When issuing a request to find content, a user's device will first filter out, from the set of devices currently in reach, those not possessing any content that matches A's interests. The remaining devices will then be ranked according to their *predicted remaining colocation duration*, and the peer that is most likely to be available long enough for a transfer to complete is selected as the source.

In devising our approach to content sharing, we have made the assumption that some commuters tend to meet regularly in their daily travel, and that their colocation patterns can be roughly predicted. People's regularity of movement, and repetition of journeys to the same place, is demonstrated convincingly by González et al [8]. We have validated this assumption on movement logs of commuters travelling within a major city's subway. In particular, we have used transport movement data, collected on two subway lines over a period of one month. We will report the results of this analysis, together with a performance evaluation of our approach, in Section 5.

The approach described in this paper focuses on maximising completed transfers, to ensure content can be efficiently shared in these environments. It does not consider the trustworthiness of the provider or the accuracy of content descriptions. We consider these issues to be orthogonal, and thus refer readers to approaches that explicitly deal with them, and that could be leveraged on top of our prediction scheme [16, 13]. Moreover, we are purely interested in the technical challenges faced when developing content sharing applications for decentralized and mobile peer-to-peer networks, and are not tackling the issue of the legality of media transfer. The release of the *Microsoft Zune* player, which automatically wraps wireless file transfers in Digital Rights Management (DRM) restrictions, demonstrates how our scenario can become a driver for media industry sales. Crucially, we want to create a system that will work with current hardware without the need for special infrastructure support, that can be deployed in the real world now.

The paper is further organised as follows: Section 2 depicts a public transport scenario and sets the scene of our work. Section 3 describes our approach, focusing on the key aspects of the prediction scheme. The implementation of the approach and its behaviour in an underground rail system is covered in Section 4. The details of the public transport traces we have used for the evaluation, the simulation settings and the results are discussed in Section 5. In Section 6 we cover some related work, before concluding and outlining the future direction of our research in Sections 7 and 8 respectively.

2. SCENARIO

To exemplify our approach to content source selection, we introduce a typical scenario where the previously made observations hold: commuters moving through a large city by means of a mass transport system (e.g., an underground metropolitan railway), carrying devices (e.g., phone/PDA/music player) that can play audio/video files and have wireless capability (namely, for this scenario, Bluetooth). Users are looking for particular content categories of interest to them, such as particular genres of music, e.g., *rock* or *pop.*

Alice is a commuter, travelling to work along her usual route, at her usual time, upon a rapid transit system train. She uses her phone as a music player for entertainment while travelling. Whilst in the transport system, she is unable to connect to the Internet as there is no network coverage (or because the connection cost is too high). However, the large number of other passengers travelling with her, offers means of procuring media for her enjoyment: during periods of peer-to-peer connectivity, devices may transfer music tracks or video clips between each other. Alice is interested in *electronica* and *rock* genres of music and would like to acquire more from fellow travellers. Bob is another commuter who travels to work along the same train line, and is often on the same train as Alice, as they both are due at work in the city centre at 9:00AM. He enjoys listening to *pop* and *rock* music. Carol has taken the day off from work and is meeting her friends in the city centre for breakfast and to do some shopping. She is on the same train, but would not normally use this line; she is listening to her music collection of *dance* and *electronica*. Both Carol's and Bob's devices are potential sources of content for Alice, as they carry genres of interest to Alice (in particular, rock from Bob and electronica from Carol). Alice's device now has to select which one to download from.

If the content size was very small and could be transferred within a few seconds, the choice of who to download from, among those having relevant content, would not be critical. However, when sharing media content that requires minutes to be transferred (e.g., high quality music files or video clips), it becomes important to select a source that will remain colocated long enough for the operation to complete. If the predicted colocation duration with Bob, computed based on previous encounters, is long enough to enable more of his data to be shared, then Bob represents a more reliable choice than Carol. In fact, as Carol is a complete stranger to Alice, the length of their colocation cannot be accurately predicted (she may be getting off in the next minute), and thus she should not be favoured.

In this scenario, predictability of colocation time is a critical parameter upon which we base the selection of the content provider. This is due to the high *churn* of people on the train, with large percentages of people leaving and boarding the train, especially at large stations with many connecting lines. In the next section, we present our approach to automatic content source selection in urban transport, based on a colocation prediction scheme that exploits routines in people's travelling, coupled with basic filtering of sources based on common categories of interests.

3. APPROACH

The thesis underpinning our work is that users show a high degree of regularity in their movements, repeating similar journeys over and over again. Although the number of unknown peers we encounter daily, while travelling on public transport in a city, will always be very high, we argue that a non-negligible set of passengers will be re-

encountered regularly. We will validate this assumption on a large set of movement traces collected from two lines of a large metropolitan subway system over a period of one month (Section 5.1). Based on this assumption, we first describe how colocation patterns are recorded over time and used to estimate future colocation durations (Section 3.1); we then illustrate how these predictions are used by our content source selection process (Section 3.2) to maximise the chance of complete downloads.

3.1 Colocation Pattern Logging

Based on the observations described above, we have designed an approach to enable a user's device to automatically select the best provider of relevant content among the (potentially large) set of co-travellers. By 'best' provider, we refer to the one, among those having 'interesting' content (see Section 4.1), who has the highest chance of remaining colocated for the whole duration of a transfer. To perform this selection, a user must be able to estimate the *remaining length of colocation* with another user, and assess whether this time would allow media content to be shared.

One technique to measure this value dynamically is to keep a running mean: for example, each user could log all her/his previous encounters, recording the mean colocation duration with every other peer, and use these values as a prediction of the length of the current encounter. However, this is an overly simplistic metric which ignores the *seasonal* variability of movements; for instance, a colocation beginning in the morning of a weekday (e.g., on a daily commute to work) may last much longer than one starting in the evening (e.g., on a trip to a local supermarket). We argue that keeping a single running mean of colocation duration for every pair of users is not sufficient, as important information about their movement patterns and seasonality is lost.

A more advanced scheme, which we have adopted in our approach, is to keep a *temporally varying mean*, that changes according to the time at which the colocation began. More specifically, our solution keeps a separate mean for each TIME_SLICE during a given TIME_PERIOD. This detailed profile of another peer's colocation pattern is kept in the form of a *personalised profile* for each peer that we have deemed as *familiar strangers*, that is, those that have been met often enough in previous journeys. If the number of previous samples during a given TIME_SLICE is not sufficient (in our experiments, less than three) to confidently determine a temporally-specific mean, then the average colocation mean, computed across all of this peer's personal profile's time slices, is used. For all other unfamiliar peers, a single *anonymous profile* is used instead (i.e., one profile for the full set of peers), containing, in each time slice, the mean length of all encounters occurred thus far with every host in that slice. The selection of values for TIME_PERIOD and TIME_SLICE are domain specific aspects and universal values do not exist that will hold in all situations. A plausible choice for our human movement scenario would be a one-day period, with a one-hour slice. Rather than hardcoding them, these values could be dynamically learnt from a domain's colocation structure, using techniques similar to the ones discussed in [6]. In this work, we have used two schemes: a basic one-day/one-hour structure, and a more advanced one, emerging from the data traces we have of people moving on public transport, with variable-length time slices (see Section 5).

