Colocation Aware Content Sharing in Urban Transport

Liam James John McNamara

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University College London.

Department of Computer Science

January 2010

I, Liam James John McNamara confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

People living in urban areas spend a considerable amount of time on public transport. During these periods, opportunities for inter-personal networking present themselves, as many of us now carry electronic devices equipped with Bluetooth or other wireless capabilities. Using these devices, individuals can share content (e.g., music, news or video clips) with fellow travellers that happen to be on the same train or bus. Transferring media takes time; in order to maximise the chances of successfully completing interesting downloads, users should identify neighbours that possess desirable content and who will travel with them for long-enough periods.

In this thesis, a peer-to-peer content distribution system for wireless devices is proposed, grounded on three main contributions: (1) a technique to predict colocation durations (2) a mechanism to exclude poorly performing peers and (3) a library advertisement protocol. The prediction scheme works on the observation that people have a high degree of regularity in their movements. Ensuring that content is accurately described and delivered is a challenge in open networks, requiring the use of a trust framework, to avoid devices that do not behave appropriately. Content advertising methodologies are investigated, showing their effect on whether popular material or niche tastes are disseminated.

We first validate our assumptions on synthetic and real datasets, particularly movement traces that are comparable to urban environments. We then illustrate real world operation using measurements from mobile devices running our system in the proposed environment. Finally, we demonstrate experimentally on these traces that our content sharing system significantly improves data communication efficiency, and file availability compared to naive approaches.

Acknowledgements

My advisors, Cecilia Mascolo and Licia Capra; throughout my undergraduate degree and during my PhD, their teaching, advice, support and friendship was crucial to my development and achievements. Particularly, Cecilia for the faith she has shown in me and the opportunities she gave.

My family, without who, this thesis would not exist.

My assessors, George Roussos and Per Gunningberg for graciously agreeing to examine me.

UCL and its Department of Computer Science; for being my alma mater and home for over seven years.

Valérie Issarny for giving me the opportunity to study in another country through an internship at INRIA Rocquencourt, Paris.

Graham Frost for his posthumous support and Sylvia Roberts for her belief in my ability to follow his footsteps.

The keen eyes of proofreaders Alan and Dan have improved this work's grammar and presentation. Also, having them as flatmates allowed me to escape the terminal window, though not necessarily technological discussions.

Friends and colleagues: Hugh, Ilias, James, Mirco, Mohamed, Scott, Thomas, Vladimir, Will and many others for their friendship, discussions and support.

Finally, the Engineering and Physical Sciences Research Council for their sponsorship.

Contents

List of Figures

List of Tables

1

Introduction

> Among three men who walk with me, there must be a teacher of mine.
>
> *Confucius*

The increased affordability and gradual miniaturisation of computer hardware has not only created the capability to have personal computers in our homes; now nearly everyone in the developed world carries a portable electronic device on their person. These devices range in functionality from mobile phones and Personal Music Players (PMPs) to highly capable laptop computers. The mobile phone is, in many ways, one of the most rapidly adopted technologies ever invented, with over 4 billion subscriptions in the world today [Int09]. Portable consumer electronic devices are used for communication, entertainment, organisation and indeed computation, with more features and capabilities being developed all the time. The concept of *convergence* has been proposed, stating that all these devices will combine their functionalities and we will be left with a single portable computing device for all purposes. Presently, we commonly have the ability to play media, such as music, and transmit data over wireless links.

It is estimated that a majority of the earth's inhabitants now live in urban areas [Fun07].

The urban population can spend a considerable amount of time travelling to work, school or recreational places, e.g., coffee shops, gyms and pubs [LC08]. While travelling, such as during a daily commute through a city, people enjoy listening to music or watching movies to entertain themselves in what would otherwise be unused time. With other passengers' digital content in such close physical proximity, it would be useful to provide access to it, expanding the playable libraries of all users and so increasing everyone's pleasure and utility.

A significant proportion of portable devices possess short range wireless network interfaces, meaning ad hoc connections can be formed, opening the door to a wide range of decentralised and ubiquitous content exchanges between fellow passengers, including not just file sharing, but interactive games, media streaming and Internet connection sharing. These interfaces become particularly useful when centralised connectivity is expensive or unavailable. The presence of these interfaces on portable devices enables spontaneous, though not always reliable, communication between devices. Despite their lack of administration and organisation, these connections can still be utilised for opportunistic data transmission and content sharing. However, a key issue for these devices is how to decide whom to interact with among the plethora of available peers.

People use their consumer electronic equipment as multimedia computers, especially in developing regions, where home computers are not readily available or affordable. They can currently collect media from many sources, e.g., physical shops, websites, peer-to-peer systems or even manually download from friends. A distribution system for such content, requiring no user management, would not only provide user satisfaction, but would increase content consumption and possibly encourage individuals to participate in and realise the potential of opportunistic networking.

1.1 Hypothesis and Contributions

Automatically performing all steps of user profiling and subsequent content acquisition is fundamental to having a system that will function without requiring a lot of attention from the user. Moreover, by using historical information to accurately predict colocation length, not only can automatically initiated file transfer completion be increased, but more files in total can be shared between users. The method of description, advertisement and selection of files to download from a given peer impacts upon how peoples' tastes are satisfied, as well as performance of the system as a whole. Particular care can be taken to favour replicating uncommon files, ensuring that even users with niche tastes can expand their libraries. The informed selection of a peer to download from (including past colocations and sharing behaviour), as well

as considering the properties of the track to download, leads to a more reliable, resilient and efficient content sharing network.

This thesis proposes a user-centric prediction scheme that collects historical colocation information to determine a long term neighbouring source to use for content download. Content advertising methodologies are also investigated, showing their effect on distribution and availability of both popular and niche material. Ensuring information is accurately described and delivered is a challenge in open networks, encouraging the use of systems such as trust frameworks, to allow the avoidance of devices that do not behave appropriately. Some simple methods for avoiding interaction with hosts possessing poor historical records are also demonstrated. We validate our assumptions and approach on synthetic and real datasets, particularly traces that are comparable to urban transport environments. We then demonstrate experimentally on these traces that our content sharing system significantly improves application data communication efficiency, and file availability compared to naive approaches.

The following contributions are presented in this thesis:

Peer Selection – We propose a method for estimating the length of a short-range wireless neighbour's future connectivity duration using historical information. The method requires a low amount of state storage (less than 0.5KB per host) and does not require significant computation for each prediction. It uses a matrix to store user specific and temporally relevant information about previous durations. A simple lookup to this matrix gives an estimation of a user's predicted duration; if no user history is present, then user agnostic temporally relevant information is used. Using these predictions to select download sources is compared against a random approach and perfect future knowledge of user movement to examine its comparative efficacy. Previous work has attempted to predict link duration for network stability, but applying this to specific mobile peer-to-peer transfers is novel.

Advertising Strategies – We present an item advertising/ordering strategy for hosts in an opportunistic sharing network. The manner that subsets of a user's library are described to others affects which tracks are selected for download. The advertisement's item ordering process and its impact on the movement of files through the network is examined, we specifically examine how this satisfies the different music tastes of users. We show how the order that (un)popular files are shared between hosts, when the amount of files that can be shared is limited, has a large effect upon the distribution of files in user's libraries. Moreover, that if particular care is not taken to prioritise unpopular files, they will be-

come completely marginalised in the network. File sharing choice has analysed in many contexts, though not with concern for the overall popularity distribution.

Music Tastes – A measurement study of 10,000s of users' music tastes from a popular social music website. The resulting dataset gathered the top 50 artists listened to by users over three different time frames. All the artists then had their associated *tag* information collected, which had been applied by users. This large body of information facilitated the construction of a model for users' digital libraries and music tastes. The music taste models allow informed investigation of the various advertising policies and file selection procedures when applied to realistic user libraries and taste distributions.

Prototype Deployment – An application for mobile phones that implements the ideas and algorithms is presented in this thesis. This application was operated in actual urban transport conditions and its behaviour is timed, recorded and analysed. It was tested during rush hour and quiet periods to see how its behaviour would be affected by the environment. We are not aware of any previous work experimenting with mobile peer-to-peer in real life subway systems.

System Evaluation – Using details gathered from the prototype we then performed a large-scale simulation of the proposed system. The user study was used to create realistic user tastes/libraries and the peer selection and item advertising algorithms were applied to them. Mobility was simulated using real-life traces from a variety of urban and sub-urban situations. Parameters of the input and algorithms were varied to examine the system's operation in a range of environments. It is demonstrated that careful consideration of a download source is important to ensure successful file transfers. Also, the order of items being shared between peers has a decisive impact on the evolution of users' libraries. Furthermore, we show how systems such as ours naturally affect the balance of popularity in multi-category systems and how to limit this possibly homogenising effect. The metropolitan city scale of the simulations is a particularly novel aspect of this work.

1.1.1 Assumptions

Many assumptions and generalisations have been made to facilitate meaningful analysis of the problems and hypotheses. The most important and overarching ones are presented below, with more specific assumptions introduced in the relevant sections.

- All content files are encoded in an open or commonly understood format, and thus may

be utilised by any device. This is simply to avoid any potential problems of some items being indecipherable to some nodes.

- Device battery power is not modelled, nor its effect of causing devices to stop functioning. However, attempts are made to not needlessly waste battery power. It is assumed that users will always try to keep their battery power from running out by charging it when possible (meaning a device only has to last while they are away from home).

- A user's music taste does not change significantly over the time frames we are considering. Equally, users do not frequently manually modify their library, by either adding to or removing from it: changes only occur from peer-to-peer transfers.

- Users' devices do not perform any other significant activities over the short range wireless interfaces, such as other bulk data transfers when performing device synchronisation. This stipulation avoids the requirement to model other user actions on the devices in question.

1.2 Constituent Papers

Part of the research presented in this thesis has been published in the following papers.

- **Media Sharing based on Colocation Prediction in Urban Transport**
  Liam McNamara, *Cecilia Mascolo* and *Licia Capra*
  Presented at **Mobicom 2008** San Francisco, CA, USA [MMC08].

- **Content Source Selection in Bluetooth Networks**
  Liam McNamara, *Cecilia Mascolo* and *Licia Capra*
  Presented at **Mobiquitous 2007** Philadelphia, PA, USA [MMC07].

- **Trust and Mobility aware Service Provision for Pervasive Computing**
  Liam McNamara, *Cecilia Mascolo* and *Licia Capra*
  Presented at **Requirements and Solutions for Pervasive Software Infrastructures (RSPSI)** (Colocated with **Pervasive 2006**) Dublin, Ireland [MMC06].

1.3 Related Work

A number of approaches have been developed in recent years to exploit the wireless connectivity of portable devices to deliver content files. In terms of short range network exploitation for media delivery, the following approaches share a similar vision to ours.

Early work in this field, from the Haggle project [LLS+06] collected data about colocation and inter-colocation in an urban setting. It applied the Delay Tolerant Network (DTN) concept to urban settings. DTN is is an approach to networking architecture that attempts to dispense with the requirement for end-to-end connectivity [Fal03]. *Bundles* of data are passed opportunistically, hop-by-hop toward their destination, hence information can be asynchronously sent between hosts that never share a complete path connecting them. The Haggle system uses real devices in a deployed experiment to monitor realistic human mobility; the architecture also includes more advanced considerations for privacy, authentication, trust and advanced data handling. Its focus was on feasible delivery ratios for bundles rather than actually performing data transfers between devices. The relatively sparse nature of the results (due to the testbed size and low density of the urban test area) differs from our proposed scenario.

Bluespots [LC06] is a public transport based content distribution system. Communication occurs via a hub that is installed on a bus, rather than in a peer-to-peer manner. Content that is deemed to be popular (e.g., music, news sites) is hosted on the hubs and is made available to the public. 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 [JLC+07] 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 amongst clients. Their goal is to support content downloads over multiple sessions, thus avoiding the problem of independently moving hosts, with short connectivity patterns. Access Points (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 ad hoc networks 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, and so only storing complete files.

A similar project is the *Push!Music* system [JRHH05], from the Viktoria Institute Future Applications Lab. In this work a thorough, yet small scale, user feedback session is performed [BWG+07]. Push!Music provides a distributed content dissemination system on handheld devices; it focuses on being a system that gains or enriches social interactions. They deployed the system upon the devices of thirteen subjects for three weeks, and held group

feedback meetings to determine how such a system satisfied users. Users could give manual feedback on what they liked/disliked through the application interface. Despite the system being technically able to automatically share music if user tastes are sufficiently alike, this feature was not investigated in the study. This was apparently due to the users not playing enough music for the system to identify similarities. Therefore, their collaborative filtering (see Section 4.1) mechanism could not reach decisions about which files should be shared, an issue likely compounded by the small testbed size. The users expressed a strong desire for automatic sharing as it would require less attention on their part and was seen as 'exciting'.

Rather than performing bulk transfers of music tracks, *tunA* [BMA04] allows live streaming of the music being listened to on one device over to another. This gives a much more immediate, socialised experience of sharing music between people. It has the benefit of creating relationships between users, and could feasibly initiate friendships between people that would otherwise not have spoken, if they were impressed enough to pay attention to the source of their music. The restriction of only being able to hear what is currently being played reduces the probability that music of some interest can be found. Also, bulk data transfer allows music to be collected at speeds faster than real time, allowing the consumption of collected music at leisure at a later date.

1.4 Thesis Outline

This section gives a brief description of each subsequent chapter:

Chapter 2 – *Motivates* this work, describing the usefulness of this research, the state of the art in the field and the envisaged scenario, followed by the challenges it presents.

Chapter 3 – Describes human movements and *colocation prediction* techniques developed over the course of this work. It also introduces the notion of *trust*, and how it can be dynamically updated and used to influence the selection of a source peer.

Chapter 4 – Discusses the methods for generating the content libraries and tastes, and how to *advertise content*.

Chapter 5 – Presents the various synthetic and traced movement *datasets*, together with the techniques used to collect and utilise user tastes from a popular social music website.

Chapter 6 – Describes the methods used to *evaluate* this work, particularly the simulator behaviour, together with the *results* collected from these experiments. Also presented is the *implementation* created to demonstrate this work, together with some of the technical issues that revealed themselves during the development.

Chapter 7 – A model of the types of system we are considering is presented in this chapter, together the model's performance when different advertising policies are used.

Chapter 8 – Summarises findings and draws *conclusions* from them, finishing with a discussion of the implications for the field of opportunistic data sharing.

2

Motivation

The reasons for undertaking this work, together with the foreseen problem space, will now be presented. When building a system facilitating content delivery between members of the general public, many factors need to be considered. Firstly, we discuss which devices people possess and what capabilities for exchanging/consuming content these devices have. Secondly, we consider how people's movements affect the possible wireless interactions between devices and how these challenges can be overcome (covered in Chapter 3). Lastly, the categories of data and interests that users have (investigated in Chapter 4) are presented, a point that is often overlooked in previous data sharing research.

2.1 Motivating Scenario

In this section we describe a scenario where mobile automatic content sharing could be employed, and consider some of the required behaviour of such a system. Commuters commonly move 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, Bluetooth). They are looking for music of interest to them, such as specific genres, e.g., *rock* or *pop*. A city's transport system provides an environ-

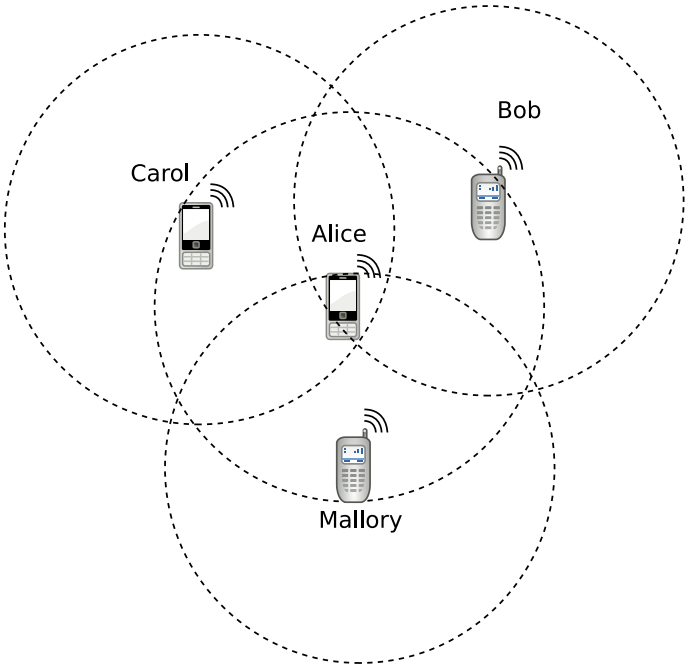

Figure 2.1: Scenario Diagram.

ment for significant mixing between people, giving the possibility of meeting people with very similar tastes, or even new and interesting ones.

Imagine *Alice* is a commuter (seen in Figure 2.1), travelling to work along her usual route, at her usual time, upon a rapid transit system train in a large city. She uses her phone as a music player for entertainment while travelling. When in the transport system, she is unable to connect to the Internet as there is no cellular network coverage (or because the connection cost is too high). However, the large number of other passengers travelling with her offer a means of procuring media for her enjoyment: during periods of peer-to-peer connectivity, devices may transfer music tracks or video clips to her. 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 start at work in the city centre at 9:00AM. He enjoys listening to *rock* and *metal* 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*. *Mallory* is a fourth traveller, he embarks with Alice as he has often done before, yet any previous time he was used as a source the files were mislabelled/poor quality, he claims to like *rock* and *pop*. All mentioned devices are within Bluetooth range and are potential sources of content for Alice, as they carry genres of interest to Alice (in particular, rock from both Bob and Mallory or electronica from Carol).

Alice's device now has to select which user to download from.

If the content size were very small and could be transferred within a few seconds, the choice of whom to download from, among those having relevant content, would not be critical. However, when sharing media content (e.g., high quality music files or video clips) that require minutes or longer to be transferred, it becomes important to select a source that will remain colocated long enough for the download to complete. If the predicted colocation duration with Bob, computed based on their 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 within the next minute), and thus she should not be favoured. Independent of the colocation history with Mallory, the fact he has given many bad files to Alice in the past means he should be dismissed from consideration.

Once Alice has chosen Bob as the source, she must decide which file to request from him. She sends a request detailing what she is interested in receiving. Bob will respond with a list containing files (artist and track names) out of his collection that he believes will be most suitable. Alice can then perform a request for one that she does not already possess, and cache the list of others that she may want in future from him. Any files in this list that Alice owns need not be advertised back to Bob if he ever asks, as she knows he already has them; he also need not re-advertise them. Through this process they can learn each other's libraries and become even more useful data sharing partners in future.

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 a large proportion of people leaving and boarding the train, especially at large stations with many connecting lines.

2.2 Users' devices

Though there is a huge amount of heterogeneity in mobile computing equipment, some common features unite a large proportion of them. We are considering small personal devices that have the ability to play media files, possess considerable storage for them, as well as wireless capabilities to transport them. The reduction of cost and physical size of digital storage has made devices with gigabytes of storage commonplace, and extensible modules such as Secure Digital (SD) cards make voluminous flexible storage a reality. Current portable consumer electronics can already store days' worth of music: as of 2009, SD cards can store up to 32GB. So whereas previously only a few audio files could be carried around, it is now feasible to hold a

person's whole music collection or a set of videos. This excess of, possibly unused, space offers somewhere to store more files for a user's entertainment at negligible additional cost.

2.2.1 Network Capabilities

Wireless communication range is limited by the signal's transmission power and its consequent dissipation, which is heavily influenced by environmental factors and device placement. There are many wireless standards with differing aims and technical requirements. A list of some popular wireless technologies available for mobile devices and their approximate ranges follows:

- **Near Field Communication (NFC)** – 15 centimetres, using magnetic induction in the range 13.56 MHz. It is a recent technology increasingly being built into phones for uses such as small monetary transactions.
- **Bluetooth (Class 2)** – 10 metres, using radio transmissions in the 2.4 GHz ISM band. Extremely popular standard, designed to wirelessly replace RS232 serial cable links for low power devices.
- **WiFi** – 100 metres, radio transmissions in either the 2.4 GHz or 5 GHz frequency bands. An umbrella term for wireless Local Area Networks (LANs) based upon the IEEE 802.11 standards. It is usually limited to inclusion on more feature rich devices, due to its significant power requirements.
- **WiMAX** – 50 kilometres, using radio transmissions in the 2 – 60 GHz frequency range. The WiMAX forum described it as *last mile wireless broadband access as a replacement for cable and Digital Subscriber Lines (DSL)*. This large range also requires significant power to operate.
- **Cellular Infrastructure** – Though transmission is limited to tens of kilometres, complete coverage (with inter-cellular handoff) can be assumed in urban areas.

If content is to be shared between people, possibly strangers, a wireless standard with the range of at least a few metres is required, to avoid having to be extremely physically proximate. The disruptive effect of mobility would be reduced by using a long range protocol, making disconnections mid-transfer unlikely. Unfortunately, long range technologies have drawbacks that can make them unsuitable for ad hoc bulk data transfer. The greater the communication range of devices, the more other devices can be overheard also using the medium. In an environment with a high density of users, being able to reach even more distant users may not provide any additional advantage. In fact, with multiple concurrent transfers occurring in the network

neighbourhood, there will be greater contention, limiting the achievable data rates. Hence, if some suitable content is already available nearby, it can actually be detrimental to increase the communication range. This concern has been addressed in the area of mesh networks, through the process of *topology control*. One solution employs dynamic power control to ensure transmissions are only as powerful as they need to be to reach the next hop [LZZ+06]. Though we will not discuss dynamic power levels any further, it is important to consider what effect the choice of wireless technology, and its range, will have.

Devices may have access to infrastructure through cellular data interfaces such as GPRS/HSPA. They allow network connectivity to communicate with non-local devices, and usually the Internet as a whole. Connectivity to a huge number of hosts is one of the reasons the Internet has become such a dominant force in modern life. Future cellular standards can theoretically approach current wired broadband speeds, with multi-megabit downlink bandwidth[1]. Though the thought of using cellular networks to allow mobile devices to act as any other Internet node is attractive, it suffers from some important limitations. Consumer packages that charge for transferred data volume are often prohibitively expensive, whereas flat fee offers usually impose limits on the usage [Peh07]. These range from data volume limits to explicit banning of bulk peer-to-peer traffic in the *terms of service*. Service limitations by commercial wired Internet Service Providers (ISPs) are common [DHGS07], and can be even more important to cellular providers, due to the technological constraints of the network type. The sharing of the cellular medium by so many users, and the operating costs of such hardware limits the economic feasibility of allowing many users to perform bulk data transfers cheaply. There are often some business concerns about allowing *Voice over IP* (VoIP) on a network that gains most of its profits from transporting voice data, often at comparatively much higher cost to the user. A more obvious limitation of using infrastructure is the requirement that the infrastructure provides suitable coverage and capacity. Remote places may possess no cellular coverage or reduced data service due to lack of subscriber density. Conversely, urban areas also present environments with no coverage, such as underground (e.g., subways and basements) or within large buildings.

The use of device-to-device wireless links does not suffer from requiring the presence or permission of a third-party to provide the network link. This removes the ability to easily monitor or censor the transfers between peers, possibly an increasingly important issue with US Telecom companies performing mass surveillance of the public, and UK ISPs secretly profiling users for advertising purposes. The 2009 political unrest in Iran lead to the suspension

[1] 3GPP Long Term Evolution (LTE) envisages around 170Mbit of downlink shared across 20MHz of spectrum.

of cellular networks[2], demonstrating how availability and cooperation can not always be expected from infrastructure. Another possibility for gaining connectivity to the Internet would be through wireless APs, such as commercial, municipal or private WiFi access points. Despite the obvious problems of connecting to static APs while on the move, an issue tackled by Breadcrumbs [NN08], there are also financial and legal problems with performing bulk data transfers over APs that are not your own. Again, connectivity does not always translate to an ability to perform bulk data transfers.

Due to the wide availability, we have chosen to motivate our work by assuming Bluetooth network interfaces. Though this causes some loss of generality, Chapter 3 details techniques that can be used irrespective of link speed. The prevalence of Bluetooth interfaces on modern devices is well known. The Bluetooth Special Interest Group (SIG) claims over two billion active devices. Class 2 Bluetooth devices have a range of 10 metres, which is suitable for communication between devices in relatively stationary proximity, but will not necessarily provide stable connections during free movement. In urban areas a lot of time is spent in close proximity to many other people, e.g., when in mass transit systems, offices or commercial districts. Most connections formed during these periods are highly transient, and are prone to being particularly unreliable. They vary from many short connections (people passing by in the street) to a few long ones (friends travelling/shopping together). Devices would obviously benefit from identifying the colocations that are not expected to last long, and avoiding setting up short connections that would lead to failed (incomplete) data transmissions. The physical locality of short range links provide an implicit method for presenting a subset of peers as suitable sources of content.

2.2.2 Usage Patterns

People's Bluetooth interfaces are increasingly being left switched on. A study [Eam06] performed in 2006 in a small UK city demonstrated around 10% of the public carry discoverable wireless devices, a figure likely to increase with deployed Bluetooth device numbers and potential benefits from leaving it switched on. Also, as battery technology improves and lower power versions of Bluetooth become available, the penalty for leaving it on decreases. Bluetooth technology is spreading to Personal Music Players (PMPs) for use as a headphone connector or manual data transfers, such as the *StormBlue A9+* and *Insignia NS-DVB4G* devices (similar to phones gaining music playing features). This will be an even more attractive possibility when

[2]Reporters Sans Frontières: *News and Information Fall Victim to Electoral Coup*. Website (accessed June 2009): `http://www.rsf.org/News-and-information-fall-victim.html`

the Ultra Low Power Bluetooth standard (previously WiBree) becomes widely available. 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.

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 reported that, in 2006, the average commuter living in large 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 [LC08]. 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* was coined by Stanley Milgram [PG04] to designate someone that we may often see but do not personally know. In fact, one of the original experiments for the paper was performed at a train station. The concept of the *familiar stranger*, is a useful property for the creation of digital relationships; as computers do not have the social barriers between each other that humans do. They can exchange information even on a first meeting, and store historical data about other hosts indefinitely.

2.3 State of the Art

2.3.1 Commercial

Wireless file transfers between consumer electronic devices have been possible for years, using popular protocols such as Bluetooth and WiFi. Bluetooth has been available on mobile phones for 10 years, since the release of the *Ericsson T36*. The legacy standard for *IEEE 802.11* was originally released in 1997 [Gro97]. Both of these popular standards allow short-range connections to be made between electronic devices. They are however very different in their behaviour, Bluetooth was aimed at small Personal Area Networks (PANs), whereas WiFi has been described as “wireless Ethernet”.

Bluetooth constructs star-shaped *piconets* of up to eight hosts, with the central one being the *master*, maintaining control of all connections in the network. Radio transmissions are performed using *Frequency Hopping Spread Spectrum* (FHSS), where data is transmitted over a pattern of 79 different frequencies. This spectrum spreading gives resilience from interference, and allows other piconets to operate in the same physical space with minimal contention; as it is unlikely they will choose the same frequency at exactly the same time. Over the past 10

years, Bluetooth has been included in a myriad of electronic hardware, including phones, PDAs, audio headsets and computer mice/keyboards. This huge deployment makes an abundance of interactions possible.

The release of the *Microsoft Zune* PMP brought a new dimension to the field of electronic media players. Their WiFi network interfaces allow communication of data (such as music files) between Zune devices. The use of *Digital Rights Management* (DRM) technology allows possibly copyrighted data to be shared without infringing the ownership rights of the copyright holder. DRM restricted music is useful for music distributors as it allows their product to be sampled by people, encouraging them to then purchase it. Besides DRM music, several other data types are freely available to share, such as sample clips, movie trailers, and Creative Commons [Cre09] licensed data.

The *Zune* demonstrates the direction of modern PMPs and how our scenario can be a driver for media industry sales. The Zune contains a WiFi interface that allows the devices to send content files between each other, so users can listen to other people's libraries. There are many restrictions placed upon this behaviour though:

- A non-standard form of WiFi is used, restricting communication to only other Zunes or APs to connect to the Zune Marketplace website, where more music can be bought.
- Content downloaded from other Zunes is restricted to three plays, after which tracks must be purchased from their marketplace to be heard again. This is enforced by wrapping files that are shared with DRM restrictions.
- File transfers must be initiated manually, with the finding, sending and receiving performed by users having to interact with their device.

These features show how there is industry interest in allowing electronic devices to share music content, and even that it does not have to be detrimental to their business. This has been termed *Superdistribution*, which involves the encouragement by copyright holders for content files to be freely shared as much as possible, while still retaining control over the actual playback of the files [MK90].

Crucially, the aim of this research is to thoroughly investigate the feasibility of a system that will work with current non-specialised hardware without the requirement of any special infrastructure support, so that it can be deployed in the real world now. The target devices have been available for a few years in developed regions, and so have a large deployed base; they will also be widely available in developing regions at least in the next few years. The content types being shared will not be limited by the system, and any device with the technical

capabilities will be able to participate. This is not to say that technological advances will be unable to increase the usefulness of the proposed system, with very low power Bluetooth soon to be added to the standard and storage becoming even cheaper.

2.3.2 Academic

There has been much research in recent years into data sharing between mobile wireless hosts. If we are explicitly aiming to share media content, such as music, then some extra assumptions about the properties of such a sharing system can be made. The data being shared is relatively static, a high quality encoding of a popular artist's discography need only be updated on release of a new album (usually the scale of years). New media will be gradually added to the global library, as albums are released by artists, though this is proportionally very small. Each item of content can be classified by many categories, such as *rock*, *dance*, *live*, *80s* or *heavy*; these categories will henceforth be referred to as *genres*. Possession of a partial file is of no interest to a user, a person owning half of their favourite track is likely to be annoying, rather than almost as useful as the complete track.

In a large distributed network, locating files of interest is one of the main challenges. The lack of centralisation means there is no single place to store an index of files. Many approaches overcome this limitation using a structured distributed information system, such as a Distributed Hash Table (DHT) [SMLN$^+$03] or information summary trees [JX07]. There has been investigation into the applicability of DHTs to MANETs [HGRW06]. One such approach [GM03] is a DHT based mobile peer-to-peer system that focuses on reducing the network overhead.

Such structured storage and retrieval of information can be very useful, particularly for stable multihop wireless networks, where users desire a particular file and need to find its location. Unfortunately, they also require maintenance of the virtual structure, this burden increases as networks become more dynamic and suffer from greater *churn*. This maintenance will have to occur even if no hosts in the network are attempting to find/acquire any files. The navigation of information *overlays* can cause needless communication with hosts that neither possess the file, nor know where it is. Also, as with all distributed networks, partitioning may occur, compromising the structure. To lookup the files in an overlay system, a *key* must be available to map to the file, and this key must be available for hosts to perform a search for the file. This requirement of already possessing a key for desired files limits the possibility of serendipitous collection of files. Usually this key is formed from search terms that a user will manually enter. The behaviour of overlays storing files at particular (key defined) locations is ill suited to file distribution, as it actually only aims to provide a method of file storage/location in the network.

Files (or indexes) having to be moved/stored around the network just to satisfy the structure, and not user desires, could lead to much wasted communication.

Files may also be found in a distributed wireless network by unstructured means, such as random walks, expanding ring searches (ESR) or even just flooding requests [PKGO09]. These approaches are much simpler and more robust to network dynamism and node failures. They are also reactive, in that they only initiate a search, and so cause traffic, in response to a desire for a file's location. Our scenario (covered in Section 2.1) is one of high churn and relatively small, though frequently changing, groups of users; it would be more suited to an unstructured information system.

Investigation of the inter-connectedness of devices has received significant research interest, with areas such as Pocket Switched Networks (PSN), within the realm of Delay Tolerant Networks (DTN), becoming established [HCS+05]. There is much interest in DTNs, using personally carried mobile devices and the opportunistic connections between them to spread information along multiple hops. Some work has focused particularly on content dissemination in such networks [SMM07], using explicit subscriptions by nodes to register their interest in content types. Whilst they are inherently comparable, our work does not focus on routing and/or remote message delivery percentage, but rather the behaviour of pair-wise interactions. Knowledge about communities, social interaction and colocation has been employed in multihop DTN [SDPG06, HCY08]. With respect to these, we also concentrate on gathering information about human behaviour to inform and improve the operation of our network.

Many of the mentioned mobile peer-to-peer systems assume that multihop communication is achievable and desirable. Being able to communicate with non-neighbouring hosts, by multihop forwarding, is preferable if access to many hosts is required. However, if successful interactions can be performed by one hop neighbours, then forwarding becomes a needless requirement upon all hosts. Intermediate nodes on a forwarding path are burdened, by having to store and forward data for others, rather than satisfy their own user's interests. In fact, as evidenced by Guptar & Kumar [GK00], when uniformly distributed multihop wireless networks grow in size, then the achievable capacity of arbitrary paths decreases to zero. Forwarding data along multihop paths causes more radio interference than if it was simply sent directly between two hosts. Further work [JLM01] showed that if the average distance between source and destination decreases as a power law, then the achievable path capacity can stay constant. This implies that keeping paths locally bounded is the only way to allow such networks to function. Also, the existence of host movement in a wireless network, while causing link instability, allows for an alternative method of increasing capacity. If hosts retain transmissions for others

and wait until their movement brings them closer to the destination, then arbitrary destinations can be communicated with, at the cost of delay [GT02]. Our scenario does not require communication with remote hosts as well as being unaffected by delay, hence these problems of scalability are avoided. If hosts are having to spend some of their time forwarding data for others, then there becomes a much greater likelihood of their own transfers not completing in time over transient connections. Even interference/contention from neighbours will increase transfer time due to reduced network performance. Experimental results [RNGT05] show that even small scale multihop configurations can suffer significantly when multiple workloads are active.

The requirement for hosts to forward data for others adds the problem of *selfish* nodes possibly not fulfilling their duties. Enforcing a forwarding requirement leads to the introduction of an incentive scheme, such as a trust or reputation system (covered in Section 3.5).

If hosts are operating in a wireless ad hoc environment, then negotiation of interactions over arbitrary paths can be problematic in itself. For example, multihop ad hoc WiFi communication would either require that all hosts communicate upon the same channel (causing needless contention), or alternatively, have each host searching all channels for other hosts to provide forwarding. Joining Bluetooth piconets together into arbitrary sized *scatternets*, while theoretically possible, has had very few implementations due to technical limitations. Due to the master/slave architecture, all communication in a piconet has to be between a master and a slave device, creating bottlenecks at all masters. If a piece of data needs to traverse many piconets, it must pass through all masters on the way, and also have some devices participate in more than one piconet, requiring a change of FHSS sequence (the sequence is defined by the piconet's master). In addition, there is the issue of creating a system of routing on top of this ill-suited architecture.

Our vision of wireless media sharing in dense urban environments is shared by HyCast [ABR07], which is a *podcast* (syndicated media source) dissemination system. As in our approach, HyCast identifies peers with similar interests and expends energy communicating with such hosts only. To combat the effects of network contention of 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. May *et al* [MLKW07] also focus on podcasting in MANETs, though they

use WiFi as the network type. This work includes experimental testing of a prototype implementation, and shows the feasibility of wireless bulk sharing of media files. However, it also shows how download times will increase, to possibly unworkable lengths, as the number of participating nodes increases. Undersound [BBM+07] is a proposed system to gather music when using urban transport and has quite a similar end user experience, however the internal operation is drastically different to ours. It requires support from infrastructure installed in the transport system, which in underground systems could have very poor coverage and consequently require a lot of deployment overhead. The system considers each station in a transport system to be a repository, with files being submitted to a station by users. Music files can also be shared directly between users, but all files have to originally come from a station's repository. Downloads are initiated manually by users to promote engagement with the process of collecting music. There is more of a focus upon creating a community specific experience.

2.4 Challenges

Traditionally, finding 'who to interact with' has mainly been seen as a problem of understanding who is offering the service required. Most service discovery and selection frameworks, developed for both traditional distributed systems and ubiquitous systems [Edw06], focus on how to describe services, how to formulate and spread queries, and then match queries with service descriptions [LI04b]. However, in this formulation of the problem, the ad hoc nature of pervasive interactions is not taken into account, in particular the following challenges have been overlooked. First, how to identify, among the providers offering a desired service, those that are likely to be connected long enough for the service provision to complete. Second, which files would be most appropriate for the recipient and would lead to the recipient being able to serve others in future. Thirdly, how to select trustworthy providers that will actually deliver the service/content as promised.

The construction of a wireless content distribution system in the given scenario presents many challenges and problems. Some of these issues are socially based, requiring sufficient penetration of the public to allow enough opportunities for data sharing. The technological challenges must be overcome, or at least mitigated, before large-scale adoption could begin. The main technological issues are now presented:

Discovering neighbours – Finding nearby hosts that are participating in the content distribution and are able to share media files. Though this feature is inherent in all Bluetooth devices, it is not a truly reliable process [Kha06]. In fact, the more frequently that hosts try to detect one another, the harder it becomes to be discovered by others. A host can

only be discovered if it is not trying to perform a discovery itself.

Choosing a peer – Selecting someone as a source whose taste indicates they will be able to offer some non-malicious tracks of interest. Furthermore, that they will be able to successfully complete the transfer of one of these files. The discovery and source selection is addressed in Chapter 3.

Exclude malicious hosts – Once a file has been transferred, it should be checked to ensure that it is a valid media file and, more importantly, the one that was requested. If a host provides too many tracks that are found to be wrong in any way, it should be excluded from consideration as a data source. Mechanisms used to decide whether to exclude a host are given in Section 3.5.