Recording and processing personalised colocation statistics incurs a small amount of storage and processor usage; this amount scales linearly with each additional familiar stranger host that is tracked. If a device is particularly limited in storage, a resource conservative policy can be used to limit the size of this set (e.g., by requiring more frequent encounters for a host to reach familiarity level). Note that this colocation pattern logging is service-agnostic, and can be implemented as a standalone component, with the information it gathers being leveraged by multiple services running on a device. For example, with a Bluetooth network, such a component would simply make each device regularly enter the *inquiry substate* to discover neighbours; discoverable Bluetooth devices periodically enter the *inquiry scan substate* and respond to device discovery requests with a Frequency Hop Synchronisation packet (FHS), containing their relevant device ID.

3.2 Content Source Selection

We now describe the steps of our content source selection protocol. At a given time slot s (uniquely determined by the current time), user A is looking for a source from where to download content files with a genre in its interest set G_A. The following steps are performed:

1. A filters out, from the list of hosts in reach, *uninteresting* peers, that is, those peers who do not have any content of interest for A. We will discuss in Section 4.1 how this pre-filtering takes place.

2. For each of the hosts left in A's list H, a prediction is made of the length of the colocation with A, based on the scheme described previously.

3. The predicted remainder of the current colocation time, for each host in H, is computed (this simply requires logging the start point of each colocation instance).

4. Hosts are ranked in decreasing order of remaining predicted colocation time.

5. Finally, A connects to the top ranked host $\overline{h}$, to obtain a summary of relevant files available for download; in our experiments, we have set the size of the summary (SUM_SIZE) to 50 (any value above 20 caused negligible variations). If the summary contains at least one desirable file (e.g., a rock music file that A does not already have), and the expected remaining colocation time is longer than the estimated download time t, the transfer begins. A host will not include a file in the summary if it was included in a previous summary, sent during the current pair's colocation; this avoids re-advertising files and allows a host to know when no more interesting files are available.

A more formalised expression of this scheme is shown in Algorithm 1. In order to decide whether to start the download (Step 5), A must estimate the time t it will take for the download to complete, based on the content size and current connection speed. Both these pieces of information can be obtained/estimated during the transfer negotiation phase[1],

---

[1] Note that file matching is only performed upon the selected host, to avoid the need for active participation of every neighbour.

Algorithm 1 Content Source Selection.

Parameters: list H of hosts in reach and their colocation start time start$[h]$; current time slot s; matrix M of personalised profiles for familiar strangers (i.e., $M[h,s]$ = mean colocation duration with host h for time slice s, $M[h,\emptyset]$ = mean of all h's colocations); vector V storing the anonymous profile; list G_X of genres of interest to X; expected transfer time t_{tr} for chosen content.

Returns: host $\overline{h}$ from where to start the download.
{Step (1) - Pre-filtering based on interest}
for all $h \in H$ **do**
  if $G_A \cap G_h = \emptyset$ **then**
    $H = H \setminus \{h\}$
  end if
end for
{Steps (2,3) - Computation of colocation predictions}
for all $h \in H$ **do**
  if host h is familiar stranger **then**
    if $M[h,s]$ populated **then**
      score$[h]$= $M[h,s]$ - $(t_{now} - \text{start}[h])$
    else
      score$[h]$= $M[h,\emptyset]$ - $(t_{now} - \text{start}[h])$
    end if
  else
    score$[h]$=$V[s]$ - $(t_{now} - \text{start}[h])$
  end if
end for
{Steps (4,5) - Rank peers and return chosen source}
sortDecreasing(H, score)
getFileSummary($H[\text{top}],G_A$)
if score[top] $> t_{tr}$ * ATTEMPT_THRESH **then**
  $\overline{h} = H[\text{top}]$ {Select host with longest predicted remaining colocation}
else
  $\overline{h} =$ null {Do not transfer, likely to fail}
end if
return $\overline{h}$

giving the approximation $t = content\_size/E(speed)$. Downloads will not be initiated if the expectation of success is too low. More precisely, $t * ATTEMPT\_THRESH$ needs to be smaller than the expected remaining colocation, with the threshold parameter ATTEMPT_ THRESH being set to 1 in our experiments. This value allows a device to account for the unpredictable nature of the actual time taken by a transfer, and the duration of actual colocation: with values higher than 1, the device takes less risks, starting content transfers only when remaining colocation is estimated to be substantially longer than t. Conversely, values lower than 1 will start downloads even if the prediction technique suggests they will not succeed. The experimental value of 1 was chosen to directly measure the efficacy of colocation predictions.

With reference to the scenario described in Section 2, let us assume that Bob's mean colocation duration with Alice in the time period 8-9AM is 20 minutes, and that they have already been colocated for 8 minutes. Bob's predicted departure time will be 12 minutes away. Let us also assume that the mean of all other host's colocations with Alice, during this time slice, is 10 minutes. As Carol is unknown to Alice, her colocation duration will be predicted using the anonymous statistics, that is, 10 minutes. If Carol has already been colocated with Alice for 5 minutes, her predicted remaining time will be 5 minutes, causing Bob to be selected as the connection partner, as he offers the best chances of completing the transfer of media content. A connection would

be established with Bob, and a file discovery performed, asking for files of type rock (Step 5). Bob's sharing application responds with a shortlist of rock files that he is able/willing to share; only a subset is advertised, as sharing whole library lists would become time consuming.

4. IMPLEMENTATION

This section details our development of a prototype application, covering the method of device discovery (Section 4.1), a high level view of the application's behaviour (Section 4.2), and finally some testing figures, Section 4.3. We implemented our system for Symbian S60 phones, utilising less than 2KLOC for the PyS60 v1.4.3 Python interpreter [18]. Our testbed comprised of Nokia N70s running S60 Platform Second Edition - Feature Pack 2, equipped with Bluetooth radio interfaces. We also installed some 2GB MMC-Mobile flash cards, to supplement the relatively small onboard memory, allowing the storage of large media libraries.

Python was chosen to facilitate rapid development and easy porting to another platform. It also avoids the restrictions of the Java security model, such as requiring applications' signing to allow sensitive operations without user confirmation. We utilised code from the Personal Distributed Information Store (PDIS) project at the Helsinki Institute for Information Technology. Their custom-built discovery code allows Bluetooth device discovery without requiring user intervention, contrary to the standard PyS60 discovery. The application can be minimised to run in the background, to release the phone to perform other tasks, while files are shared, allowing a user to start the program and forget about it.

4.1 Bluetooth Service and Content Matching

In this section, we detail our content matching mechanism and its operation over Bluetooth. Bluetooth offers device and service discovery mechanisms as part of its specification [3]. A service is described by a *Service Record*, which consists of a list of *Service Attributes*. Some of these attributes are predefined: most importantly, the *ServiceClassID*, a Universally Unique Identifier (UUID) number which enables efficient identification of the services that a device supports (in our scenario, the music sharing functionality). Our protocol can make use of further arbitrary attributes to communicate, as part of the Bluetooth Service Discovery Protocol (SDP), the category of content (i.e., genres) that a device makes available for sharing.

More precisely, each genre is encoded as the MD5 digest of the genre's plain text name, obtaining a 128bit value to be used as a service attribute's value within a service record. Each genre of interest is hashed to a separate service attribute and added to the service record. Hashes need only be computed when a device changes interests, and thus does not represent any significant computational overhead. The hashing allows category names of arbitrary length to be included in service records, and adds some obfuscation of a host's interests, as it would be time consuming to compute a lookup table of genre names (this could be improved by using a salt of the Bluetooth address) However, hosts looking for particular genre types will only have to hash the small number of their interests.

During a Bluetooth discovery phase, service records of peers in reach are obtained by performing a *SDP_ServiceSearchAttributeRequest* transaction. Received

Action	Duration (s)
Startup	2
Device discovery	10
File negotiation	1
5MB transfer (quiet)	117.9
5MB transfer (busy)	186.9
Copy 5MB to memory card	2

Table 1: Action timings

service records can be efficiently checked for matching interests; hash collisions are extremely unlikely, especially as genre names are typically short. This step allows the searching host to know the categories of files that are likely to be available on a peer device, without having to communicate with the server application and manually searching the library. If the interests advertised by a peer do not match any genre of interest to the querying device, the peer will simply be ignored. The pre-defined *ServiceAvailability* attribute allows notification of whether the service is already busy sourcing to a different host, thus enabling further filtering of hosts already engaged in other transactions.

With reference to the scenario from Section 2, Alice would perform a service discovery on both Bob and Carol, who respond with their service records, which include their encoded genre service attributes, as well as the ServiceAvailability attribute (set to either $0x00$ if available, or $0xFF$ if busy). Alice then has enough information to select her source. The whole searching process is done using the Bluetooth SDP layer only, without interaction from Bob or Carol's music application, thus exploiting Bluetooth features to efficiently respond to genre searches.

4.2 Operation

Upon startup the application searches the predetermined directory (**E:/Music**) for MP3 files, and reads each file's ID3 meta-tag to build an internal meta-library. A list of all genres that a device contains is used as the *interest list*. An RFCOMM port is bound to by the *server* thread of the application, to listen for peer requests. The *discovering* thread then begins to periodically discover all neighbouring devices, recording their Bluetooth addresses, and updating the colocation profiles. The *client* thread regularly selects the best neighbour and attempts to initiate a download of content matching a category from the *interest list*. Files are downloaded to the fast internal phone memory and, upon completion, moved to the memory card's music folder for long term storage.

4.3 Prototype Performance

We tested our prototype on an actual mass transit train, both at busy and quiet times. Table 1 shows the time it takes to perform some of the basic actions of our system; these values were used to inform the parameterisation of the simulations. The difference between the time taken to transfer a 5MB file, when on busy trains and when near empty trains, was significant, increasing transfer time by nearly 50%. If many fellow passengers were engaged in data transfers, the time taken would increase even more.

When using small unbranded memory cards, data writing took non-negligible time, though once the larger, higher quality, ones were installed, this became a relatively constant 2 second overhead. This shows the exact behaviour of real

devices is dependent upon all hardware elements present. However, these timings seem to be reasonable values for the modelling of modern generation smartphones. The prototype demonstrates the feasibility of running a system such as ours on a phone released in the last few years.

5. EVALUATION

In this section we report on the performance evaluation of our content source selection protocol. We begin the discussion with an analysis of the dataset used (Section 5.1), to prove that the assumption we made about regularity of people's movement on public transport does hold. Section 5.2 then describes how we generated content libraries and distributed user interests, following realistic data models. We then illustrate the protocol variants we are comparing (Section 5.3), and the simulation settings we used (Section 5.4), before moving on to discuss the actual results of the simulations (Section 5.5). These tests were conducted with OMNeT++ [23], an open-source discrete event simulator written in C++.

5.1 Dataset

Despite the availability of some wireless traces of people using public transport, mainly from the Crawdad repository at Dartmouth [5], we wanted to perform large scale simulation of a city and its inhabitants. The movement traces used in this study were collected anonymously from a large metropolitan mass transit system over the course of one month. The use of Radio Frequency Identification (RFID) cards for electronic payment has been introduced in many city's transport networks over the course of the last few years, including Hong Kong's Octopus, Japan's Suica, London's Oyster and Washington, DC's SmarTrip. They are small electronic devices that can communicate wirelessly over short distances and store data. This technology allows easy payment and faster movement through ticket checkpoints; more importantly to our work, it enables the monitoring of passenger's movement in the system. We obtained data from two busy lines, each containing around 1 million journeys over the duration of the sample. These lines run through the centre of the city, and are used by commuters, shoppers and tourists.

All data was completely anonymised before processing, and contained fields *user_id*, *day*, *entry_time*, *entry_station*, *exit_time* and *exit_station*, with times being accurate to one minute. Only journeys that begin and end on the same lines are included. Journeys that are not complete (i.e., those missing an entry or an exit detection) are also not included.

5.1.1 Journey attributes

We begin our analysis of the dataset by studying the changing nature of journeys through the day (Figure 1). The 'volume' of passengers is shown as the percentage of people travelling at that period, with respect to the number of people travelling during a whole day. For each time of the day, we also show the mean length of journeys that begin at that time, and the standard deviation of journey durations. All plots use the same x-axis of time of day, and the two y-axis scales represent percentage on the left-hand side (for volume) and minutes on the right-hand side (for duration and standard deviation). There is an obvious bimodal nature to the volume of journeys due to the morning and evening rush hours, peaking at 500 minutes (8:20AM) and 1100 minutes
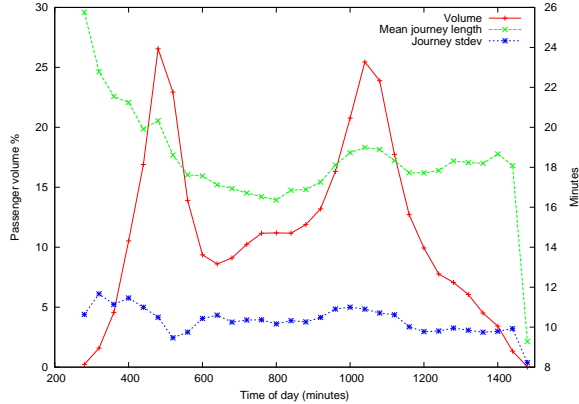
Figure 1: Volume and duration of journeys.