Choosing a track – Maximising utility for a user (the suitability of the music) and system as whole (ensuring appropriate replication) from the procurement of a track; discussed in Chapter 4.

Assessing viability – Deciding whether to proceed with the transfer of a particular track. If a file will take longer to download than the predicted colocation time, then it should probably not be initiated, as this will just lead to wasted effort.

2.5 Summary

This chapter presented our motivation for undertaking this work. The capabilities of modern mobile devices were described, particularly widely deployed wireless network interfaces. The commercial and academic offerings for systems that provide features that would be required by our proposed system were also investigated. Importantly, a likely scenario where a system such as ours would be used was described, including the operating environment and resulting system behaviour. Lastly, the challenges presented by this scenario, that will need to be overcome, were listed.

3

Peer Selection

In this chapter we discuss the challenges presented by opportunistic networks, and some techniques that can be used to mitigate them. For highly dynamic wireless networks, the constantly changing structure can cause many problems when attempting to achieve reliable communication. The undetermined duration of colocations means communication is unreliable and achieving successful data transfers can be problematic. Time required to transfer data is not known a priori, due to variable transfer rates from the unstable wireless environment. In order to maximise the chances of completing a download, as much time as possible should be available for performing the transfer. Therefore, choosing a neighbouring host that will be present for a sufficient time for a download to complete is mandatory. Being able to determine the duration of the neighbours' colocation becomes an important consideration for performing successful data transfers. We will present some techniques for estimating this colocation duration using historical information of the local network state. If downloads are to be initiated automatically, without users intentionally staying proximate to the file source, then many downloads may not be completed before the transient connection breaks. Devices should identify stable neighbours who will be colocated with them for sufficient periods for a download to complete.

The process of selecting a download source host from a set of peers is a problem that has

received much attention on Internet deployed systems. Possibly the most popular file sharing system currently in operation is *BitTorrent*. It is designed to overcome the distribution load/costs for content providers by allowing downloaders to combine their efforts when attempting to procure a file. The capacity of paths between Internet hosts is often less than the total capacity of a host's immediate upstream link, meaning downloading from many hosts is likely to use more of their connection capacity. In wireless networks, this subdivision of file transfers allows files to be gathered across multiple time periods rather than from multiple sources. In fact, concurrently downloading from multiple sources in wireless networks can be detrimental due to contention in the shared communication medium.

Rather than enabling partial transfers over short lived links [JLC+07], we aim to ensure that selected links exist for long enough to perform a whole transfer. A swarming-like approach is orthogonal to ours, and could be combined to gain the benefits of both. Stable links are of advantage to many other uses in short range wireless networks, and so still represent an objective benefit. Possible applications that would benefit from identifying stable neighbours, or at least avoiding unstable ones, include multiplayer games, Internet connection sharing and music streaming. The rest of this chapter contains a discussion about how device movement affects wireless networks and a section presenting some aspects of the relationships that form in urban places. Methods of opportunistically selecting a peer to communicate with are also presented, together with the formalisation of the algorithm. The second part of this chapter will discuss some of the problems presented by poorly performing devices and some of the techniques that can be used to limit their effects on system as a whole. The nature of computational trust is discussed, followed by how to store and access this information. Finally, the mechanism for dynamically updating this information in response to (un)successful interactions between peers is presented.

3.1 Dynamic Wireless Networks

Neighbouring devices that can be wirelessly communicated with are a product of a person's physical location. Over time, the detected devices are a product of the person's movement patterns. The resulting networks created from these devices can be very dynamic, with the available neighbours possibly changing many times a minute in dense urban environments. Bluetooth networks only function over short distances (typically 10 metres), they are subject to intermittent and unreliable connections, caused by electronic interference and physical mobility. The changing topology of a Bluetooth network will significantly affect its operation, dictating which hosts can communicate with each other. A device's underlying physical movement cre-

ates the resulting pattern of colocations (when combined with all other device's movements): two hosts moving away from each other will quickly lose contact. Understanding these movement patterns is important for designing a system that can still operate under such stresses.

In everyday life people often follow *seasonal movement* patterns, regularly travelling very similar routes and visiting the same places (e.g., the same journey to work, visiting a coffee shop, going to their local pub, etc.). People possessing similar destinations and journey patterns will be more likely to be regularly colocated with each other. A device's connection history could thus be used to infer future connection patterns. This knowledge can be used to only initiate communications that will complete before a connection breaks and so improve data communication. Even if nodes did move randomly, which could be the case for some specific environments [TG05], there would still be some expected link duration information to be leveraged, so that the future of colocation durations could be roughly predicted and exploited. A colocation-aware peer selection framework could use this knowledge to select those providers that will most likely remain connected to the host for the duration of the required service. Also, though an aspect not investigated in this work, the peers in the immediate area during many points in a user's day, will be people the user has some connection with and likely shares interests. This link of social and interest distance is investigated further in [CGD+09].

Creating a functioning and reliable system on top of volatile connections is a huge challenge. Perfect prediction of colocation duration is obviously also quite hard, as humans can have very variable movement behaviour. A person's future absolute position can, to some degree, be predicted, and much work has been performed to achieve this, such as with the Reality Mining project at MIT [Pen07]. They were able to predict the next physical *place of interest* a user would travel to with a possible 95% accuracy using Hidden Markov Models. However, the locations were extremely coarse-grained, such as *work* or *home* and would not necessarily be useful to predict colocation with other devices. Place of interest prediction is not the same as predicting the physical distance between devices, also environment effects can occur that interfere with or block wireless signals, i.e., there is not necessarily a correlation between position and connectivity. Other approaches [GBNQ06, MS02] also reinforce the observation that people do not usually move between location randomly, and their movements follow relatively regular patterns that can be programmatically learnt.

Location information can come from sources as varied as GPS locations to a user's journey plan (possibly gathered from their calendar). The sharing of such personal information represents a risk for user's privacy, it would be unwise for a vulnerable person's device to broadcast details of their journey. Also, although this information may be available on a laptop, PDA or

phone it would not be stored on a less personalised device such as a PMP. Therefore, we assume that no extra information is shared between users to aid in the source selection procedure, only locally available information gathered by the network interface.

Obtaining information about the absolute movement of a device requires either access to a positioning system, such as GPS (as in [SLG00]), or a technique of inference from other sources. Previous work has used the available WiFi access points as beacons to guess at the current location of a device, through both the hardcoding of pre-known WiFi access point positions/Received Signal Strength Indications (RSSI) [HFL$^+$04] and dynamic calibration based on sporadic GPS signals [LHSC05, VPKL08]. These blended approaches can reliably achieve an accuracy of around 30-100 metres. For a device to have GPS information it must not only contain (or be able to communicate with) GPS hardware, but must also have a line of sight to at least three GPS satellites. This can be a problem in urban environments where the *urban canyon* problem, caused by the surrounding buildings, blocks many of the satellite signals [Zha02]. Despite the success of these approaches, they all require additional hardware and are based on the assumption that absolute position is the most important measure of mobility. However, considering our scenario, the most important measurement is *relative* distance, and the ability to communicate. Furthermore, the geographical positioning solutions do not provide a granularity of less than 20 metres, limiting their ability in predicting the existence of smaller scale proximity. Also, our scenario considers mobile devices that want to communicate with other devices likely moving in unison, implying relative position is a more important factor. Therefore, we decided to only focus on network detected colocation as a measure of two devices' likelihood of being able to communicate effectively.

Colocation aware routing algorithms, such as PRoPHET [LDDG09] and CAR [MHM05] already exist; however, less has been done to exploit this knowledge in service provision [TJK04, Har04]. These approaches mostly attempt to overcome unstable connections caused by mobility to hide the nature of the network. CAR uses assumptions about social properties of connectivity to predict (using Kalman filters) whether hosts will meet again in future, in order to forward messages. Our aim is not to transport data to specific hosts (routing), but rather to collect data with appropriate attributes. We agree with the assumption that being in proximity of someone suggests a relationship, and implies properties about any future proximity. This is a useful method, as it does not require any extra information, such as geographic position. Breadcrumbs [NN08] uses information about the changing connectivity to make predictions about future service from wireless infrastructure, to enable devices to choose when it is best to initiate network events. Again showing that using knowledge outside of the immediate

network state can give applications on mobile devices much better behaviour for users.

3.2 Urban Social Networks

Social network theory can be used to analyse the relationships between people, where they are represented as nodes and the links between them as some sort of relationship. Most people spend a majority of their time at either home or work/school, where there are usually well defined network access and administration procedures. We are interested in how people can electronically communicate when they are elsewhere or in the *third space* [Old89]. These other places, such as buses, trains, bars or coffee houses, may not have permanent members but will have some people that regularly frequent them. The interaction between people that are frequently in close proximity represent, in some weak sense, a social relationship. This aspect of urban living was initially investigated by Milgram's important 1977 sociological paper on familiar strangers.

During the *rush hour* in a major city, thousands of people will be commuting across it; each weekday morning (8–10AM) London has over 1 million people travel into its centre [UK 06]. Commutes are usually along familiar routes (e.g., home to office) performed at similar times (e.g. rush hour, or perhaps late at night for shift workers). In London, each weekday there are over 3 million trips on 'the tube', 1.8 million trips on main line rail services and about 5.9 million trips on buses (representing 40% of all UK bus journeys). This shows a vast number of people using public transport; in fact around 27% of all journeys in the UK are made on public transport. The average number of passengers per bus is 14.7 (higher during rush hour), offering a significant number of people to interact with. The average commute time recorded is 43 minutes, though 10% took over 90 minutes, a significant amount of time and long enough to transfer over 475MB (with Bluetooth version 1.1 at effective bit rate). Table 3.2 shows a breakdown of methods of transport used by the public for their commute into London each morning during 2006 (source [UK 06]). These figures represent a huge number of people moving through a small area, with a lot of physical proximity and networking opportunities.

In relation to urban transport systems, people perform many short journeys throughout the week. These journeys may be to different places depending upon the time that they occur. For instance, a journey beginning at 8AM may be the long commute to work, whereas one at midday will be a quick journey to a restaurant for lunch. The colocation durations that arise from different journeys would therefore be time dependent. If a colocation duration prediction is to be made for another user, then not only is that user's identity of relevance, but the time this prediction occurs is of importance as well. For any hosts that are frequently seen by A

Transport Method	People (thousands/morning)
All	1064
Bus	115
Rail	473
London Underground + DLR	342
Car	84
Motorbike	16
Bike	17
Coach	9
Taxi	8

Table 3.1: London Commuter Volume.

at the time of prediction, there will be a historical record of the colocation duration with that host during this period of the day, allowing an informed prediction to be made. A majority of the other peers seen in the system by A will be strangers who are very rarely met, precluding any information being gathered about them. In this case, A's only knowledge comes from the aggregate experience of how all other peers generally behave at this time. If another host B has been seen previously by A, then the history of colocations with B will give a more user specific perspective. If B was seen at many different times previously (e.g., for a short time in the morning, long at night) then the prediction can be further specified by using user specific and temporally relevant history. Therefore, we propose attempting to use the temporally specific history of colocations with the particular user at that time as the best prediction choice. If temporally specific information about that user is absent (e.g., it is the first time A has met B in the evening), just the user specific information is used. If it is an unknown user being considered, then user agnostic information from the present time can be used.

3.3 Peer Connection Process

When a device wants to initiate a transfer it must choose the best source for a download from the currently available neighbours. Moreover, it must decide whether the predicted colocation length of that host will last long enough to perform the transfer. This selection can be aided by historical information to make a more educated guess. The 'best quality' neighbour to initiate a download from is arguably not simply the one that will be colocated for the longest period, the library of files on each other host has a bearing. Potentially a host could connect to every neighbour and download a complete list of all files available, and the selection made

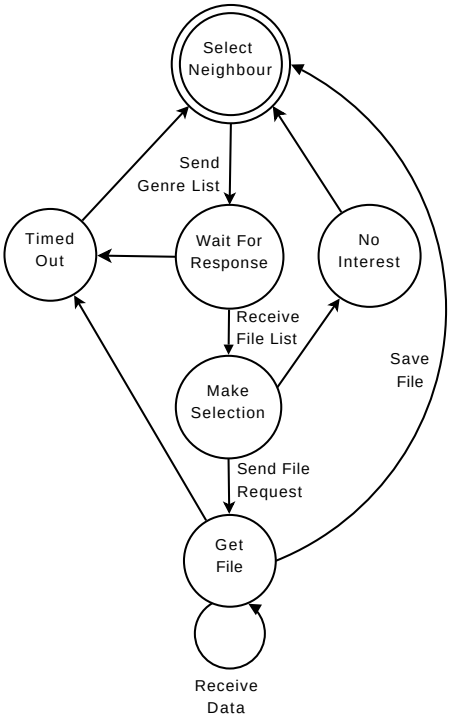

Figure 3.1: Downloader's Finite State Machine.

from that superset. However, this requires a lot of non-local knowledge to reason about all neighbour's libraries and which could be termed 'best'. Bluetooth and its Service Discovery Protocol (SDP), requires a separate connection to each neighbour to communicate data; there is no true broadcast capability. Spending time connecting to every neighbour and exchanging library/interest negotiation information is time consuming. Having a host dedicate potential transfer time communicating with every neighbour, especially if the neighbour set is changing, would create significant overhead. Therefore, we first choose the host that we are most likely to receive the most data from, then only negotiate the file choice with that host. This gives the benefit of less connection creation and file advertisement overhead. Once the source has been chosen, the negotiation for which file will be sent is initiated, followed by any desired transfer. The transfer process as a whole is a receiver initiated interaction. If this process was inverted, and all neighbours were searched for files before the source was chosen, then it would possible to search more effectively. Though this would be at the expense of actually successfully transferring these, potentially, more appropriate files. We will now explain the process and reasoning a host performs when initiating this interaction cycle.

Figure 3.1 shows a Finite State Machine (FSM) of the process that hosts perform when downloading. Upon starting a host will enter the default *Select Neighbour* state, ready to begin a cycle of attempting downloads with neighbours. The host progresses through a series of ne-

gotiation states with a selected neighbouring peer, each state having a timeout that will trigger if the neighbour moves out of range. This timeout will move the host into the *Timed Out* state, causing the process to move back to the initial selection phase, after clearing any temporary data. The negotiation consists of sending a requested *Genre List* followed by receiving a response. An appropriate file is selected from this *File List* and a file request made, a process described further in Chapter 4. If there is no file that the downloader desires, then the process is initiated again. Though this could lead to loops where the downloader is continually selecting hosts and not finding suitable content, it will not cause infinite loops. When a source indicates it has sent all matching files (through an empty File List), the downloader will not select it again for the remainder of their colocation. Two important points to make are that the file is only saved once the download is complete, any partial files are discarded. Any failure during the negotiation places the host in the *Timed Out* state, this means peer selection is reassessed after a timeout, as the neighbourhood may have changed significantly in that time. The selection state does not however cause any radio transmissions, it is a decision made with local information.

3.3.1 Peer Selection Technique

We will now investigate how the *Select Neighbour* step is performed. Device discoveries are performed regularly each time period t, which gives a set of surrounding neighbours N_t. These neighbour snapshots can be combined to form a neighbourhood history. This historical record may then be used to provide indication of the future state of the network. An important aspect to note, is that as Bluetooth device discovery is a probabilistic process [Kha06], a physically proximate device may not be found during a given discovery phase. Therefore, when considering *colocations* the system should only treat another device as having left when it has not been detected for two consecutive discovery phases. The set of devices that are deemed to still be colocated at period t shall be represented as $\hat{N}_t = N_t \cup N_{t-1}$. This means if a host was detected in the current (N_t) or previous (N_{t-1}) discovery, it will be present in the practical neighbour set $\hat{N}_t$. A neighbour is also deemed to be *unavailable* if it is already sourcing a file to another host. It will then not be considered for selection (though its presence is still recorded); this avoids having popular sources uploading many files at once and having none actually complete.

For a given set of colocated devices $\hat{N}_t$, there are many possible techniques for choosing which should be selected, which are investigated in Chapter 6:

- **Random** – A random available host is selected as the source.
- **First Colocation** – The available neighbour whose colocation started earliest is chosen.

- **Last Colocation** – The available neighbour that most recently entered the surround area is selected.
- **Mean Colocation** – The available neighbour with the previous longest average colocation duration is favoured.
- **Temporal Mean** – A temporally varying colocation estimation approach is used, explained below.

It should be noted that the *Random* approach is not truly random behaviour. A host would still only select an available neighbour that also shares some genres of interest. The only random behaviour is in the selection of which feasible neighbour to use.

3.3.2 Slotted Temporal Mean

One technique to predict colocation length is to keep a running mean: for example, each user could log their 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 possible *temporal* variability of movements; for instance, a colocation beginning in the morning (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 slot across a given 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 a *familiar stranger*, that is, someone who has been met often in previous journeys. A host that has been encountered a sufficient number of times, will gain a profile entry in the depicted Table 3.2. If the number of previous samples during a given time slice is not sufficient 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 encountered 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 $Period$ and $Slice$ are domain specific aspects and universal values do not exist that will hold in all situations. For the purposes of this explanation $Period$ and $Slice$ are 24 hours and 4

User	**Time Slice** (hours)						
	Overall	0-4	4-8	8-12	12-16	16-20	20-24
Anonymous	15.9	25.7	10	3.5	5	2.8	18
Bob	12.4	-	20	7	13.8	-	-

Table 3.2: Temporal Mean Table (values in minutes).

hours, respectively. Rather than hardcoding them, these values can be dynamically learnt from a domain's colocation structure, using techniques similar to the ones discussed in [DM07].

The time slice index S to be used when making a prediction at time t is calculated by the simple formula:

$$S = floor(\frac{t \bmod Period}{Slice}) \tag{3.1}$$

For example, with the profile structure given in Table 3.2, a colocation with Bob occurring at time 09:00 hours: S will be 2. Thus, the colocation with Bob will have its duration computed into the 8-12 slice. Also, Bob's $Overall$ mean will include the value in its calculation as well. The $Anonymous$ profile will also have all colocations included in its calculation (for both the $Overall$ and slice 2). So if Bob loses his familiar stranger status at some point in the future, there will still be some impact from him in the profile table.