Figure 2: Passenger journey regularity.

(5:30PM) respectively. The evening rush hour lasts longer and finishes gradually, probably due to people finishing work at a wider variety of times, and people travelling to go out for the night. The journey durations also vary throughout the day, with longer journeys in the morning, assumedly from more journeys being made between the suburbs and the city centre; the nighttime drop is caused by the subway closing. Throughout the day there is a large variance in the mean durations, highlighting the need of distinguishing long colocation duration passengers from the short ones.

5.1.2 User attributes

We studied the frequency with which individuals travel in the trace. A very similar distribution appears for both train lines: the vast majority of travellers use the train only a few times during the sample, with around 70% of users being unique (i.e., they are seen in the system only once). Most importantly for us, a substantial subset of people frequently commute in and out of the city, and are seen between 20-40 times over the duration of the sample.

To gain a more precise idea of how regular passenger travel times are, we have analysed the occurrence of journeys over the time of day. The most regular user behaviour would involve travelling at the same time every day, and the least regular would never travel at a similar time on any day. We define a regularity metric for a particular user's journey as the number of their other journeys that occurred within 10 minutes of its *entry_time*. A user's overall regularity measure is then defined as their average journey regularity. Figure 2 depicts the number of users with a given regularity value; the three curves show the regularity results for *all users*, *users that travel more than five times* and *users that travel more than 10 times* in the sample. The number of users obtaining low regularity scores decreases significantly as the minimum number of journey threshold is increased, while the number of high scoring users stays constant. This confirms that the more frequently travelling hosts are also the ones having more regular journey times.

5.1.3 Passenger colocation

The data set only records when/where people entered and exited the subway system, and not who they were colocated
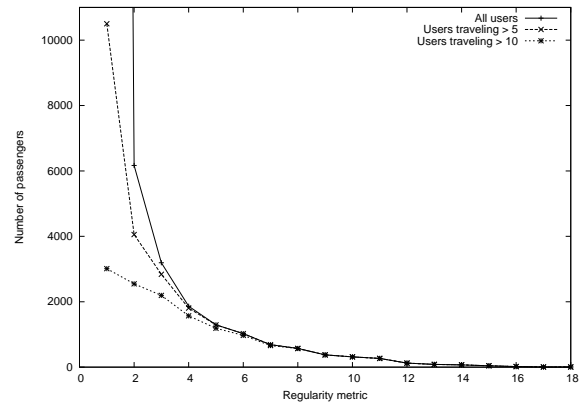

Algorithm 2 Same train algorithm.

---
for each journey i **do**
    for each other journey j **do**
        $path$=overlappedStations(i,j)
        T_1=timeAtStation($path_{start}$,i)
        T_2=timeAtStation($path_{start}$,j)
        if abs($T_1 - T_2$)≤**TRAIN_THRESH then**
            if $i_{entry\_station} == j_{entry\_station}$ **then**
                C_{start}=max(i_{begin},j_{begin})+**STNENTRY**
            else
                C_{start}=max(T_1,T_2)
            end if
            T_3=timeAtStation($path_{end}$,i)
            T_4=timeAtStation($path_{end}$,j)
            C_{end} = min(T_3,T_4)
            recordColocation(i,j,C,$path$)
        end if
    end for
end for

---

with in terms of wireless device ranges. Therefore we had to process the traces to extract meaningful colocation information to perform the content sharing experiments. The colocation time between a pair of users was estimated by making some simplifying assumptions. Each pair of journeys, travelling on the same line and in the same direction, is analysed to see if they overlap in time and space. In particular, we looked at a journey's *exit_time*, together with the official train journey times, to calculate when/how long the pair was on the same section of the line. If the users were traversing the same station within TRAIN_THRESH of each other, they are deemed to have been on the same train. This assumes people leave the system soon after their train arrives at their final stop. Algorithm 2 gives a formalised version of the processing used. The function $timeAtStation()$ estimates the time a passenger i was at a given station during their journey as: exit time (i_{exit_time}), minus the time it takes to leave the station (STNEXIT), minus the travel time between the exit station ($i_{exit\_station}$) and the station in question, according to the timetabled journey times. STNEXIT and STNENTRY are both assumed to be 1 minute, TRAIN_THRESH is the mean period between consecutive trains for that hour of the day.
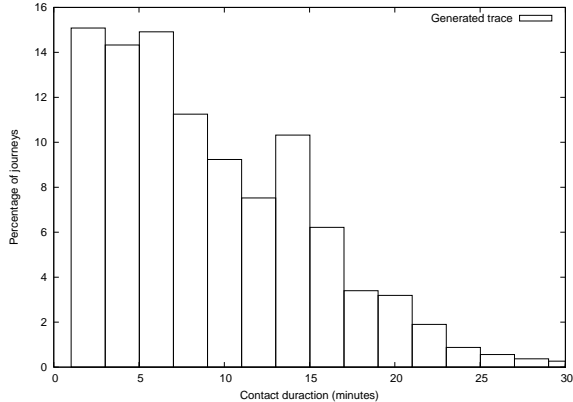
Figure 3: Contact times.

Figure 4: Intercontact time CCDF.

We consider Bluetooth connectivity range to be the specification rating of 10 meters for Class 2 devices. Its relatively short range means that even passengers on-board the same subway train will not necessarily be in range. Our testing (Section 4.3) found connectivity to be possible between adjoining carriages, and also over 10 meters if the carriages were not busy. We elected to distribute hosts uniformly randomly throughout the trains, a simplification for analysis.

Karagiannis et al [12] have showed that wireless traces often exhibit similar unifying features. As the generated colocation dataset is only derived from real movement traces, and not collected from actual radio devices, we have validated it possesses suitable features. The metrics of contact duration and inter-contact times are shown in Figure 3 and 4. The contact duration distribution follows an approximately exponential decay, with two key differing features. The pronounced spike, at 15 minutes, in the contact durations, is caused by the underlying distribution of stations and the journeys that people make between them, with the most popular journeys lasting around 12-15 minutes. Also, the proportionally low occurrence of short contacts is due to people walking rather than taking public transport for short journeys. The intercontact time shows a very clear exponential decay, with λ being approximately 0.165 for periods below 20 days (the data set is not long enough to give reliable data after this period). This matches with Karagiannis' observation of exponential decay in the tails of the intercontact CCDF.

5.2 Content Generation

In order to simulate a music sharing system, the tastes and libraries of users must be respectively assigned and populated. We make the assumption that users are interested in a set of genres, and would like to receive more music of those types (provided they do not already own that specific file); moreover, we assume that, within the one month simulation time we have, users do not change their interests.

Previous studies have recognised files in peer-to-peer systems to follow a generalised-Zipf distribution [21]. To verify these observations, particularly in the scenario of digital music libraries, we have collected data from Last.fm [1]. Last.fm is a large (21 million users) "social music" website that cre-
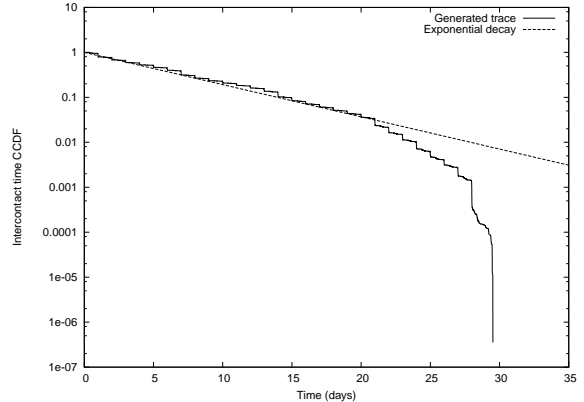
ates profiles of musical tastes by tracking what songs users listen to more often. Users are also encouraged to attach *tags* to artists, thus creating a folksonomy of music classification. Using the Audioscrobbler Web Services, we performed a breadth-first crawl of 500,000 users, together with their top 50 most played artists. For each artist, we also crawled its 50 most frequently associated tags. By ranking tags in order of their frequency across the whole crawl, we were capable of ranking music categories in terms of popularity (e.g., rock being more popular than electronica); moreover, when focusing on a particular category, we were able to rank artist popularity (e.g., *Red Hot Chili Peppers* being more popular, within rock, than *Metallica*). The same artist may appear under different genres, as they can be tagged in many different ways. Note also that some tags used do not refer to standard musical classifications (e.g., *seen live*, *awesome*, etc.). Users will inevitably tag files with non-standard terminology, and these tags will simply be leveraged by our approach.

We used this information both to associate music tastes to users, and to create music libraries accordingly. Each user is assigned a set of *interests*, that is, a subset of all the file categories; the precise number of interests per user is a parameter in our simulations, and it has been varied according to the observed interest distribution within individuals' libraries. The probability of a genre being selected as a user's interest is proportional to its popularity within the Last.fm dataset. To generate a particular user's library, a genre is first (uniformly randomly) selected from the user's interest list; an artist is then chosen according to its popularity within the chosen genre, and a music file created. If that file already exists in the library, another file is chosen. The process continues until the library is populated, with the initial library size being a simulation parameter. We assume content to be already categorised at the beginning of our simulation, and that reclassification (i.e., changing a file's genre from 'rock' to 'indie') does not occur (i.e., files are not mislabeled).

5.3 Protocol Variants

We compare three methods of content source selection, in order to validate the usefulness of our proposal.

- **Random (RANDOM)** - The content source is chosen at random from a peer's current neighbours, provided that there exists a non-empty overlap in interests. Due to Bluetooth's discovery procedure, this is actually the same as employing a *first-come first-served* selection mechanism [20]. Random selection provides a lower-bound on our expected performance of a source selection technique. Note that this is not the same as truly random selection, as hosts with disjoint musical tastes or busy ones are still pruned.

- **Colocation prediction (PREDICTION)** - The approach described in Section 3 is used: peers with shared interests are ranked in decreasing order of expected colocation, and the peer at the top, who is not already sourcing a download, is chosen. The download begins only if the expected remaining colocation is at least $t * ATTEMPT\_THRESH$, with t being the estimated download time, and the threshold parameter ATTEMPT_ THRESH set to 1 in all experiments.

- **Oracle (ORACLE)** - An *oracle* with perfect knowledge of future colocation durations informs a host about exactly how long its neighbours will remain colocated (though network speed variations are not known). The host thus knows, and always selects, to download a file from the neighbour with shared interests and with the longest remaining colocation time. Despite this not being an implementable decentralized algorithm, it provides an upper-bound to the performance of any selection technique.

These protocols differ only on the actual selection stage, with all other file acquisition steps being the same.

5.4   Simulation parameters

To ensure the simulations modeled reality as closely as possible, the timing and behaviour of our prototype was used to inform the selection of the simulation parameters. The full list of parameters used, together with an explanation of how we set them, is provided below. Note that there is no *learning phase* employed, the simulations are begun from a cold start, thus giving a pessimistic evaluation of our prediction based scheme. Performance is likely to improve upon the results presented here, once patterns have been learned.

- **Node number** - We have studied the effect of increasing the number of hosts (from 100 to 3000) on our performance metrics, and we have observed no significant changes once 2000 are being simulated. In all experiments, we have thus simulated 2000 nodes running our prediction protocol, as this number does approximate the real system. We chose to individually model the most frequently travelling passengers, as they stand to benefit the most from our system.

- **File size** - We are assuming compressed music/video files are being shared, with a three-minute CD quality encoded MP3 file being around 5 MB in size. File sizes are chosen from a normal distribution with a mean of 5 MB and standard deviation of 1 MB. The mean was also varied in the range [3, 10] MB, to represent content from small music files up to larger video clips.

- **Transfer rate** - Indicative experiments we ran on underground trains showed a large range in the achievable Bluetooth data rate, from 100kb/s to over 400kb/s; Bluetooth version v1.1 specification highest symmetric transfer mode DH5 (no FEC) is rated at 433.92 kb/s. In our simulations, the value for this parameter was a range of 100, increasing in increments of 50; from [100, 200] to [300, 400], and also [433, 433] to show a perfect network, giving a full range of possible network conditions. Real life achievable speed changes based on how busy the train is (human bodies interfere with Bluetooth's 2.4 GHz microwave frequencies), and other environmental factors, such as the vehicle dimensions. Also, other active Bluetooth devices in the area will interfere with the communication signal, thus varying the achievable connection speed.

- **Time slice** - Given that we are dealing with human movement, the time period has been fixed to one day. The duration of slots, used to group colocations into means, has been chosen to be a fine-grained 1 hour (thus 24 slots in a given period). We have also run experiments with a coarse-grained, more informed domain specific division, with the minute slices being <400, 400-600, 600-1080, >1000; representing *early morning*, *rush-hour*, *day time* and *evening*, respectively. These groups were chosen with reference to the journey duration distribution (Figure 1); at run-time, they can be dynamically learned using a lightweight technique such as the one described in [6]. Using fewer slots requires less state being maintained about each familiar stranger.

- **User interests** - Each user possesses N different interests selected from the set of all genres. N is selected from a uniformly random distribution in the range [5:15]. This was chosen to reflect the fact the majority of Last.fm user's libraries were classifiable with a small set of categories.

- **Initial library size** - The size of the initial library directly impacts file diversity across the system, with large libraries giving a greater selection of files that could match a user's interest, yet also increasing the likelihood of a user already having a given file. We are aiming to cater for people wanting to find content to entertain themselves, and thus may not already own a large media collection. Each hosts initial library size is thus set as a uniformly random variable in the range [10:30].