The manner that the prediction matrix is constructed will now be presented. The initial matrix has a single row dedicated to the *Anonymous* user, i.e. the amalgamation of all users. This row, in a similar format to all subsequent rows, has a field for each time period and an extra *Overall* one that is not time specific. This Anonymous entry will be modified after all colocations with another user. Each field contains the mean of all colocations that previously occurred in that time period. Hence, after a colocation in time period Z, the mean in slot Z will be updated accordingly. The Overall slot will also be updated to include the colocation in its population. Each sighting of a neighbour n is also counted, when this count exceeds a predefined limit[1], the neighbour is promoted to familiar stranger status. This causes a new row to be created in the colocation history matrix, dedicated to the newly tracked user n. Up to this point the only impact n had on the matrix was through the Anonymous profile, subsequently it will affect both the Anonymous and its personal row. The field modification process is exactly the same as for the Anonymous profile, the Overall and time specific field will be updated to reflect a new population mean after any colocations. This whole process will be repeated for

[1] A default of 3 is used in this work, though this value could also be dynamically learnt [DM07]

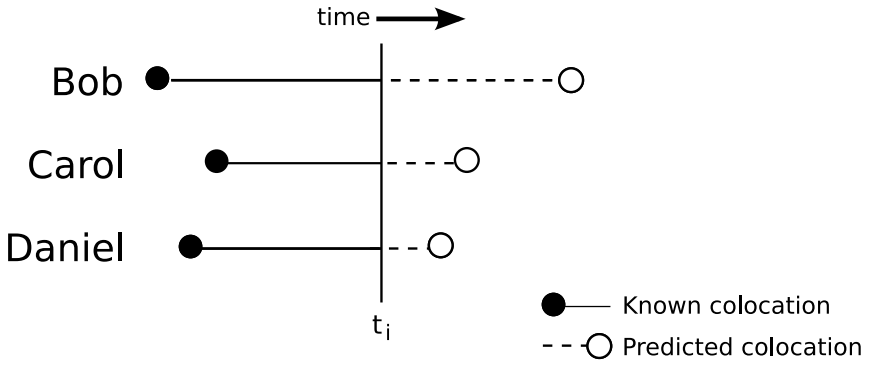

Figure 3.2: Peer Selection Example.

any and all hosts that are met. If hosts were concerned with storage requirements, old host data could be purged with a scheme such as least recently used (LRU).

With reference to the scenario described in Section 2.1, let us assume that $Alice$ has to make a choice between three neighbours at time t (around 09:00 hours), depicted in Figure 3.2. Bob's mean colocation duration with Alice in the time period 08:00-12:00 is 20 minutes, and 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 more minutes, causing Bob to be selected as the connection partner, as he offers the best chances of transferring most media content. If there were another (never before seen) host to choose from, $Daniel$, who became colocated before Carol, then he would represent an even worse choice. As both Carol and Daniel will receive the same predicted colocation length and Daniel has already spent more of his colocation time with Alice, his predicted remaining colocation duration will be shorter.

3.4 Algorithm

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

1. For each of the hosts in A's list $\hat{N}_t$, a prediction is made of the length of the colocation with A, by performing a lookup into the colocation profile table.

2. The predicted remainder of the current colocation time, for each host in $\hat{N}_t$, is computed (this simply requires logging the colocation start point of each host).

3. Hosts are ranked in decreasing order of remaining predicted colocation time, top being the longest predicted.

4. Finally, A connects to the top ranked host $\overline{h}$, to obtain a summary of relevant files available for download, which is a list of files with length $Shortlist$. 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 $duration$, the transfer begins.

A more formal expression of this scheme is shown in Algorithm 3.1. In order to decide whether to start the download (Step 4), A must estimate the time 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, giving the approximate download duration: $duration = size(file)/speed(\hat{N}_t[top])$. Downloads will not be initiated if the expectation of success is too low. More precisely, $duration = (size(file)/speed(\hat{N}_t[top])) * Threshold$ needs to be smaller than the expected remaining colocation. This value allows a device to account for the unpredictable nature of the 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 $duration$. Conversely, values lower than 1 will start downloads even if the prediction technique suggests they will not succeed. A value of 1 would directly measure the efficacy of colocation predictions.

3.4.1 Peer Prediction Overhead

Regular device discoveries will need to be performed concurrently with downloads, in order to populate the neighbour set N. For example, such a component would simply make each device regularly enter the Bluetooth *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.

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. With Bluetooth device addresses (BD_ADDRs) being 48bits and a double precision floating point number being 64bits, each entry can be stored using a small amount of data. From the formula: 112bit + (*slices* $\times$ 64bit), if six time slices are used, then each entry is less than 0.5KB. More basic sighting history must also be kept, to enable the device to recognise when a peer has been seen enough to become a familiar stranger, or not seen for so long that

Parameters: List $\hat{N}_t$ of neighbours in reach and their colocation start time start[h];
Current time slot s;
Matrix M of colocation profiles (i.e., $M[h, s]$ = mean colocation duration with host h for time slice s,
$M[h, \emptyset]$ = overall mean of all h's colocations;
$M[Anon]$ is the profile of the anonymous profile);
List G_X of genres of interest to X;
Returns: host $\overline{h}$ from where to negotiate a download.

{Steps (1,2) - Computation of colocation predictions}
for all $h \in \hat{N}_t$ **do**
  if host h is familiar stranger **then**
    if $M[h, s]$ is populated **then**
      score[h]= $M[h, s]$ - (t_{now} - start[h])
    else
      score[h]= $M[h, \emptyset]$ - (t_{now} - start[h])
  else
    score[h]=$M[Anon, s]$ - (t_{now} - start[h])
{Steps (3,4) - Rank peers and return chosen source}
sortDecreasing($\hat{N}_t$, score)
file = getFileSummary($\hat{N}_t[top]$,G_A)
if score[top] >($(size(file)/speed(\hat{N}_t[top]))$ * $Threshold$) **then**
  $\overline{h} = \hat{N}_t[top]$ {Select host with longest predicted remaining colocation}
else
  $\overline{h}$ = null {Do not transfer, likely to fail}
return $\overline{h}$

Algorithm 3.1: Content Source Selection.

its familiar stranger status should be revoked. An entry for this table would be much smaller, around 100bits. Though these numbers are many orders of magnitude smaller than high quality data items, a large number of them will add up, and space should not be wasted without consideration. If a device is particularly limited in storage, a resource conservative policy can be used to limit the size of these sets (e.g., by forgetting hosts when they have not been seen in a long time). 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. This colocation prediction information can then be shared by any reasonable type of Inter-Process Communication (IPC), such as a local socket or pipe.

3.5 Trust

Open networks, such as public wireless spaces, are inherently open to abuse by selfish and malicious users. *Selfish* users try to maximise their own utility and unfairly gain more resources than is deserved, such as downloading files while never offering any to others. *Malicious* users behave in a manner that purely aims to disrupt the utility of others, even at the expense of their own, such as pretending to initiate transfer of files they do not possess. Some users may not fall into either of these categories, but due to misconfiguration or data corruption still have a detrimental effect on the system. As these networks are open for anyone to join, the detrimental effect of these 'bad' users is often performed by introducing the concept of *trust* or *incentives* into the system.

Incentive based systems work by rewarding hosts with benefits for providing favourable services/interactions to other hosts. In wireless routing, for example, this could be a host reciprocally granting forwarding privileges to another host that forwards its own packets. Micro-payment systems are an incentive based approach where interactions are purchased/sold for small quantities of real money. Non-malicious devices can then spend a small amount to gain services, while being paid for any services they can offer. If a host is as much a benefit to the system as it is a burden, it will expend a negligible net amount, encouraging non-malicious cooperation. Malicious users will have to spend a lot of money to consistently avoid providing services to others. Systems such as these function well if the accounting system can not be gamed; they are however often complex. Mobile Bazaar [CABP05] is a reasonably advanced micro-payment system for wireless collaborative data services. Competition between service providers is created through an economy of very small monetary payments, ensuring peers are reimbursed for any work they do for others (such as forwarding data). The free market construction creates a self organising system of charging for services and collaboration. The requirement of a secure accessible entity to record and charge/recompense all interactions in a disconnected small-scale wireless network is unrealistic. The inclusion of a monetary punishment would not promote the system to users as a method of accumulating music in a low risk manner. Therefore, rather than attempt to employ a monetary based incentive system to avoid poorly performing peers, we shall investigate trust based systems.

A frequently used definition of 'trust' by Gambetta: "*[Trust] (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of his capacity of ever be able to monitor it) and in a context in which it affects [our] own action*" [Gam88]. This view emphasises the subjective and context-dependent nature of trust, two factors that are also

especially pertinent to music appreciation. In our conceptualisation of the system, the terms *untrustworthy* and *malicious* are both used to indicate any data source that is not perceived as 'beneficial' to the receiving host. We thus classify as untrusted, the hosts that provide garbage data (poisoners), push advertisements (spammers) or incorrectly encode/label music tracks (polluters). We do not distinguish between these cases any further, with the rationale that if content supplied by a host is of no use, then the receiver does not care about the reasons behind it, and just wants to avoid similar problems in the future and ensure that the host is not selected as a source any more. Mislabelling of tracks can even be considered a subjective problem: one user may consider *The Beatles* being classified as *R&B*[2] to be an error, while another may not. We do not treat failures due to broken connections as malicious, as they are inherent in the scenario we are considering. Also a selfish host cutting off the connection half-way through a download would save more energy by initially refusing to serve the file.

Trust is used in many real systems to enhance their utility to their users, particularly Internet systems. Similar to user-based recommender systems, trust is often used to aid navigation of large amounts of information. The website *Epinions* is an online opinion source, it uses explicit trust ratings to qualify how much faith should be placed on another user's opinion; its behaviour has been analysed [Guh03]. Advogato is a website for rating the ability of open-source programmers, which was created with the intention of studying how trust behaves in large online communities[3]. A trust-aware client for the Internet based peer-to-peer network Gnutella has also been developed [CDV+02]. This work demonstrates the interest in ensuring that content acquired from peer-to-peer systems is what it claims to be, and that peers behave appropriately.

Wireless networks have greater accessibility compared to wired networks, due to physical proximity giving the ability to connect, rather than needing physical access to someone's network hardware. The ease that devices can join them is one of their major strengths, however it is also the main security weakness. For example, any device within range may unfairly monopolise the medium or even jam the spectrum to prevent usage by anyone else. Due to continuous changes in the network topology as hosts move around, similar abuses could be regularly perpetrated fairly easily without ever being caught and stopped. Thankfully, malicious devices present a much smaller overall risk in short range wireless networks compared to wired networks due to the lack of global addressing and connectivity: a single bad host is not able to

[2]Rhythm and Blues as a term, has evolved over time from denoting gospel, blues and disco to the modern meaning of soul/funk influenced pop music.

[3]Deployed trust websites: `http://epinions.com` and `http://advogato.com`

directly attack everyone in the network. Therefore, compromised devices only have a locally limited effect, preventing developments similar to *botnets* on the Internet. Though there may be the problem of transitive bad behaviour, e.g., viruses or mislabelled content getting passed through the network by non-malicious hosts without their knowledge.

There are many types of node behaviour that it would be advantageous (though not necessarily possible) for a content dissemination system to automatically restrict. The most important one is when a peer distributes data other than what was requested, such as sending an audio advert when a rock track was requested. This would be immediately obvious to a human user when this has happened, yet very hard to automatically determine, especially if the advert was embedded in a normal music track. Alternatively, a non-matching file may be offered due to a user's library being poorly managed and labelled: this might not strictly speaking be considered malicious behaviour, but is definitely problematic for other users. A selfish node may never offer a file list when asked, in order to avoid using its battery power and capacity to aid other users. This type of behaviour is hard to detect because a user with little or no files would behave in the same manner.

3.5.1 Computational Trust

When selecting a peer to download from, we would like to quantify the belief that the host will not behave in any of the previously mentioned 'negative' ways. This belief can be represented as a *trust value* in the range $[-1, +1]$; with -1 representing total distrust, $+1$ being absolute trust, and 0 indicating a neutral opinion. A default bootstrapping value of 0 could be used to represent the trust level in an unknown host. *Trust management* systems have been proposed that enable reasoning about the trust value of other peers by means of their historical behaviour. If past behaviour functions as an indicator of a node's future performance, which is a reasonable assumption, previously malicious peers can be avoided and the good hosts preferred by recording a host's actions. A host that has behaved inappropriately in the past may be ignored by others, segregating it from the system. This exclusion not only serves to limit the damage a bad user can have on the system, but is also an incentive to behave for all peers.

It has been proposed that trust should not be transferred across different contexts, for example, a host having a bad trust value in providing service X, should not have that reflected in its service Z valuation [SS05]. Hence a trust value is service dependent, one global value can not be used for other services, unlike the colocation prediction mechanism, which is service independent. Attempting to apply transitive trust between services facilitates the use of more available information, however it would introduce a lot more complexity and possible errors into the system [CH97]. To investigate this issue in detail, a multi-service scenario would have

to be considered and evaluated. The issue of how much (dis)trust should be transferred to other services offered by a host, depends on how much the quality of a host's service X implies about its service Z. Host B could provide extremely beneficial and relevant music from host A's perspective, but due to differing political views of the owner of device B, have very poor news quality. The use of our wide definition of trust make the transference of trust from one domain to another an especially unwise choice. Previous research has considered how to use history from other contexts to bootstrap trust valuations [QHC07].

Ubiquitous systems on personal devices can be characterised by seasonal movement patterns of groups of devices. We thus argue that a trust-aware content source selection protocol could be effective in estimating a peer's future behaviour, isolating malicious peers and thus further reducing the chances of service provision failures. The most important hosts to have trust data about, are the ones that are met frequently, these are also the hosts that we can build up the best estimation of their merit. Obviously infrequently met hosts are the majority, but in many environments there are a small number of frequently met hosts: familiar strangers, friends, colleagues, neighbours, etc. These hosts are the same class of peers that can have their colocation durations predicted (Chapter 3). If a malicious host is seen frequently, it becomes even more important to recognise its bad behaviour and take steps to avoid its impact.

The remainder of this chapter gives an introduction to the field of trust in peer-to-peer and mobile systems. The importance of node identity in trust systems is discussed, as well as the methods to store and modify the trust valuations of peers. The way our system utilises the trust valuations when performing a source selection is subsequently detailed. It should be stressed that we are not attempting to develop new mechanisms for storing, updating or managing trust. We simply aim to employ some previous work on trust to the problem of item distribution on urban transport to maximise users' utilities. With the scenario's particular features of high churn and users' lack of interest in the total population of data items.

3.5.2 Trust Systems

Fundamentally, a trust system gives a mapping from a host's *identity* to a *trust valuation*. Establishing a device's owner in our scenario is not particularly important as the people are unknown anyway, unlike when sending a sensitive message to a particular person. However, it is important to know when we are interacting with a device that we have dealt with before; i.e., a host that claims to be host B is indeed host B. Large-scale distributed systems (especially disconnected ones) have problems achieving 100% verifiable node identification. Malicious nodes can possibly maintain multiple identities, to avoid negative associations with an identity caused by their actions, called a *Sybil attack* [Dou02]. This can be avoided by enforcing a single,

unique identity, which is an extremely hard task, usually requiring centralised assignment to avoid clashes and possibly secure computing hardware to avoid device tampering. For the purposes of distributing content for entertainment purposes, specialist hardware, such as a Trusted Platform Module [Tru05] is a relatively onerous requirement.

Cryptographic proofs of identity are a very useful technique, though they present a barrier to entry for resource poor devices. Using public key cryptography, hosts can create public/private key combinations allowing them to sign messages, proving they are from the user who holds that key. Without a trusted third party, such as a hierarchical certificate authority or web of trust system, the public key can not easily be linked to a person; though as we have stated that is not an aim in our system. The greatest flaw in an approach such as this, is that many public/private keys can be generated, but this requires certain amount of processing to be expended. Hence, this approach would give the system the property of *non-repudiation*, it will not enable the prevention of Sybil attacks.

Preset, immutable identities, such as Media Access Control (MAC) addresses or Bluetooth Device Addresses (BD_ADDR) are defined for many network interfaces. These require a centrally administered system of address distribution to avoid clashes, an aspect often handled by the hardware manufacturers. Using the BD_ADDRs as the identity does not require any additional identity infrastructure or overhead for peers in the system. Though it would be possible to spoof the BD_ADDRs with custom hardware or highly functional devices, our main aim is not to harden the system against all dedicated malicious attacks, but to mitigate attackers' effects and to segregate poorly performing devices with corrupted libraries. Therefore, we use a device's BD_ADDR as the means of identification in our system.

3.5.3 Trust Management

The architecture of a trust system defines many facets of its operation, such as where trust values are stored and how they are accessed by hosts. Having access to the trust values is crucial for hosts to make an informed decision. It would be advantageous to have all interaction histories and trust information stored centrally, allowing all previous interactions to influence a host's trust valuation. Interaction histories would be processed, to allow identities to map to a rating, which would all happen at a single third party that all other hosts have regular access to, so they can receive all trust information. This approach can easily be used with Internet based systems such as the online auction website eBay[4], where every system interaction is performed through the central trusted entity. This would not be possible in our scenario as there is no

[4]eBay's Website: `http://www.ebay.com`

frequently accessible third party for all peers. The centralisation would also give a single point for failure/monitoring. Alternatively, a distributed system can be used, where hosts do not rely on a single trust repository. Considering the arguments from Section 2.3.2, it can again be seen that a structured distributed system of arranging trust information would not be appropriate for our scenario. Furthermore, the challenges presented by possibly malicious hosts would make this an even worse choice, as subverting the trust system itself would be a very powerful attack.

Unstructured distributed approaches for storing trust valuations can gather their information in two ways:

- **Direct experience**: Each host maintains its own view of the trust ratings for other hosts. All information used to create a trust valuation is only gathered from interactions that the host participates in. Therefore, all information used to create the valuation is inherently trusted. This approach lends itself to creating personalised subjective measures based upon behaviour observed by the local device. This avoids any control overhead from the trust system, as each host only uses local information.

- **Direct experience and recommendations**: As well as using interactions that the host has directly participated in, trust values are also influenced by *gossipping* recommendations between peers. These recommendations allow peers to share their trust valuations with other peers in the network, thereby gaining non-local information. This approach is particularly suited to closed networks of a limited size, so that trust values can more easily converge to globally accepted values.

To avoid dependencies on other nodes for providing trust information, direct experience trust systems have gained momentum, such as in [DDB04], where each host maintains its own opinion of other hosts using local information, thus adhering to the human intuition of trust as a highly subjective metric. A potential disadvantage of this local trust management is bootstrapping: whenever a node moves into a new environment where it is unknown, it will take some time to build up a good standing with neighbours, and to form accurate judgements about others. Based on our observation that hosts in pervasive computing settings often meet a subset of the total peers, we can argue that, after a relatively small initial bootstrapping time, the trust values about that subset will develop into accurate estimators of the providers' true utility.