All results shown in the next section were obtained with the ATTEMPT_THRESH parameter set to 1. We have run experiments with other values; as expected, when the parameter approaches 0 (i.e., start the download no matter how long the predicted colocation is), a behaviour similar to RANDOM is achieved. For higher values, more downloads complete successfully, thus increasing efficiency, but false negatives (i.e., unattempted downloads that would have been successful) rise too, reducing the amount of files gained. In the interest of space, we limit our discussion to the value 1, thus being able to directly assess the accuracy of our prediction engine.

When two peers lose their colocation while performing a transfer, a failure of communication is registered and a short

wait is forced (of uniformly random [50, 100] second duration) before selecting a different download source. This is to represent the time it would take to register the break in connection, and for a new discovery procedure to be performed. Note that failures are most likely to occur when the train is at a station, and it would thus be wise to wait for all departing people to leave and for any new passengers to board.

Hosts will have to perform regular discovery procedures to detect the colocated neighbours. Data transfer rates are thus reduced to cater for the overhead of performing discovery of length 5.12 seconds every 2 minutes. The estimated transfer value that hosts use when selected, $E(speed)$, is taken as the average value of the range of the transfer rate parameter being used.

When a file transfer completes, rather than performing a new search and selection, another download will be immediately attempted between the same hosts (unless the threshold mechanism indicates it will probably fail). This will reduce the possibility of finding the best source available at some points in time, but it will in turn cut the overhead involved in re-running the discovery protocol, allowing for more file transfers to occur during each 'session' with a selected source; moreover, if a source has one appropriate file, it is likely that they will have more.

Though we do not model wireless propagation effects, many Bluetooth transfers occurring in an enclosed space will impact the achievable transfer speeds. Bluetooth uses Frequency Hopping Spread Spectrum (FHSS) to mitigate interference between physically proximate piconets, changing the transmission frequency every 625μ seconds. The higher the number of active transfers occurring in the immediate area, the greater the likelihood a slot from a neighbouring piconet is transmitted in the same frequency, colliding and corrupting a transmission's slot. We are only considering the data communication, and ignore the effects of capture and signal attenuation. To realistically model throughput reduction due to collisions, we use the analysis presented by Liu [22], assuming all packets are DH5, thus dynamically reducing the throughput of a transfer according to the number of neighbours currently sending data.

5.5   Results

This section analyses the performance results of our approach, with respect to the different source selection techniques outlined before. We have studied how the simulation parameters affect the network, and measured the following metrics:

- **Transfer success rate** - In the range [0:100], it denotes the percentage of initiated transfers that completed. Failed transfers benefit no-one, wasting energy, time and causing extra network contention. It is a measure of the prediction technique accuracy.

- **Library increase** - The mean number of new files gained by each host at the end of the simulation period gives a measure of the utility gained by the users of the system.

- **Efficiency** - The efficiency of the protocol is defined as the ratio of useful data traffic (from complete files) that is received out of the total amount of data traffic received. In the range [0:1], it denotes whether the
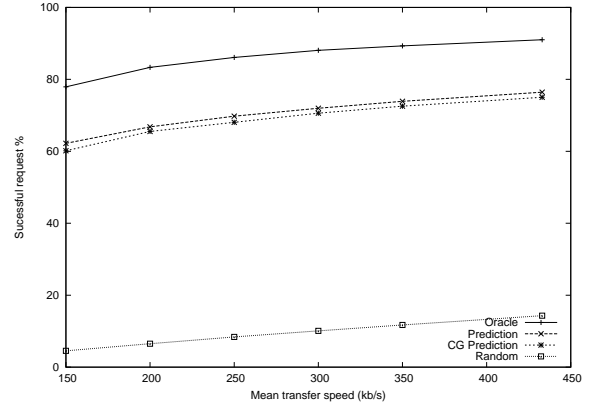

Figure 5: Transfer speed effect.

data distribution network was perfect, 1, or a complete failure, 0. The **transfer success rate** metric does not factor the difference between a failure early in the download, or just before completion. The efficiency shows how much utility (i.e., complete file data procured) is gained for the work expended transferring data.

Our vanilla experiment with the default parameters showed the mean number of familiar stranger nodes (when using our prediction) was less than 200, thus causing a relatively small amount of state (less than 60kB for our fine-grained prediction method) being carried around by the mobile device.

5.5.1   Transfer success rate

Figure 5 shows the percentage of successfully completed transfers while varying the achievable transfer speed, with the file size being set to 5MB. For all selection methods, the success rate increases as the speed does, asymptotically approaching 100%, as transfers become more likely to finish before the source departs. Oracle shows the optimal selection performance, it is not 100% as colocation length alone is not sufficient to know how much information can be transferred in varying network conditions. Our uniform fine-grained time slice prediction mechanism (Prediction) is always significantly higher than the Random approach, at least 50% better, approaching 80% successes at fast transfer speeds. As well as the fine-grained uniform one-hour time slice division, we also tested the performance of our selection scheme when using four time slices within a one day period, carefully chosen using our domain specific knowledge, as discussed in Section 5.1. This more coarse-grained time slice scheme (CG Prediction) suffers a small, consistent penalty for storing less detailed information. This indicates that it is possible to reduce storage overhead, by compromising some performance, when knowledge about movement routines can be learnt from the dataset. Consequently, we only present the results for the fine-grained version in subsequent graphs.

5.5.2   Efficiency

For all approaches, the efficiency of network usage increases with the network speed (Figure 6). At the most conservative end of the transfer speed (100-200kb/s) 57%
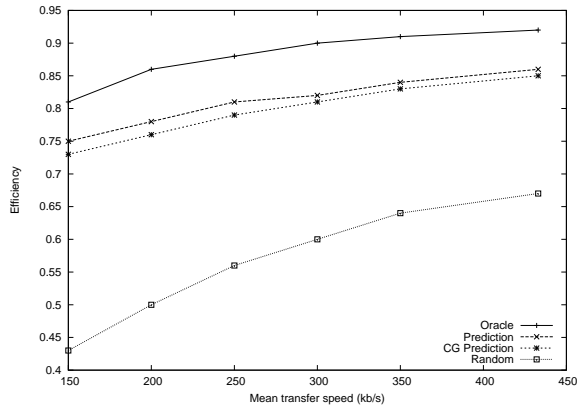
Figure 6: Efficiency of network.

Figure 7: Library size increase.

of all transferred data is not useful when using Random. Wasted work is only 25% for our prediction, even the perfect Oracle cannot avoid some waste, of 19%, due to the false positives caused by the sub-100% success rate. At higher speeds, the efficiency improves for all approaches, with our scheme constantly exhibiting at least 18% better network efficiency than Random.

5.5.3 Library increase

Figure 7 shows how the mean host library size increases roughly linearly as the network transmits the files faster. Random gains the least files, due to choosing poor content sources and its overly aggressive initiation of transfers causing the excess of traffic to collide with neighbours, extending their downloads even further. Oracle and Prediction, due to them employing an initiation threshold, do not cause as much network contention, and subsequently are actually able to transmit more files over the experiment. Prediction enables each host to gain from 5 to 10 more files per host than Random, as network speed increases, with Oracle only finding another 2 files per host, thanks to its perfect colocation duration knowledge.

We have also studied the impact of file size (from 3MB to 10MB mean value) on library increase, with realistic throughput of [200, 300] kb/s. As expected, larger files take longer to transfer, causing smaller library increases for all techniques. Prediction trails Oracle, with Oracle getting on average 2 files more than Prediction almost regardless of file size, while Random remains behind, with between 10 and 2 files less than Prediction, for 3MB to 10MB files respectively.

5.5.4 File receipt distribution

The expected effect on user libraries from running our system is to increase the amount of music they have to listen to. It is important to check that this improvement occurs for all genres, not just the most popular ones. Figure 8 shows the effect on the genre occurrence distribution in the network from using our system. The 'Initial distribution' and 'Gained distribution' are combined in to the 'Final distribution'. As desired, all media categories become increasingly available across the network, with a relative greater increase observed for less popular categories, as the most popular
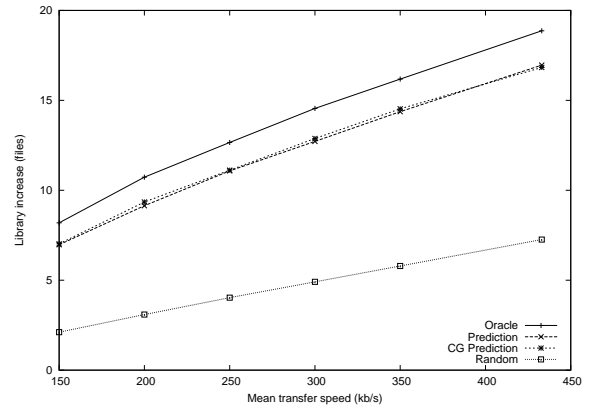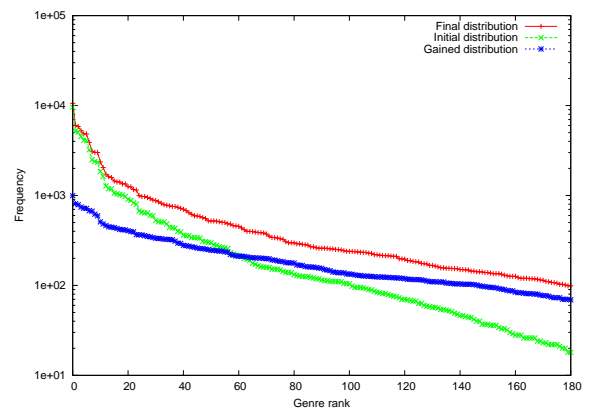

Figure 8: Genre distribution.

ones become so widely available that they are not desired as much as at the beginning of the simulation. Note also how the final distribution still follows a generalised-Zipf distribution: we are thus not altering the natural distribution of files in the system, we are simply making them more widely available, in a way that reflects the natural distribution of user's interest.

Other experiments that are not presented due to space limitations are now discussed. Once the mean number of user's interests exceeds 7, no real benefits are seen in library increase, as it then becomes almost equally likely any given set of neighbours will have some files of interest to a user. The success rate and efficiency metric are mostly unaffected by the mean amount of genre interests, as unattempted transfers, due to a lack of suitable files, do not impact efficiency calculation. Varying the initial library size above 3 files per user has a very small impact on the system's performance, as just a few successful transfers give hosts enough files to satisfy future searches. The choice of the search response shortlist (Section 3.2, step 5) had less impact on system behaviour than expected. Advertising a random selection of files was nearly as effective as advertis-

ing the least shared files (attempting to increase replication). We feel that this would be important in a deployed system with real users, possibly advertising the most listened to files, and so spreading a users tastes through the system, responding to desires for new popular tracks.

6. RELATED WORK

Investigation of of the inter-connectedness of personal devices has received significant interest, with areas such as Pocket Switched Networks (PSN), within the realm of Delay Tolerant Networks (DTN), becoming established [9]. There is much interest in DTNs, using personally carried mobile devices and the opportunistic connections between them to spread information along multiple hops. Whilst being inherently comparable, our work does not focus on routing and/or remote message delivery, but rather the behaviour of pairwise interactions. Knowledge about communities, social interaction and colocation has been employed in multi-hop DTN [17, 10]. With respect to these, we also concentrate on gathering information about human behaviour to inform and improve the operation of our network.

A number of approaches have been developed in recent years to exploit the wireless connectivity of our portable devices and deliver localised content sharing. In terms of Bluetooth network exploitation for media delivery two approaches share a similar vision to ours.

Bluespots [14] is a public transport based content distribution system. Communication occurs via a hub that is installed on the bus, rather than peer-to-peer. Content that is deemed to be popular (e.g., music, news sites) is hosted on the hubs and is made available to the public. Though this centralisation not only causes contention issues, but also restricts the flexibility of what data can be shared and has single points of failure.

Bluetorrent [11] is a peer-to-peer file sharing system using Bluetooth. It is similar in operation to Bittorrent, where files are split into small pieces, then downloaded and shared by clients. Their goal is to support content download over multiple sessions, thus avoiding the problem of independently moving hosts, with short connectivity patterns. APs are used to seed and spread selected content, requiring the creation of this infrastructure and management of the injection of content into the system. The work relies on enough people serving the same version of a file to gain the advantage of *swarming* and includes a communication overhead of informing other peers of individual file progress.

The highly temporally disconnected nature of the networks being considered would require an automatic method to purge partially downloaded files that are unlikely to ever complete. This could be due to rarity of a file, destined to never complete, or even corrupted/malicious files becoming available. We use a different approach, that is, to exploit the regularity of human movement to select a source from where we can reliably download a complete file in a single session, thus only storing complete files. An hybrid approach could be designed where our source selection approach is used to select a peer from whom the most chunks will be received in a single session, reducing the control information communication overhead.

Our scenario of wireless media sharing in dense urban environments is shared by HyCast [2], a *podcast* (syndicated media source) dissemination system. As in our approach, HyCast identifies peers with similar interests and only ex-

pends energy communicating with such hosts. To combat the effects of network contention from many unicast transfers, it forms clusters out of nodes with similar interests. It attempts to deal with transfer failures from node mobility by simply waiting until churn stops before transferring data. This is implausible in environments with constant churn and could result in wasting whatever connection opportunities are available. State is only maintained about other hosts with reference to the content being shared between them, without leveraging any historical information about colocations and their patterns.