Recommendations

Rather than only relying upon personal experience, it is possible for local repositories to gain information from indirect interactions by the use of sharing recommendations, first proposed

by Abdul-Rahman and Hailes [ARH00]. When meeting another host, instead of just sharing data, peers can also share information about the outcomes of their past interactions, spreading information about who else has performed well/badly. This process allows hosts that may not have directly received a file from a peer, to still collect some information about its past performance. In a sparse network such as ours, where files are only collected from a small subset of the total population, it will be rare to have direct historical information about a random neighbour R. The ability to ask other neighbours if they have interacted with R previously allows more informed decisions to be made. Distributed trust management systems such as this are useful in networks without access to a centralised repository. EigenTrust [KSGM03] is an example of a global view trust methodology for peer-to-peer systems. Through the sharing of its recommendations, all peers in the system approach the same global view about other peers in the system. In order to work effectively, simultaneous connectivity to a sufficient number of non-malicious peers is required, which may not be guaranteed in pervasive computing situations.

The opinion another host has about the quality of a service could be considered a subjective measurement. If we are accepting opinions from others, we may not be sure they are forming their opinions in a consistent manner to ourselves. The main problem with allowing recommendations it that as well as allowing malicious hosts to be gossipped about, it also allows malicious users to spread their own gossip, thereby allowing malicious gossip to be used as another tool to subvert the system. If a group of malicious users want to destroy the reputation of host R, then they could all spread false reports of bad behaviour, causing R to be treated as a malicious peer. It has been argued that using untrusted information to influence our views on trust is inherently flawed, and information should only be gained from explicit evidence [Obr04].

Some approaches separate the trust they have in another host into trust in delivering a service and trust in giving recommendations [LI04a]. It also makes sense to track how much of a trust valuation comes from direct experience and how much from recommendations, possibly even which host's recommendations were used. This will then allow a host's effect on the trust repository to be revoked if it becomes apparent that it is a malicious entity. Although recommendations can aid the gathering of information in new environments, when all neighbours are unknown; our scenario posits that not only are familiar strangers often present, but the high-churn strangers are rarely seen again, minimising the benefit of gossipping about their behaviour. The frequently seen peers can then simply be learnt about through direct experience, saving the overhead of recommendations and not providing another mechanism to subvert the system (through false recommendations).

Though they can aid the gathering of information in new environments, the proposed reg-

ularity of user movement should allow devices to directly collect information about common neighbours.

Detecting Failures

A precondition of knowing which hosts are malicious/selfish is the ability to recognise what actions are bad. For our automated system, it would be desirable to have automated judgements made about other host's actions. If a shared file is corrupt or incorrectly encoded, a user's device can detect this and mark the transfer as bad. Deciding The decision of whether or not a correctly encoded audio file matches the requested type of content can not be programmatically made, as it requires human feedback. This feedback can only be provided after the user listens to the track, via a feature of the user interface. This means that after a file has been downloaded, it should only be made available for sharing to others once it has been listened to. This is to avoid a good user passing on bad files before they have had a chance to classify the file.

3.5.4 Updating Trust Values

The importance of trust in a wide variety of computer systems has been identified by previous research, particularly in the realms of the Internet and the web [GS00, Mas06, AG07]. The dynamic update of an identity's trust value is the most distinctive aspect of a trust system. Many approaches have been used, including Bayes' theorem [QHC06]. Once a host has been deemed to have behaved badly, its trust level should be modified to reflect this behaviour. A common approach in trust systems, due to its easy computability, is the beta distribution [LI07]. It is a probability distribution, whose probability (p) is parameterised by two positive parameters: α and β. These variables can represent the count of times a given host has performed well (α) and badly (β). Therefore, a trust value can be computerised for a given host, which has α successes and β failures, using the following formula (note both α and β are initially 1):

$$E(p) = \frac{\alpha}{\alpha + \beta} \tag{3.2}$$

The beta distribution is a continuous probability in range [0,1], which can easily be scaled to the standard trust range of [-1,$+1$]:

$$BetaTrust = (2 \times E(p)) - 1 \tag{3.3}$$

The computation of this trust metric is not complex and only requires the storage of two integers. A visual representation of three different beta distributions is shown in Figure 3.3. The left-most curve demonstrates a particularly untrustworthy host (having performed 20 failures and 4 successes) with $-0.6\ldots$ for the trust value. The central curve has less trustworthy value of 0,

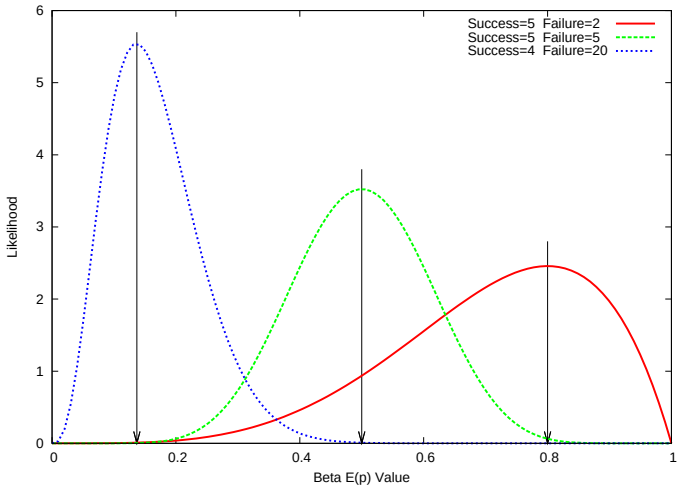

Figure 3.3: Beta Distributions

and the right-most curve a comparatively unsure, yet high, trust value of 0.43. The probability density function of the beta distribution can be calculated using the formula:

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}\,du} \tag{3.4}$$

This equation gives a measure of the confidence in $E(p)$, i.e., the y value of the peak of curves shown in Figure 3.3. Therefore, a host can gain an estimate of how reliable its trust valuation of another is. Though not analysed in this work, the estimation confidence would potentially be useful in a deployed system. If a user was manually browsing hosts to find more files, the user interface of the content distribution program could also present not just the trust value of a neighbouring host, but the confidence in the valuation as well, thereby informing the user about the amount of information used to come to a trust rating. It could also be used to differentiate between two hosts that have the same colocation prediction and absolute trust ratings.

3.5.5 Selecting with Trust

Once a host is deemed to have performed sufficiently badly, it should no longer be communicated with to avoid its negative effects. For our algorithm (Section 3.1), this is performed by removing the set of malicious hosts M from the parameter list: $\hat{N}_t = \hat{N}_t \setminus M$. This will exclude the host from the system, providing an incentive to behave well and limiting the damage bad hosts can have on the system. If a host achieves a malicious classification, then it will be avoided in all future interactions. Similar to the colocation records, this information could be purged after a long timeout to avoid the excess accumulation of state. Hosts employ a device defined threshold value $TrustThresh$ to the trust value of neighbours, when it falls below the threshold they will be ignored in future source selections. A node that has been deemed mali-

Peer	Genres	Predicted Colocation (min)	Successes	Failures	Malicious
Bob	rock, metal	12	32	2	No
Carol	dance, electronica	5	0	0	No
Mallory	rock, pop	15	1	4	Yes

Table 3.3: Alice's Trusted Peer Selection.

cious will never be given a chance to redeem itself, leaving it permanently sanctioned. In our scenario, this will not significantly limit the amount of possible sources, as there are so many other nodes. Optionally, the negative value could slowly increase back to the threshold value, allowing it to participate again; this feature is not being pursed for investigation in our work.

Setting the default trust value of a unknown identity to have a low value, means it would rarely be of benefit for a malicious host to change from the current identity to a new unknown one. This has the problem of unknown (yet possibly good) hosts not being selected, making it hard for an unknown host to be given the opportunity to improve its trust valuation. In a large, sparse and dynamic network, many encountered hosts will be unknown, so only particularly cautious users would want to use an especially low default trust value.

With reference to our target scenario (Section 2.1), suppose Alice has a highly positive trust value for Bob, as a consequence of a successful history of interactions; while she has a neutral trust valuation for Carol, as there have been no previous interactions between them. Alice has a particularly low trust value for Mallory, as a consequence of a history of poor interactions, although he is often colocated for long periods. The information Alice has to make a judgement on whom to select as a download source is shown in Table 3.3. It can be seen that if selection was purely based upon longest remaining prediction colocation, Mallory would be chosen as a source (15 being the largest prediction). However, as he is deemed 'malicious', she will exclude Mallory from the shortlist of hosts with matching interests, and only be left with Bob and Carol. Bob, being the highest rated non-malicious neighbour, is thus selected as the download source. This trust valuation will neither improve nor decrease the score resulting from colocation analysis, it simply acts to allow segregation of poorly behaved/malicious nodes. Hence it serves to act a filter for untrusted nodes, rather than giving preference to trusted nodes.

Described formally, the additional step of trust reasoning serves to add a simple pre-processing step to the peer selection algorithm (Section 3.4), to remove all malicious nodes from the neighbour set $\hat{N}_t$.

for all $h \in \hat{N}_t$ **do**
  $Rating = ((2\alpha)/(\alpha + \beta)) - 1$
  if $Rating < TrustThresh$ **then**
    $\hat{N}_t = \hat{N}_t \setminus \{h\}$

Algorithm 3.2: Malicious Neighbour Pruning.

3.5.6 Summary

This chapter presented some background research on the topic of predicting network properties in opportunistic networks, then proposed an approach for the selection of a stable neighbouring peer. The actions a device will have to perform and the data it will have to store to achieve this process was described. In short, to predict the length of colocation a neighbour will have, it should attempt to use the mean of previous colocations with that neighbour that occurred at a similar time. If that host has never been seen at a similar time, then that user's overall mean colocation duration should be used. If a neighbour has not been encountered frequently or a similar time before, the mean of colocations with all other devices that occurred at a similar time should be used. This requires keeping a small amount of state about each host that is to be remembered and used for more specific predictions.

The chapter also introduced the concept of trust in a content dissemination network, defining what we mean by trust, where it is stored and how it is updated. The manner that trust is used in the download source selection process is also defined. Principally, if an automatic and anonymous data sharing system is desired then strong trust can not feasibly be achieved. However, considering the challenging nature of the scenario, the cooperative and successful behaviour of others should not just be assumed. Therefore, employing a trust system that can at least prevent egregious long-term misbehaving neighbours, is of value, particularly if it does not incur significant extra overhead on the system.

4

Content Selection

The previous chapter explored how important the careful selection of a download's source is to achieve complete transfers. The initial peer selection is based on colocation prediction of the source devices; if the assumption that all others hosts have an item of interest to download is dropped, the downloading process requires further negotiation. This chapter discusses the selection of which file to download from the selected source, including the negotiation process and overhead. File choice is important for the receiving user and, due to future interactions, all other hosts in the system. A host's choice of file to download impacts the behaviour and utility of the system as a whole, because it can affect all the future interactions for the downloader. More specifically, when a file is shared with someone else it affects the file's availability/permanence in the network, and thus how easily that file can be found by users, or indeed lost from the system (upon deletion or data loss). On a personal level, if the system is not gathering files of interest to users, they will not be inclined to run the software, due to resource usage on their device.

This chapter gives background to the problem of deciding which item of content a user desires, how and when file negotiation should be performed by our system, and most importantly, which policies can be used by this process.

4.1 Automatic Selection

Conceptually, every person has a taste in a subset of all music; an ideal automatic music distribution system would not only collect music that they already like, but would attempt to collect some that they would like if they heard it and some that they could develop an interest in. It would anticipate gradual changes in their tastes, and shifts occurring in the music world. Even in powerful centralised systems with millions of users, this can be challenging and is a very active area of research, often framed as *Recommender Systems*. These mechanisms [SKR99] are made practical use of in many modern e-commerce websites, such as Amazon, eBay and Netflix[1].

Recommender systems perform a type of filtering; they are conceived as a mechanism for avoiding the information overload that is present in modern digital systems, such as the Internet. They aim to take a large set of items of interest and present a relevant and manageable subset to users, allowing easier finding and consumption. Often, collaborative filtering is used, where items are suggested to users based upon which items are favoured by similar users, known as *homophily* [MLC01]. This makes the assumption that if another person has similar tastes, then it would be wise to assume items that they enjoy represent good suggestions. A common criticism of recommender systems, is that they result the homogenisation of users, assuming that people want to become even more similar to related users. The ability to perform such user comparisons encourages the use of a powerful centralised architecture, to allow the aggregation of as many user's information as possible.

Decentralised systems suffer from the absence of easily available global knowledge and thus the inability to capitalise upon the potentially vast amounts of information in the system as a whole. This reduces the ability of processes such as collaborative filtering to function effectively. More specifically, short range wireless systems will only be able to make use of data from the local host and its neighbours (possibly including some historical behaviour) and they do not have large amounts of processing power to utilise. Previous work, such as mobHinter [SPGR08] suggests that it is possible to approach the efficacy of centralised system, though it takes time for information to fully propagate through the system. Also, mobHinter assumes a closed system of less than a thousand users, which facilitates complete propagation more easily than a system of millions. A real large scale system would likely have constant churn, possibly making the eventual complete collection of global knowledge about other users impossible.

An alternative approach to selecting subsets of items is to focus on aspects of the content of the items rather than the similarity between users, commonly termed *content filtering*. In

[1] Website URLs: `http://amazon.com`, `http://ebay.com` and `http://netflix.com`

fact some have suggested that focusing on items is not only inherently more scalable, but can be more accurate [SKKR01]. Technology exists to perform similarity analysis of music tracks, considering the volume, tempo, syncopation and frequency analysis, etc [SLC07, Swa02]. This can successfully identify similar tracks, but takes a lot of processing and only focuses on low level constituent parts of the track. Hence, though this could be performed independently by pairs of users, to see which tracks seem the most suitable to send to each other, it would not be computationally feasible.

Defining a user's potential interests is a very nebulous and subtle challenge, which we do not attempt to completely solve. To use a high level and yet flexible manner to discern which tracks are appropriate for a user, our approach uses the natural categorisation of *genres*. In fact, we use music genre as the most important feature of a track, more important even than the artist or album. The justification for this is that we are only trying to procure more music that a user may be interested in, so as to create a cache of interesting music for them to explore. The aim is not to complete a list of artists or albums that a user wants, they can collect specific files themselves. The only behaviour we aim to achieve is to collect more files of the categories a user already has in their library. That way, the library will be filled with files that are similar to ones they have previously obtained, and hence ones they have previously shown interest in. Meaning, if a user has *rock* and *pop* in their library, all the system should attempt to do, is get more rock and pop.

Collecting more of the same genre also allows for more file matches with neighbours than just artist or album alone. Another problem lies in collecting more of the same artist, where a user may have their favourite album on their music player and yet not have loaded others that they possess. If only files by matching artists are collected, there may be a lot of repetition in the procured files.

This genre of a track is directly available in the meta-tags of digital media (such as ID3 for MP3) and is a concept that users already use to define what their tastes are. To configure the device for their tastes, a user must therefore place a selection of files that they enjoy in the device's library. These files could be obtained from online music stores or digitally encoded from their existing music collection.

In a fully deployed system we would expect the user interface to have the ability to black-list particular bands/albums. Conversely, it would be feasible to employ a whitelist system, whereby particular desired artists/albums are preferred over others of the same genre. The actual selection of the file to download is performed by the receiver, leaving the final decision in their hands. These additional behaviours are not investigated in this work due the challenges of

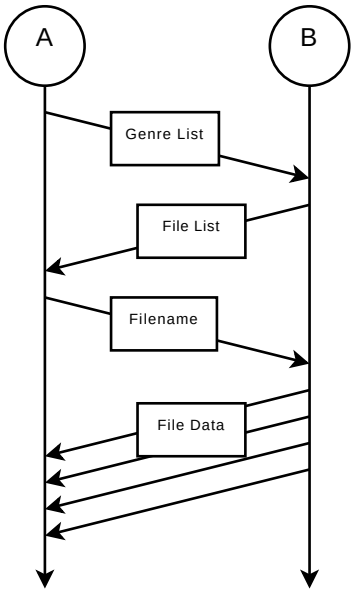

Figure 4.1: Advertisement Protocol Diagram.

statistically modelling a user's particular (dis)like of specific artists.

4.2 Advertisement Process

When a host decides to download a file from the source peer, the negotiation of the file selection (depicted in Figure 4.1) will take up useful downloading time; we should therefore spend as little time as possible performing it. However, an attempt should be made to choose an appropriate file for the user's taste, a short but incremental learning (across each download phase) of the source library would be preferable.

After a downloader, A, connects to the source, B, it sends a string list of genres it is interested in receiving, allowing B to search its library for files of matching genres. Rather than have B send its complete library listing (which could run to thousands) and allow the downloader to choose its file, a limit should be placed upon the size of this list $ShortlistSize$. The source then sends a shortlist of genre matching files; A must then select a file from this list. Firstly, it purges any files that it already possesses from this list, to gain a shortlist of acceptable files. However this pre-purged list still provides some information, it gives A a list of files that need not be advertised back to B in future, as B already owns them. Therefore, each host should maintain a list for every other host, of which files need not be advertised, containing files that either A or B has advertised in previous negotiation processes. This process is shown in Figure 4.1; it should be noted that if no suitable files are found by A, the source

selection process is begun anew. Maintaining a list of all non-advertisable files for every other host will lead to very large storage requirements. Therefore, these lists are only maintained for current neighbours, i.e., when two hosts disconnect from each other their non-advertise lists are deleted. This will greatly reduce the amount of state being kept at the cost of some needless advertisements when/if the hosts meet again. For devices with plentiful storage, all this information could be kept between sessions, and non-advertise lists only purged when a host has not been seen for an extremely long time.

for $file \in library$ **do**
  if $file.genre \in h.interest$ **then**
    if $file \notin advertised$ **then**
      $shortlist = shortlist \cup file$
if $shortlist = \{\emptyset\}$ **then**
  sort $shortlist$ by $policy$
  $advert$ = $shortlist[0 : Shortlist]$
  reply with $advert$ to start transfer

Algorithm 4.1: File advertisement algorithm.

The files in a single advertisement may not be the only ones A is aware that B possesses, track names from previous negotiation phases (that were not downloaded or were already owned) are also known to A. From this, possibly large, track list, any one can be selected and unless extra meta-information is included (increasing the shortlist data size) it could be hard for the downloader to decide which is best. The source however does have a lot more of the meta-information about the file, such as how many times it has been downloaded from B, how many other people have advertised this file to B. Therefore we allow the source to explicitly sort the shortlist and have the downloader select files in order that they are advertised. This allows B to indicate which files it believes should be downloaded first, based upon its knowledge of the meta-data (though the downloader still has the freedom to select whichever file it would actually prefer).

4.3 Advertisement Policy

During each file selection phase, the source will send a list of potential files to the downloader, of length $ShortlistSize$. Subsequent file selection phases will contain additional advertisement lists; however the same file should not be re-advertised by one host to another. The downloader will cache previous advertisements in a list N, thus not requiring any re-advertisements. These caches will gradually grow in size over time, possibly taking up significant storage space. To limit the overhead of keeping these caches, they are purged when two hosts disconnect from

each other. Though this represents a loss of gathered information, it is preferable to keeping filelist caches for all hosts that have ever been seen.

The order that a host B will advertise its library to A is decided by the *advertisement policy*. The set of possible files P for inclusion in the advertisements is a subset of the host's library L, where each file matches one of the downloader's (A) genres of interest (G_A) and has not been previously advertised (list N):

$$P \subseteq L, \forall p \in P[genre(p) \in G_A \wedge p \notin N] \tag{4.1}$$

This set P is then sorted according to the advertising policy employed by the source. Possible advertising policies B could use include:

- **Random** – The set P is arranged in a random order.
- **Common** – The set P is arranged in descending order of the amount of times B has seen the file advertised by others.
- **Uncommon** – The reverse order of Common.
- **Popular** – The set P is arranged in descending order of the amount of times a file has been downloaded from B.
- **Unpopular** – The reverse order of Popular.

The advertising policy that the source uses can have significant effect on the behaviour of the system as a whole. *Uncommon* attempts to ensure that rare files are replicated, ensuring sufficient availability in the system for obscure content. *Popular* will focus on making sure that files that are desired by many hosts in the system exist in many places, so the majority of people will always have new files to procure.

4.3.1 Overhead

If either the Uncommon or Common approach are to be used, then the host must record how many times it has seen each file in its library appear in an advertisement. This will require an additional integer number be stored in the meta-data of each library file. To use a popularity metric, there must be a meta-data field stored in library file recording how many times the host has transferred that file to another peer, again another integer per file. Both these pieces of meta-data will result in a minor increase in storage size for each piece of content and a small amount of additional processing upon each advertisement/download, respectively. Rather than parsing and sorting large libraries for each file negotiation, the files can be pre-sorted and indexed for any relevant attribute to avoid any significant processing overhead per negotiation.

4.4 Summary

This chapter covered the methods used to advertise and select a file from a given source. Advertising policies for prioritising library shortlists were defined together with the required overhead of providing this behaviour. The process of negotiating a file to download was defined together with the information that should be stored to avoid repetition of advertisements and facilitate the learning of a given neighbour's library.

5

Datasets

This chapter describes the datasets that were used as input to the simulation experiments we used in the evaluation of this work (Chapter 6). First we describe the processing and analysis of wireless user traces, including some gathered specially for this research (Section 5.1). The second covers how a popular social music website was *crawled* and its data recorded (Section 5.2). This information, about the site's users, their friends, and musical listening habits was then used to form a model of peoples' music tastes. This data was gathered to better understand the sphere where our proposed system would be deployed and the data it would be used to share. The music library model and its use in the simulations is also described.

5.1 Colocation Traces

Due to the difficulty, cost and workload of deploying large scale mobile computing systems, simulations are often used to gather preliminary requirements and perform analysis of a proposed system before, possibly wasted, energy is spent. If wireless mobile hosts are to be simulated, their inter-connections that form the dynamic network topology need to be modelled. This data can either be gathered from real devices [NDGG07] or be created using synthetic *mobility models*. 'Real' data can be gathered when a testbed is experimentally mapped, the

monitored network state can then be virtually 'replayed' later and comparisons of device behaviour investigated.

Mobility models use a formalised definition of device movement, which can be particularly useful for analysis, allowing controlled variation of device movement and analysis of the resultant behaviour. They can be very simplistic representations of how people move, such as *Random Waypoint*, where each node chooses a random destination point in a virtual environment and a random movement speed. The host then moves to its destination in a straight line at the chosen speed; this process is repeated in sequential legs for the duration of an experiment. Alternatively, rather than assuming completely random movement, extra constraints can be placed on the nodes, such as only being able to move along certain routes. This approach is exemplified in the *Manhattan* mobility model, where nodes traverse a grid topology of parallel and orthogonal interlinking *roads*, reflecting American cities. At each intersection a node carries straight on or turns with certain probabilities, as an analogy to vehicles moving through an urban environment. Each of these approaches treat people as independent agents following a simple set of static rules. Further detail can be added by including group movement of hosts [HGPC99]. This behaviour decreases the amount of changes in connectivity compared to random movement, however it can also increase the amount of contention if hosts are moving in dense groups. When considering humans, added realism can be achieved by considering their social ties and how this affects people's movement. An early example of this approach is demonstrated in the *Community* mobility model [MM06]. It is a model founded on social network theory, allowing connections of hosts to be grouped together in a way that is based on social relationships between the individuals. This grouping is then mapped to a topographical space, with movements influenced by the strength of social ties, that may also change with time. The properties of the synthetically generated traces have been validated against real traces from the Haggle project. Our observation here is that people with strong social links are likely to be geographically colocated often or from time to time.

The generated nature of mobility models allows the prescription of particular features of the resultant dataset, e.g., number of hosts, connectivity or movement speed. Unfortunately, they are based on an idealised concept, and even if certain parameters are the same as real traces (e.g., mean inter-contact time or clustering coefficients) they are arguably not necessarily a reflection of how real devices behave.

In general, a real wireless network's state can be defined by either considering the connections that are detected, or the geographic position of nodes combined with a signal propagation model; from the position and range/connectivity of connections the network state can be de-

rived. Colocation has the advantage that it implicitly takes in to account signal propagation and environmental factors, though it restricts analysis to a particular radio/network type. This section now documents some collected network traces that are used in our investigations and network simulations.

Some basic analysis of the traces provides some insight into the differing properties of the environments they were collected in. Importantly for Bluetooth traces, a discovery inquiry will not always detect every other physically proximate device due to the FHSS, as a device may search a disjoint set of the frequencies that another device is broadcasting on. This will cause some colocations to not be detected in scans where there is physical proximity, adding some false negatives into the colocation traces. This does not affect established data transfers, as it is only the discovery process where this happens. Other devices are probabilistically detected depending on the Bluetooth version used [Kha06]:

- **Version 1.1** – 98.95% of devices discovered in 5.12s;
- **Version 1.2** – 99.99% of devices discovered in 3.83s;
- **Version 1.2 with Interlaced Inquiry** – 99.99% of device discovered in 1.28s.

5.1.1 Trace Details

The Trace sets that are used for evaluation of this work and are described in this section:

- **London Underground** – Journey traces collected over one month containing two separate lines at the end of 2007. Millions of journeys are made by hundreds of unique passengers.
- **Unitrans** – Bluetooth traces collected on 33 buses of the Unitrans public bus system at the University of California, Davis, USA in early 2006 [Jas06].
- **Reality Mining** – Bluetooth traces collected from devices deployed to 100 users at the Massachusetts Institute for Technology (MIT) over the course of the 2004–2005 academic year. Constituting 350,000 hours of continuous data of human movement [EP05].
- **Haggle** – Bluetooth traces collected during January 2005 at the Computer Laboratory, University of Cambridge, UK [SGC+06].

London Underground

Despite the availability of some wireless traces of people using public transport, mainly from the Crawdad repository at Dartmouth [Dar06], we wanted to perform large scale simulation of a city and its inhabitants. The movement traces in this set were collected anonymously from the London Underground rail system over the course of one month. The 'tube' is an underground metropolitan mass transit system serving most of Greater London. It carries 1.07 million passengers per weekday morning rush-hour (8AM-10AM) on average, totalling 3 million per weekday [UK 05]. Over 22% of all Tube journeys are paid using *Oyster* cards. The use of Oyster Radio Frequency Identification (RFID) cards for electronic payment has been introduced over the last few years as in many cities' transport networks, including Hong Kong's *Octopus*, Japan's *Suica* and Washington DC's *SmarTrip*. RFIDs 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; crucially for our work, it enables the monitoring of passenger's movement in the system. We obtained the data from *Transport for London* (TfL), the local government body in charge of the London Underground. The rail lines run through the centre of the London metropolitan area and are used by commuters, shoppers and tourists.

The data used in our analysis is from two different underground railway lines:

- **Victoria line** – which connects the south-west of London to the north-east. It is 13.25km long, carrying around 511,714 passengers per weekday [UK 05]. We have data from 10 stations, over 1.05 million journeys.
- **Bakerloo line** – which runs from south-east to north-west London. It is 23.2 km long, with around 302,869 journeys per weekday. We have the data from 20 stations, covering over 950,000 journeys.

All data was anonymised by TfL, and contained the following fields for each journey:

```
<uid> <day> <in_time> <in_station> <exit_time> <exit_station>
```

The uid gives a unique user identification number and the day field is simply a numerical determination of the day of the journey. The time a user passes the station entrances gates (in_time) and leaves ($exit\_time$) are provided, both are accurate to one minute. The stations ($in\_station$ and $exit\_station$) are also given as numbers, which correspond to a lookup table that was provided with the data. 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. The method of transformation to a colocation based format is given in Section 5.1.2.

Unitrans

The traceset was downloaded from the Crawdad repository together with the subsequent two traces. The Unitrans trace subset *Run3* was collected on the Unitrans bus system at University of California, Davis, USA in early 2006. *Unitrans* is student managed and operated transit system for the university, performing over three million passenger trips per year. Davis is a small city with a large university population, Unitrans is controlled by the student population and as such the passengers are biased towards the student population. Scanning devices (Intel iMotes) were fixed into 33 buses giving a trace of all the bus routes of the town throughout a five day period. The static nature of the placement of the devices is however not exactly comparable to our scenario as it captures user/vehicle colocation rather than user/user. Due to the small size of the buses (some are ex-London Transport double-decker buses), we assumed that two hosts in range of the bus are also in range of each other. The Bluetooth inquiry scan phases lasted for 5.12s, they were repeated every 2 minutes, giving quite a fine granularity of device detection. The traceset is separated into a file of detections from each iMote scanner, in the format:

```
<vendor> <uid> <start> <end> <since_this> <since_any>
```

The fields are defined as follows: $vendor$ is the manufacturer portion of the BD_ADDR for the Bluetooth device seen in this contact; uid is the anonymised device portion for the detected device; $start$ defines the start time for the contact (in seconds); end is the end time for the contact (also seconds). The field $since\_this$ represents the amount of time passed since this iMote last saw the device in this contact, and $since\_any$ is the amount of time passed since this iMote saw *any* other device before this contact. The last two fields are not of any interest to this work. Some lines contain all zeros, indicating that a reset occurred between the previous line and the following line. The ramification of this is that for data collected after this time, we can no longer be sure about synchronisation between this iMote and the other iMotes. The format can easily be converted to a useful format for us by simply adding a field to show the iMote device that the detection came from. Erroneous lines were omitted if they did not contain all values or if the MAC address was corrupt.

Reality Mining

This dataset was collected at the MIT Media Laboratory during the 2004-2005 academic year and contains over 350,000 hours of data. The experiment was performed by giving Nokia 6600 smartphones equipped with monitoring software to one hundred people (students and faculty at MIT). Though many statistics were recorded, we only use the Bluetooth colocation information. The inquiry scans were repeated every 5 minutes giving reasonable fidelity to the detected colocation patterns, while limiting battery usage. The extremely long deployment of this study provides the invaluable ability to analyse long term relationships between nodes. All participants spend their days in a similar geographic area, and do not provide particularly varied subjects. As the study lasts for such an extended period, the devices were subject to power failures, approximately 2.5 times on average per month according to the subjects. This, combined with intentional phone deactivation and data corruption lead to 85.3% coverage of the deployed time. Devices achieved an average of 36 hours of standby time. Colocation data was recorded in the following format:

```
<uid1> <end> <start> <uid2> <device_id>
```

One fifth of the subjects manually turn the phone off on a regular basis during specific contexts such as classes, movies, and (most frequently) when sleeping. From surveys, they found that 30% of our subjects claim to never forget their phones, while 40% reported forgetting it about once each month, and the remaining 30% state that they forgot the phone approximately once each week.

Haggle

This data was collected as part of the Haggle project at the University of Cambridge during 2005. Bluetooth sightings were recorded by 36 users, carrying small Class 2 Bluetooth enabled devices (iMotes), for just under 12 days, in office and conference environments. The inquiry scan length was also 5.12s with 10 minutes between scans, arguably a slightly coarse granularity that will not catch very many short colocation periods. Using iMotes, had the advantage of the devices being dedicated to the task of recording the mobility, avoiding the problems of users turning off the device for reasons not connected with the study.

The Cambridge Haggle project traceset is different from the others due to it not including singular sightings, and thus colocations of length 0. Though this focuses the data on actual colocations and removes the effect of the very transient host churn, it skews the data, increasing

both the mean and standard deviation of colocation duration. The format was as follows:

```
<uid1> <uid2> <start> <end> <since_this> <since_any>
```

The first column $uid1$ gives the identification of the device who recorded the sighting, and the second $uid2$ giving the identification of the device that was seen (it may be another iMote, or an external device). The third and fourth column describe, respectively, the first and last time when the address of $uid2$ was detected by $uid1$ in this current colocation. The fifth and sixth column are not of interest to this work. Note that the contacts may not be completely mutual between a pair of iMotes, because scanning period of different devices are not synchronised, and because the sightings might not be symmetric.

All of these trace files were then converted in to a common format suitable for our event based simulator, with each line representing a connection event or disconnection event:

```
<uid1> <uid2> <(dis)connection> <time>
```

For all the traces except TfL, this is simply a textual translation, creating two events from one colocation line with the appropriate columns. For the TfL traces, deeper analysis and transformation must be performed. This process is described in the following section.

5.1.2 Passenger Colocation Generation

The TfL dataset only records when/where people entered and exited the subway system, and not who they were colocated 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 about the nature of people's journeys. To process the journeys into colocations, each pair of journeys is analysed to see if their trains overlap in time and space.

We assumed that each passenger spends a constant amount of time entering and leaving any station. Also only colocations that occur on the platform or on trains moving in the same direction and on the same line are valid. To discern if two passengers are colocated we must find which passengers are on the same train. Therefore, we need to know where each passenger is and at what time.

A user's position is determined by working backwards from when they leave the transport system. They leave at time t_{exit}, then assuming a constant station negotiation time $StnExit$,

it means they alighted their train at $t_{alight} = t_{exit} - StnExit$. We did not use the alighting time to assume which specific timetabled train was used, as there is significant variability in specific train movements compared to the official timetables. With the alighting time known, using official TfL train timetables we can still accurately discern travel times between stations. In fact due to diurnal patterns inter-station travel times vary through the day, hence we used an hour specific inter-station travel-time matrix created from the official timetables. With the knowledge of alighting time and inter-station travel time, we are able to work backward through a user's journey to determine when they were at each station on their journey path. Thus, a user exiting Brixton station at time t_exit will be estimated to have been at Stockwell station at the time $t_{stockwell} = t_{exit} - StnExit - travelTime(Brixton, Stockwell)$. Any additional time that a user has from when they entered the system (t_{entry}) to when this methods predicts they left that station on a train is assumed to be time spent waiting on the platform.

Once user position and time is known, it becomes possible to intersect all journeys in time and space to determine which users are occupying the same spatio-temporal position and hence are colocated. If any pair of users were traversing the same station within $TrainThresh$ of each other, they are deemed to have been on the same train. This threshold allows for a small amount of inaccuracy in the position timing algorithm and accounts for the fact if two users traversed a station 1 minute apart and at that time trains were coming every 10 minutes, then they were probably on the same train.

If two hosts are deemed to be on the same train, then they may still not be in wireless range of each other. The hosts physical position on the train is determined, uniformly randomly across the physical space. The rolling stock (tube trains) in use on both the considered lines is roughly 128 meters in length. To give a comparable amount of host density, considering that the data only covers roughly a quarter of passengers, we have shortened the train by the same amount. Hence, each host is positioned on its train and any other passengers on the same train and within 10 meters will be colocated. Different runs of the experiment will have users placed in different positions on the train to avoid artefacts in the generated traces. Algorithm 5.1.2 gives a formalised version of the processing used.

Note that we are assuming almost total penetration of devices running the proposed service. Though an optimistic proposal, it is in some ways the pathological scenario for achieving successful file transfers as there will be the maximum network contention. It will however also also enable the learning of other host's movement patterns slightly easier (as they will be seen more frequently).

Our testing (Section 6.1.3) found connectivity to sometimes be possible between adjoin-

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($T_1 - T_2$)≤**TrainThresh 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)
      T_3=timeAtStation($path_{end}$,i)
      T_4=timeAtStation($path_{end}$,j)
      C_{end} = min(T_3,T_4)
      recordColocation(i,j,C,$path$)

Algorithm 5.1: Same Train Algorithm.

ing carriages, and also over 10 metres if the carriages were not busy. However, we consider Bluetooth connectivity range to be the specification rating of 10 metres for Class 2 devices. Its relatively short range means that even passengers on-board the same subway train will not necessarily be in range. We elected to distribute hosts uniformly randomly throughout the trains, a simplification for purposes of analysis, and to take into account that people naturally spread themselves through the train to avoid physical proximity. Some of the users in the traceset displayed extremely divergent behaviour, entering and exiting a station many times per hour. Further inspection showed that these users had staff oyster cards, and were thus performing duties at work inconsistent with normal passengers; therefore, they were removed from the dataset.

5.1.3 Trace Analysis

Previous work shows wireless traces often exhibit similar unifying features, such as Karagiannis *et al* [KBV07]. Some statistics of all utilised traces will now be presented and compared. Also, as the generated TfL colocation dataset is derived from real movement traces, and not collected from actual radio devices, we have validated that it possesses similar features. The TfL distribution of colocation duration is shown in Figure 5.1. 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 journeys that people make, with the most popular journeys lasting around 12-15 minutes. Also, the relatively low occurrence of short contacts is due to people walking or using the bus rather than

taking the tube for short journeys.

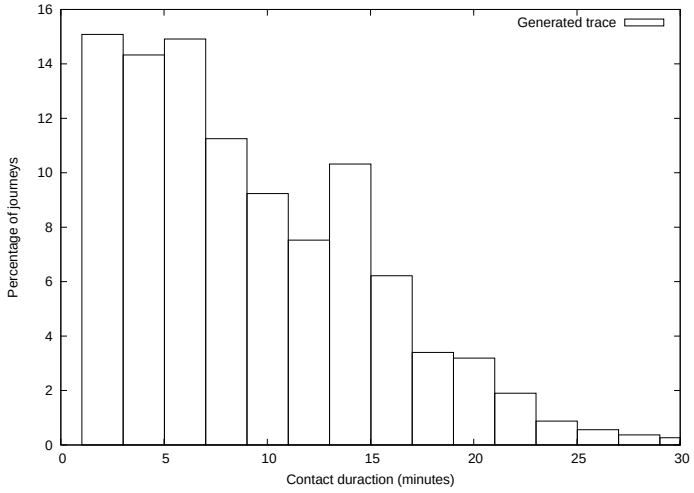

Figure 5.1: TfL Contact times.