7. CONCLUSIONS

In this paper we have presented an approach to content sharing over urban transport. The approach used a mechanism for device colocation pattern detection; such patterns are exploited for the runtime selection of the best content source among available peers. We have described the performance of our approach through the use of extensive underground transport usage traces collected in a large city. The results show that the prediction of human colocation length is a feasible and useful strategy for content source selection. Even in large metropolitan cities, the regularity of people's movement can be leveraged to identify familiar strangers, and exploit the learned colocation patterns.

Though our results have been validated only in the domain of public transport movement, and not other user patterns, many of the attributes of our test data will apply to other human activities, such as coffee-shop or pub visiting. Our findings show that, when automatically sharing content files (e.g., music files and video clips) in human contact networks subject to churn, the source selection stage is important. Aggressive policies on whether to initiate downloads (e.g., Random) may procure more files, but lead to more incomplete transfers, increasing the burden on the network (reducing neighbour's throughput), wasting time that could be spent on successful downloads and needlessly draining device battery-life. Additionally, with some domain knowledge, it is possible to achieve very similar results to our prediction technique, while storing much less state.

The ATTEMPT_THRESH parameter we use can be tuned to effect the download initiation likelihood, and optionally reduce the amount of download failures (at the expense of some successes from false negatives). Its value can be set on a per-user basis and be linked to other factors such as, for example, battery level, causing a device to become progressively more conservative with its download attempts as the available power drains.

8. FUTURE WORK

An aspect we have not included yet in this work is considering a peer's behaviour. Some nodes may, for example, misbehave and share unrequested/corrupted content. When selecting sources, we should thus rely on those that have behaved repeatedly well in the past, as opposed to deceiving/unknown ones. We are investigating the integration of a distributed trust management scheme within the source selection stage, to cope with maliciousness/selfishness. The sharing of recommendations about other host's performance has already been proposed as a means to build trust in MANETs [4]. The combination of knowledge about another host's reputation, together with its movement pat-

terns, could then be exploited to maximise the efficient download of relevant quality files.

We are aiming to increase the amount of platforms the prototype can operate on, and eventually realise a large scale real deployment. This is not only to gather technical data on the operation of our system, but to enable accurate user feedback on what they liked about the system, and what additional features would be most appreciated.

9. REFERENCES

[1] Last.fm - http://www.last.fm, audioscrobbler data source - http://www.audioscrobbler.net.

[2] Adrian Andronache, Matthias R. Brust, and Steffen Rothkugel. Hycast- podcast discovery in mobile networks. In *WMuNeP '07: Proc. of the 3rd ACM Workshop on Wireless Multimedia Networking and Performance Modeling*, pages 27–34, New York, NY, USA, 2007. ACM.

[3] Bluetooth SIG. Core Specification v2.1 + EDR - Specification of the Bluetooth System. July 2007.

[4] S. Buchegger and J.-Y. L. Boudec. A Robust Reputation System for P2P and Mobile Ad-hoc Networks. In *Proc. of the 2nd Workshop on the Economics of Peer-to-Peer Systems*, June 2004.

[5] Dartmoth College. CRAWDAD community resource for archiving wireless data). Available at http://crawdad.cs.dartmouth.edu/, Nov 2006.

[6] V. Dyo and C. Mascolo. A Node Discovery Service for Partially Mobile Sensor Networks. In *Proceedings of IEEE International Workshop on Sensor Network Middleware (MIDSENS07), Colocated with Middleware 2007*, November 2007.

[7] E. O'Neill, V. Kostakos, T. Kindberg, A. Fatah gen. Schiek, A. Penn, D. Stanton Fraser and T. Jones. Instrumenting the City: Developing Methods for Observing and Understanding the Digital Cityscape. In *International Conference on Ubiquitous Computing (UbiComp)*, 2006.

[8] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding Individual Human Mobility Patterns. *Nature*, 453:779–782, June 2008.

[9] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Packet Switched Networks and Human Mobility in Conference Environments. In *Proceeding of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN)*, pages 244–251. ACM Press, August 2005.

[10] P. Hui, J. Crowcroft, and E. Yoneki. BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks. In *Proc. of 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Hong Kong, May 2008.

[11] S. Jung, U. Lee, A. Chang, D. Cho, and M. Gerla. BlueTorrent: Cooperative Content Sharing for Bluetooth Users. *Percom*, pages 47–56, March 2007.

[12] Thomas Karagiannis, Jean-Yves Le Boudec, and Milan Vojnović. Power Law and Exponential Decay of Inter Contact Times Between Mobile Devices. In *MobiCom '07: Proceedings of the 13th annual ACM International Conference on Mobile Computing and Networking*, pages 183–194, New York, NY, USA, 2007. ACM.

[13] L. McNamara, C. Mascolo and L. Capra. Content Source Selection in Bluetooth Networks. In *Proc. of International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous)*, August 2007.

[14] J. LeBrun and C. Chuah. Bluetooth Content Distribution Stations on Public Transit. In *MobiShare '06: Proc of the 1st Workshop on Decentralized Resource Sharing in Mobile Computing and Networking*, pages 63–65, NY, USA, 2006. ACM.

[15] Glenn Lyons and Kiron Chatterjee. A Human Perspective on the Daily Commute: Costs, Benefits and Trade-offs. *Transport Reviews*, 28:181–198, March 2008.

[16] S. Moloney and P. Ginzboorg. Security for Interactions in Pervasive Networks: Applicability of Recommendation Systems. In *Proceedings of ESAS*, pages 95–106, 2004.

[17] M. Piórkowski N. S-Djukic and M. Grossglauser. Island Hopping: Efficient Mobility Assisted Forwarding in Partitioned Networks. In *Proc. of the 3rd IEEE SECON*, Sept 2006.

[18] Nokia. PyS60 Sourceforge homepage - http://sourceforge.net/projects/pys60.

[19] E. Paulos and E. Goodman. The Familiar Stranger: Anxiety, Comfort, and Play in Public Places. In *CHI '04: Proc of the SIGCHI on Human Factors in Computing Systems*, pages 223–230, NY, USA, April 2004. ACM.

[20] R.O. Kharoufeh J.P. Peterson, B.S. Baldwin. Bluetooth Inquiry Time Characterization and Selection. *IEEE Transactions on Mobile Computing*, 5(9):1173–1187, 2006.

[21] Subhabrata Sen and Jia Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. *IEEE/ACM Trans. Netw.*, 12(2):219–232, 2004.

[22] Ting-Yu Lin and Yu-Chee Tseng. Collision Analysis for a Multi-Bluetooth Picocells. *IEEE Commun. Lett.*, 7:475–477, Oct 2003.

[23] A. Varga. The OMNeT++ Discrete Event Simulation System. In *Proc. of the European Simulation Multiconference (ESM'01)*, June 2001.