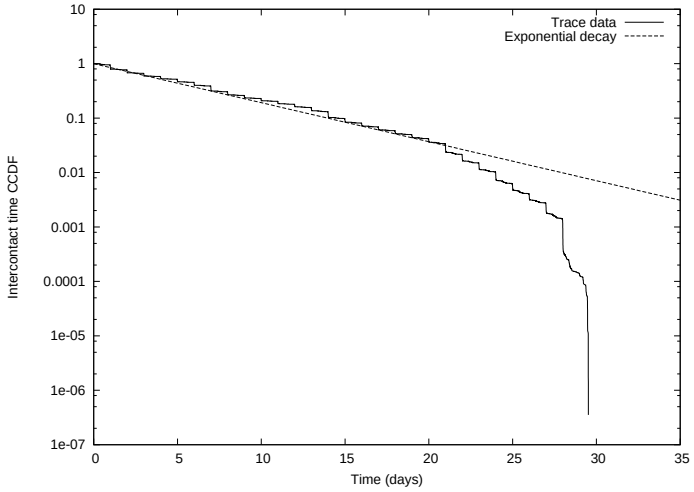

Figure 5.2: TfL Intercontact time CCDF.

The time between the contact of two nodes (inter-contact time) is of particular importance for DTNs. Our system does not consider the time messages take to move from one host to another. Though it is interesting to see that the intercontact time shows a clear fit with an exponential decay $e^{-\lambda t}$, where λ is approximately 0.165 below periods of 20 days, the data set is not long enough to give reliable data after this period. This matches with Karagiannis [KBV07] observation of exponential decay in the tails of the intercontact complementary cumulative colocation distribution (CCDF) (Figure 5.2).

The differences in colocation length distribution is plotted in Figure 5.3, showing the complementary cumulative distribution function of colocation durations. All the traces show a

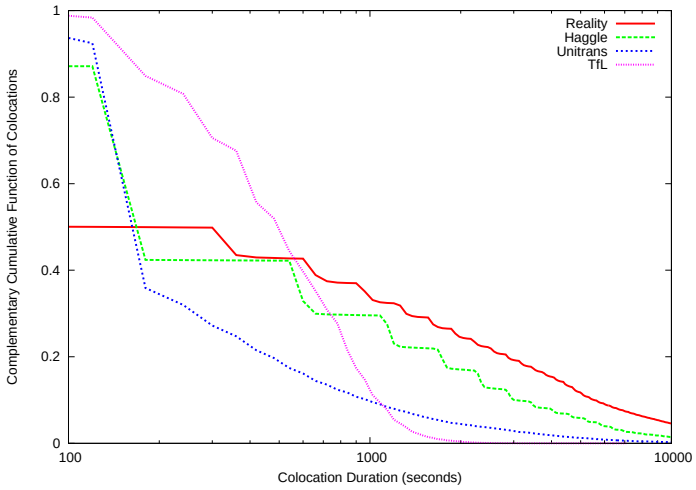

Figure 5.3: All Trace's Colocation Durations.

Statistic	TfL	Reality	Unitrans	Haggle
Duration (days)	30	365	5	11.7
Colocation Count (1000s)	29,577.8	192.6	11.7	17.9
Mean Colocation (seconds)	540.38	2012.23	466.16	1300.84
Min (seconds)	1	1	4	1
Max (seconds)	7,080	125,174	25,051	207,234
Standard Deviation (seconds)	15.83	108.02	88.63	136.87

Table 5.1: Movement Trace Statistics.

very rapid decrease in colocation length at shorter times (under 1000s), with a small number persisting for long times (above 3000s). The pattern for all traces is roughly an exponential decay. The Reality Mining data shows much longer colocations than the other traces, but still has a similar near exponential decay curve. This is due to the traces being taken over such a long time, and the nature of the environment they were gathered in.

All the traces contain many short transient colocations, especially Reality Mining. Identification of these short transient connections (so they are avoided) will increase expected colocation duration, allowing transfer of larger amounts of data. It will also reduce situations where colocation terminates before a transfer completes wasting the energy put into the transmission, and time spent negotiating this transfer.

The average colocation length when split into hours of the day is shown for Reality Mining and Unitrans in Figure 5.4. The Y axis is the average time of colocation for that hour relative to the mean length of the colocation for all the traces. As expected there is considerable change

over the course of a day, with colocations lasting longest when they begin at hour 15 (3 PM) for both traces. This similarity is encouraging as it shows that both trace sets have the same underlying properties despite being collected in very different environments. Interestingly, for both traces most colocations start in the same hour (4 PM), with the smallest number beginning the early hours of the morning when people were likely at home and not travelling through the city.

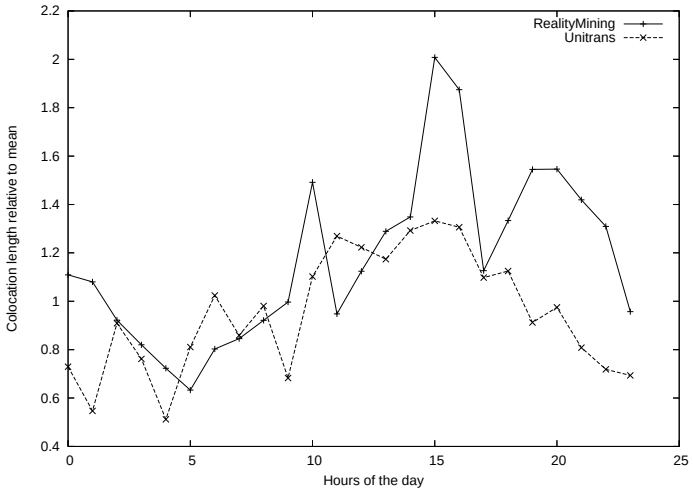

Figure 5.4: Colocation Length by Hour.

Journey Attributes

We begin our analysis of the TfL dataset by studying the changing nature of journeys through the day (Figure 5.5). 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 (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 night time 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 short ones.

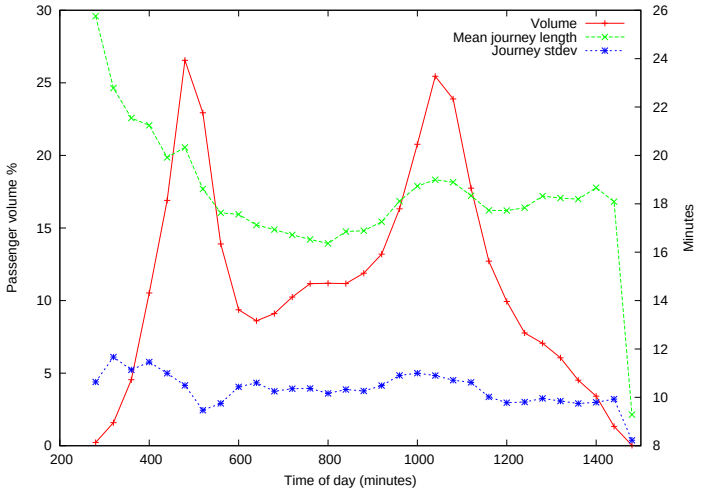

Figure 5.5: Volume and Duration of Journeys.

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 tube only a few times during the sample, with over 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 5.6 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.

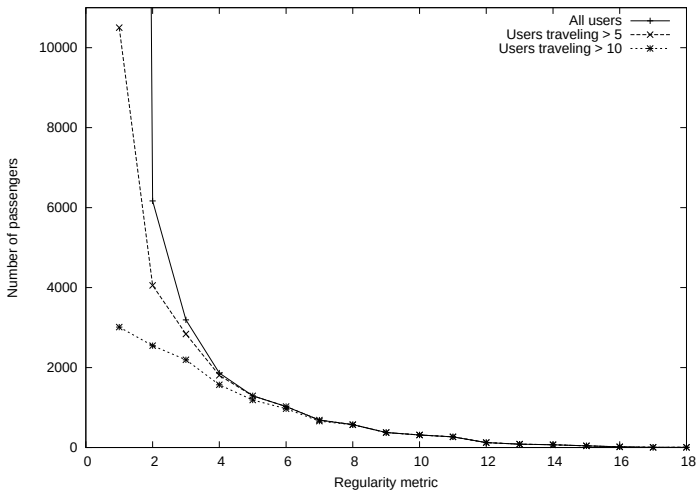

Figure 5.6: Passenger Journey Regularity.

5.2 Content Library Dataset

In order to simulate a music sharing system thoroughly, the tastes and libraries of users must be assigned and populated, respectively. 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 considered timeframe, users do not change their interests. Previous studies have recognised file availability in peer-to-peer systems to follow a generalised-Zipf distribution [SW04]. In all content distribution systems there will be a small number of popular items and many items that are much less popular. To verify these observations, particularly in the scenario of digital music libraries, we have collected data from Last.fm [las09]. Last.fm is a popular UK-based music community website, that claimed over 30 million registered users in March 2009. It creates profiles of musical tastes by tracking what songs users listen to and suggests more music. Users are also encouraged to attach *tags* to artists and tracks, thus creating a *folksonomy* of music classification [HJSS06].

We collected this information both to associate music tastes to users, and to create their 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. 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 mislabelled or relabelled).

5.2.1 Last.fm Crawler

We assume the user population of Last.fm will share a similar demographic to that of a wireless music sharing system. This seems an acceptable assumption given that Last.fm's users are technology literate music fans, and thus represent a useful case study. Unfortunately, there is a certain bias of detail and accuracy towards artists that are more popular, as they will be tagged more frequently and thoroughly. The quantity and appropriateness of this data still gives it immense value for our purposes. Using their Audioscrobbler Web Services[1] to access the website's user/artist data, we performed a large breadth-first crawl of 500,000 user's profiles, together with their top 50 most played artists. The links for navigation between users were chosen to be their *friends* rather than *neighbours* of the profile. The friends represent people that a user has decided to add manually, rather than the neighbours, which is a list of 50 people that are deemed by Last.fm to share similar tastes. This was done to mitigate only collecting users of similar tastes, leading to a relatively homogeneous user set. The webcrawler was also initiated from many randomly chosen starting points, to further ensure an unbiased set was obtained. For each user, several items of information were gathered, *friends*, *neighbours* and *top played artists* lists. The top played lists is defined over five different time frames, *Last 7 Days*, *Last 3 Months*, *Last 6 Months*, *Last 12 Months* and *Forever*. For each artist, we also crawled its 50 most frequently associated tags. 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, these tags will simply be leveraged by our approach. All this information was then stored in flat files for utilisation in library modelling by the simulations.

5.2.2 Last.fm Data

Each facet of the music libraries is explored in this section, specifically the *genres*, *artists* and individual *tracks* that occur in user's libraries. Out of all the users that were crawled, 68% have the full 50 items in their list, the rest have not listened to enough tracks to fully populate the list.

Genre

The differing categories of music is one of the most important facets of the data collected from this source. For all forms of generalised *content*, the categories that individual items can fit in to are of great importance to any mechanism used to distribute them, similar to how music is usually arranged by genre in physical music stores. The distribution of how popular different

[1] Last.fm Audioscrobbler Website: `http://www.audioscrobbler.net`

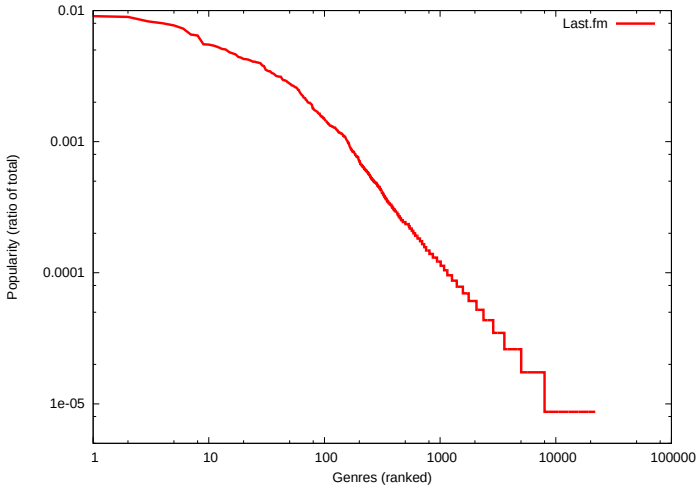

Figure 5.7: Genre Popularity.

Rank	Genre	Popularity (%)
1	Rock	14.7
2	Electronic	11.4
3	Alternative	11.2
4	Metal	10.4
5	Indie	10.0
6	Punk	9.7
7	Pop	9.2
8	Hardcore	8.3
9	Electronica	8.1
10	Hiphop	7.0

Table 5.2: Artist's Genre Popularity.

categories are will now be discussed. Figure 5.7 depicts the amount different categorisation tags are used in the Last.fm data we gathered. Genres are arranged by decreasing rank on the X axis, with their corresponding frequency of appearance on the Y axis. It is immediately apparent that the most popular genres occur vastly more often than the more niche classifications.

By ranking tags in order of their frequency across the whole crawl, we were able to rank 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*).

Despite the majority of files belonging to a few popular genres, a significant proportion are

Rank	Artist	Popularity (%)
1	Radiohead	0.49
2	The Beatles	0.38
3	Muse	0.35
4	Red Hot Chilli Peppers	0.34
5	System of a Down	0.32
6	Linkin Park	0.32
7	Metallica	0.30
8	Placebo	0.30
9	Nirvana	0.30
10	Coldplay	0.28

Table 5.3: Overall Artist Popularity.

not; more importantly, many users are not interested in any of these mainstream categories. A truly useful and fair distribution mechanism should not solely focus on these mainstream genres, niche tastes must be catered for as well. One interesting aspect of user taste distribution, is that people with mainstream tastes are more likely to also have homogeneous interests; e.g., a person that likes *pop* and *rock* music is more likely to only enjoy these two genres. Whereas, a user liking *nerdcore*, *darkwave* or *industrial* will often enjoy many other genres besides. This shows that people with niche tastes are also likely to have a more eclectic taste. This is advantageous, as if a user with obscure taste has a specific niche interest that can not be satisfied by the system, it is likely that the user can be satisfied by collecting one of their other many interests.

Artists

In almost all areas of media, the creator of a piece is of great importance; whether it be a song's artist, a film's director or a podcast's maker. Usually if a person likes one item of content by a creator they will like more by the same person(s). Though it would seem advantageous to record the artists that a user enjoys, and attempt to procure more of those artists, we decided against this behaviour. Due to a user possibly owning an artist's full discography, yet only storing one album on their device. Thereby leading to already owned, though not currently stored, albums being downloaded, when they were intentionally not placed on the device. If a system was to then explicitly try and get more files by that artist, when the user has all other albums stored elsewhere, useless work would be performed. Therefore, we believe it is preferable to only attempt to collect more files of particular genres of interest to a user to mitigate this risk. This

Rank	Rock Artist	Popularity (%)	Metal Artist	Popularity (%)
1	Radiohead	0.980	My Chemical Romance	1.136
2	The Beatles	0.781	KoRn	1.043
3	Muse	0.711	Nine Inch Nails	1.035
4	Red Hot Chilli Peppers	0.688	Nightwish	1.001
5	System of a Down	0.656	Rammstein	0.940

Table 5.4: Genre Artist Popularity.

approach will also promote a greater understanding of the wider musical genres that the user identifies with, broadening their taste but in an accessible way. The most popular artists and their respective popularities are shown in Table 5.3.

Within different genres, different artists are deemed popular, as would be assumed; this can be seen in Figure 5.2.2. The two musically similar genres $Rock$ and $Metal$ are shown with their most popular artists, with the percentage popularity, from each. The percentage popularity is defined as the percentage of tracks that fall into the relevant genre that are performed by the respective artist. It can be seen that despite the loose similarity between the two categories of music, not only are none of the artists the same, but there is a difference in how the popularity is distributed. This can be seen more clearly on the graph in Figure 5.8. Each genre had a count made of matching artists from the Last.fm dataset, the artists are arranged by order of rank on the X axis for each genre, with frequency on the Y axis. It can be seen that there are a few very popular artists in each genre, with the popularity trailing off to a large tail. The less popular genre of $Punk$, has a few relative 'superstars', whereas the ubiquitous $Rock$ is much more evenly distributed. In fact, the more popular a genre, the greater the relative popularity of the often played artists. This is demonstrated by the steeper curve for $Punk$.

Tracks

Each artist/creator has an oeuvre of content items, which could be divided into collections (e.g., albums) or simply individual pieces. Rather than restrict ourselves to types of content with many explicit collections (such as music rather than podcasts) we treat each item of content as independent. Similar to the problem mentioned above, if incomplete albums were prioritised to be gathered in their entirety, a user with only their favourite tracks from an album would always be destined to gather the rest, no matter their avoidance of the disliked album tracks.

If the system is to model down to the granularity of individual tracks, then there must be a way to distinguish them, to know if a track is already owned. However, collecting name and

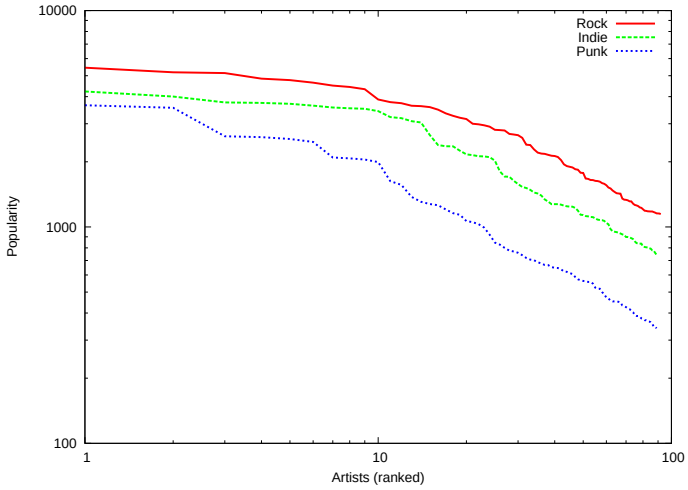

Figure 5.8: Genre Artist Popularity.

popularity information about every track by every artist would not only involve the collection and processing of a lot of data, but that information is not available from Audioscrobbler (only an artist's top 50 tracks are accessible). Therefore, we decided to generalise the track name property: a track's name will be represented by a number, unique to the artist. More important than the particular name of a track, is its frequency of occurrence in user's libraries, so that an accurate model can be built of an artists discography.

An artist's tracks differ in their relative popularity: nearly all artists have some tracks that are more popular than others. This distribution is important because it affects how likely it is that two hosts that share interests, will have matching files that are not already owned. The distribution of relative popularity for each artist's ranked track is simply calculated by normalising according to the most popular track of that artist, thus ignoring the absolute track popularity. A general distribution of track popularity can then be calculated for all artists by taking the average proportion of each $rank$'s position:

$$\sum_{n=1}^{artistCount} relativePopularity(artist_n, rank)/artistCount \tag{5.1}$$

For example, if on average the second ranked track is only played half as often as the most popular track, it will have an overall relative popularity of 0.5. Figure 5.9 depicts this relative popularity of tracks for all artists, and separate stratified plots for popular artists and small niche artists. For the combination of all artists, a curve was fitted programmatically resulting in the equation:

$$y = -0.2 \times log(x + 1) + 0.85 \tag{5.2}$$

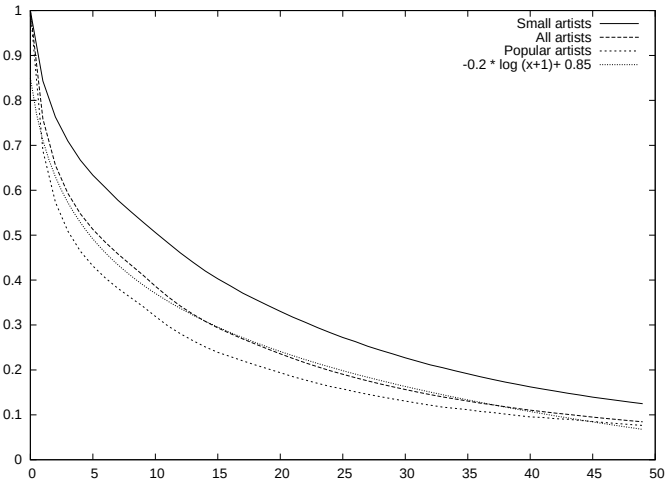

Figure 5.9: Track Popularity.

It can be seen that artists' tracks appear to decrease asymptotically in popularity, a property that holds for every artist individually inspected. The only difference between artists is the rate of this popularity decrease; the artists were thus divided in to *popular* acts and *niche* ones. Interestingly, it can be seen that tracks from a niche band decrease in relative popularity slower and are therefore more uniform, indicating that the very popular artists have a small number of extremely popular items. Conversely, smaller bands have fans that display a wide interest in all of their music. Equation 5.2 is used in the simulations (Section 5.2.4) to generate which tracks exist in a user's library, as described in the next section. Again this is not meant to be an observation of the nature of all music artist's discographies, but just a trend observed in our collected data, that we are using to make our evaluation have more of a grounding in digital music consumption.

5.2.3 Content Library Modelling

If an accurate appraisal of a content sharing system is to be performed, the content libraries and interests of users should be modelled. This allows users to have an interest formed of only a subset of the population of all content items. If done realistically, it gives our formulation of the problem a more accurate representation of the differing popularity distribution across different data items. Despite this desire, the exact properties of music interest and libraries is not in anyway inherent to our model, it is only used to evince the scenario more effectively. Rather than simply assume that a suitable file can always be shared with another peer, we model a file's genre, artist and the specific track. This is an important level of detail if we are to be able to realistically model a large urban peer-to-peer music network. People have their own interests in

for each emptyTrack t **do**
$genre = choose(G)$
$t.artist = choose(genre)$
$t.trackname = rand(0, 10)$
$tracklist = tracklist \bigcup \{t\}$

Algorithm 5.2: Proportional Track Selection Algorithm.

a subset of the files in the system as a whole, the size and distribution of these subsets should have a bearing on our designs. A prerequisite for a successful transfer is that the *source* has a file that the *downloader* is interested in and that it does not already have. The distribution of interests and files will therefore have a large effect upon the success of transfers and quality of those transfers in the network. Moreover, the system performance should be considered with respect to individual user's properties. We should consider whether users with unpopular tastes still receive some benefit from the system, or are their concerns marginalised in favour of *mainstream* users. The same consideration should be applied to individual tracks, and whether only the 'hits' of an artist are shared, or does their whole discography get spread.

Our initial aim was to investigate general content dissemination, including music, video and news; explicitly modelling music files loses the generality of this aim. However, music is probably the most available and desired form of media for the consumer electronic devices that we envision a system such as ours being used for. Phones, PDAs, laptops and PMPs all have the capability for playing sounds, even if they do not have a visual display. Also people can more easily solely listen to audio rather than watch some accompanying video. There is also more available data about what music people listen to using their computers than all the short video clips and news pieces that are accessed with current technology.

Initially, it was considered that it would be appropriate to generate a list of library files for users by simply choosing a *genre* for the track according to the probability of its occurrence out of all crawled tracks. An *artist* is then selected according to its probability of appearance in the selected genre, as in Algorithm 5.2. Upon analysis of the libraries this approach created, it was obvious that the libraries were not very convincing examples of real user's tastes: extremely eclectic sets would be generated, with no consideration of the likelihood of a particular user's underlying taste. For example, nearly all libraries would contain a track from the *pop* and *rock* genres. Therefore it was decided to employ a more cohesive method of generating libraries.

5.2.4 Library Generation

Each simulated user gains the tastes of a random user u_j from the Last.fm dataset. The Last.fm user's top 50 artists A is used to define a simulated user's genres of interest. Each artist a_i from this list has an associated user preference, $P(U_j, a_i)$, derived from their play count of that artist. An approximation of the user's music taste can be derived from the artists in this list. The most popular category associated with each artist is added to the user's interest list. This list defines what categories of content a user is willing to accept.

Each user u_j has a parameter setting its initial library size, $libSize$, which is uniformly distributed in the range [$libSizeMin$,$libSizeMax$]. The library is instantiated with $libSize$ files, by performing the following process until the library is of sufficient size.

- Choose a category from the user's interest list Cat.
- Probabilistically choose an artist from that category, in proportion to how popular an artist is within Cat (stratified random sampling). This causes more popular artists to be chosen more often, and allows an artist to feature in multiple genre lists.
- Once the file's artist has been chosen, the particular track must also be selected. Rather than attempt to represent all individual tracks within our system (and require names and popularity to be gathered for all artist's music), we simplify this stage for analysis. Track 'names' are represented numerically, and are chosen from the distribution $Y = -0.2log(X + 1) + 0.85$, where Y is uniformly randomly selected from $Y \in [0, 1]$; as demonstrated in Figure 5.9.
- The generated file is added to the user's library, which will ignore it if the file is already present.

The dataset's libraries follow a clear distribution, as show in Figure 5.10. A large set of users are ranked along the X axis according to the cardinality of their interest lists. It is easily seen that most hosts are interested in between 30 and 60 categories, with a minority outside of this range.

5.3 Summary

This chapter presented the datasets that will be used in the following evaluation section. The movement trace's places of origin was described to give an indication of the type of human movement that they capture, particularly the information collecting mechanisms. To allow objective comparison some basic statistics of the data was covered including number of hosts,

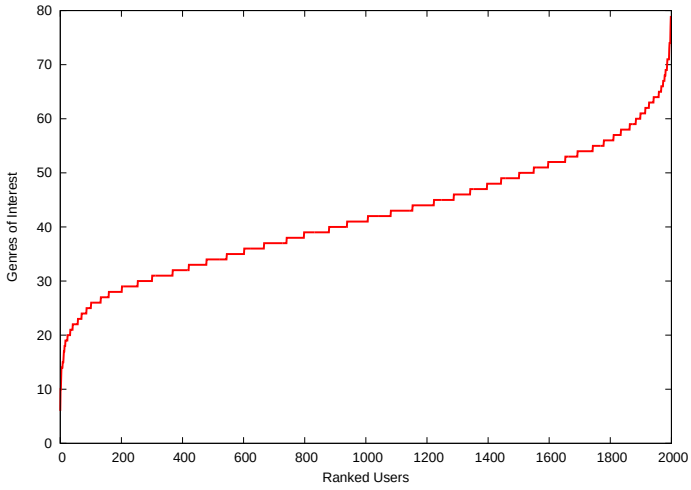

Figure 5.10: Interest Size Distribution.

colocation durations and inter-contact times. Our main dataset of the subway journeys was covered in detail and, crucially, how the raw journey information was converted to colocation information. This colocation information was then compared to the available true Bluetooth scan datasets to ensure that similar properties were observed. The music library dataset was also presented, together with how we collected this information and how it is used to create realistic user libraries of music.

6

Simulation Evaluation

The evaluation of our system by use of a thorough deployment and testing on a city-wide scale would be extremely challenging, requiring thousands of man-hours and a large budget for all the devices. Therefore, a method must be employed that can analyse the behaviour of such a system without physically deploying it. Often researchers achieve this using network simulators, which can emulate the behaviour of many devices interacting with each other and record their actions. This chapter describes an implementation and test deployment of a prototype system that was developed (Section 6.1). The information learnt through the development process and measurements collected was the used to inform large scale simulation of our scenario. Section 6.2 describes the choice of simulator, its configuration and how it performs the algorithms defined in previous chapters. The results from various parameterisations are then discussed to give an indication of how our proposed system behaves under varying network and user conditions.

6.1 Implementation

This section documents the creation of a software implementation of the system proposed in this thesis. Design concerns and decisions are mentioned, together with the results of testbed measurements from running this code on real devices.

6.1.1 Mobile Phone Applications

Mobile phones can be a much more challenging platform to develop software for than fully functional desktop/laptop computers. Not only are resources (e.g., CPU/RAM/storage) much scarcer, but the operating systems are more limited in their functionality and flexibility. The rigidity of the programming interface and process lifecycle is not purely because of hardware limitations. Phones are similar to embedded systems, in that program failures should be avoided and programs are often expected to run unattended for long periods of time. The device (and its software) needs to be extremely robust; to the user it must seem like they *just work*. Therefore access to hardware must be carefully handled, and often heavy security restrictions are placed upon programs and devices.

6.1.2 B2B

We implemented our content distributed program for the popular Symbian S60 platform[1]. It was named **B2B**, short for Bluetooth-to-Bluetooth, and was created in less than 2KLOC for the PyS60 v1.4.3 Python interpreter. 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 on-board memory (32MB), allowing the storage of large media libraries.

Python was chosen to facilitate rapid development and allow easy porting to another platform. It also avoids the restrictions of the Java security model, such as requiring application signing or only allowing sensitive operations (such as a Bluetooth scan) with user confirmation. Access to low level operating system functionalities can then be provided by Symbian C++ modules that are able to interact with the higher level python system. We utilised code from the Personal Distributed Information Store (PDIS) project[2] 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 let the phone perform other tasks, while files are shared. This allows a user to start the program and leave it collecting files without managing the application, a crucial behaviour if a user is to be able to run and forget about this application. Presently it does not have a Graphical User Interface (GUI), information is presented textually, informing the user of neighbour detections and file download progression. Except for starting the program, the user does not input any information, all behaviour is automated.

[1] Symbian website: `http://www.symbian.com`

[2] PDIS Project homepage: `http://pdis.hiit.fi/pdis/`

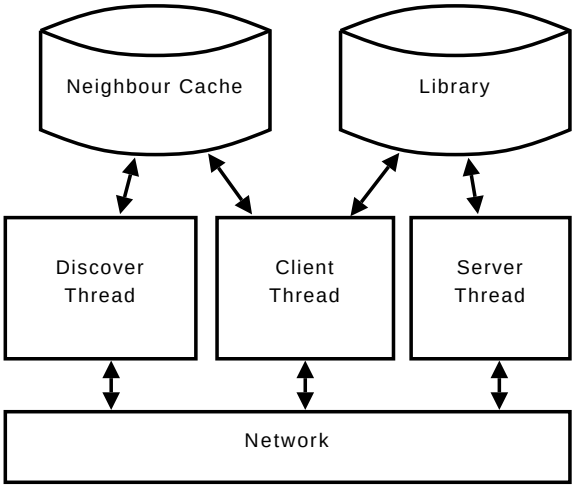

Figure 6.1: Implementation Architecture.

An important aspect realised during the development of the implementation was the effect of the memory card write speed. Writing data to the memory card one block at a time was an order of magnitude slower than ideal network speeds, which due to the blocking nature of the write calls, slowed the transfer speed significantly. The smaller on-board memory was much quicker, providing a feasible buffer to store the file while being transferred. This approach only works for files small enough to fit on to available on-board memory. Once a file has been completely transferred to on-board memory, it may be written to the memory card with no penalty for the sending node; due to DMA-like behaviour, writing the whole file in one go is much quicker than block-by-block.

Upon startup the application searches the predetermined directory (`E:/Music`) for MP3 files, and reads each file's ID3 tag to build an internal meta-library. The fields *title*, *artist*, *genre* and *file path* are stored. Any files with unparsable meta-data is ignored. Each genre that exists in the library is added to the device's *interest list*. The ID3 meta-data of MP3 files is read using the Python module `ID3`; only MP3 files are considered for this prototype, though extra file type support could be included without any loss of generality, e.g., Ogg Vorbis (OGG), Windows Media Video (WMV) or Advanced Audio Coding (AAC).

The architecture of the implementation program is depicted in Figure 6.1, showing three threads that interact with two information stores. The operation of each of these threads will now be briefly described.

Device Discovery

A *discovering* thread is started to periodically discover all neighbouring devices. For each discovery phase a non-user prompted device discovery is performed using the aosocket interface. Every device that is discovered that is not currently deemed to be *colocated* has a service discovery performed upon it, to check it is a B2B device, and thus interested in sharing music. It then has an entry added into the list of currently colocated devices. Existing B2B neighbours simply have their last seen time updated to the current time. Any devices that are in the colocated list, yet have not been detected during the past two discovery phases are considered to have departed. Their colocation profiles are updated correspondingly, as described in Section 3.3.2.

Server

A separate *server* thread is created at start-up to listen for connection from other nodes desiring content. An RFCOMM port is bound to by the *server* thread, to listen for connecting sockets. This server thread does not operate as a standard child-spawning server, multiple clients will not be served at the same time, as with powerful Internet connected services. Upon connecting to a server, the downloader sends a textual, comma separated, newline terminated list of genres to initiate the download process. The server executes the following procedure:

1. Read genre list. Search meta-library for matching genres.
2. Prune already advertised files. Sort by policy and write shortlist.
3. Add shortlist to advertised file list.
4. Read file "`Artist-Trackname`" request.
5. Open the file at related *file path*.
6. Write filesize then blocks of data to the socket.
7. Update library with successful send.

Any socket errors, possibly caused by the pair moving out of communication range, are trapped and the transfer attempt ended. The thread then resumes listening, waiting for the next connection.

Client

The *client* thread performs the actual content pulls, regularly selecting the best neighbour and attempting to initiate a download of content matching a category from its *interest list*. The colocated list of B2B devices is consulted, and a colocation duration prediction is made for

Action	Duration (seconds)
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 6.1: Action Timings.

each device. Devices are contacted in descending order of their predicted future colocation, waiting for a host to respond, i.e., its server thread is available for providing a service. The best available neighbour has a connection made to the RFCOMM socket, and the negotiation is initiated.

Upon receipt of the shortlist, the already advertised list is updated for this host, to avoid advertising any of the files back to the server in a future file negotiation phase. Any file that is already in the library is removed from the shortlist. The first remaining file is selected as the desired download file. As a response to the request for this file, the filesize will be sent by the source. It can then be checked that the storage space needed is available. Enough space is needed on the internal drive and the memory card if a fast download is to be achieved. Files are downloaded to the fast internal phone memory and upon completion, moved to the memory card's music folder (`E:/music`) for long term storage. The meta-library is also updated with the new track.

6.1.3 Testbed Behaviour

To inform our choice of timings used in the simulation, we tested our prototype on an actual mass transit train, at both busy and quiet times. Table 6.1 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 compared between busy trains and near empty trains was significant, increasing transfer time by nearly 50%. If many fellow passengers were engaged in data transfers, time taken would increase even more. Detection of other hosts was also impaired by operating in a busy environment; this was likely due to how many hosts responded to a discovery (only a limited number can be discovered) and interference from other devices (such as from Bluetooth headphones).

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.

6.1.4 Range

The communication range of devices is integral to the system and the actual range of our testbed phones was examined. The proposed environment of underground urban trains, possess non-ideal features for wireless communication. Not only is it a confined area, it will have metal walls and, during rush hour, many human bodies in the volume. All these factors limit the electromagnetic propagation of radio waves. When operating in residential/office environments devices could discover each other with relative ease and despite attaining transfer speed well below nominal bitrates for Bluetooth 2.0 devices, is still consistent.

6.2 Simulator Details

There are many options for simulating a network, a generic class are denoted as *discrete event simulators* (DES), and they can be applied to a wide range of problems, whereas others are purely intended for analysis of computer networks. Some popular DES include ns-2, OPNET, GloMoSim and JiST [3]. We decided to use OMNeT++ [Var01], an open-source discrete event simulator written in C++. This particular simulation environment was chosen due to its many strengths:

- It is an established and mature framework that has been used in academic pursuits for over fifteen years.
- The open source code base allowed viewing and modification of its inner workings, something that was useful on many occasions during this research. OMNET++ is also free for academic purposes, allowing other researchers to easily confirm results using the same tools.
- The framework is well structured and component based, promoting good design in the problem specific code that was developed for this work.
- There is extensive and clear documentation of all components and accessible examples.

[3] Simulator reference websites; ns2: `http://www.isi.edu/nsnam/ns`, OPNET: `http://www.opnet.com`, GloMoSim: `http://pcl.cs.ucla.edu/projects/glomosim`, JiST: `http://jist.ece.cornell.edu`, OMNet++: `http://www.omnetpp.org`

- Its *Mobility Framework* models wireless channels and signal propagation, which was used during the early stages of this research (though the overhead was too great for the desired scale of our simulations).

Our simulator is divided into the following logical parts (described here to aid future explanations):

- **Host** – The most important module in the system, representing a single device in the simulation. It stores much of the state and contains all other components.
- **Library** – The container for user's music tracks, whose access and searching is very important for the performance of the simulator.
- **Chooser** – This module selects which neighbour should be chosen as a potential source of data (Chapter 3).
- **Trust Repository** – This module stores and maintains the trust valuations of peers that have been interacted with (Section 3.5).
- **Matcher** – This component determines which tracks should be advertised (Chapter 4).

6.2.1 Host Behaviour

The steps that the simulator performs upon initialisation and during normal operation are described below.

Start up

Before the simulation can begin, various sets of data must be read and processed to give hosts their initial characteristics and prepare the connectivity changes. Due to the limitation of computing resources, not all hosts can be simulated in the very large trace sets; thus priority of representation needs to be associated with all hosts in the trace set. The list of *interesting* hosts consists of all hosts from the relevant trace set, ordered in descending order of the frequency of occurrence. Frequency of appearance was chosen as the sorting metric over the total duration of existence in the system. This was to favour hosts that have many interactions rather than hosts that may just have a few static long term presences. We believe this will give a fairer temporal distribution of behaviour, and focus on the most dynamic hosts and relationships.

The manner that taste profiles are assigned to hosts from a movement trace is entirely random. This assumption of complete independence between a user's taste and their movement, is not easily justifiable. Unfortunately, there is no data available that maps, in a reasoned and

```
while wanting files do
  if neighbours == ∅ then
    sleep(Alone)
    continue
  prune malicious neighbours
  score neighbours
  rank neighbours
  perform transfer neighbours[0]
  update library with file
```

Algorithm 6.1: Download loop.

consistent manner, how people move and what musical content they enjoy. While not ideal, the effect of this is limited by the large scale of the simulation and having multiple simulation runs with differently seeded random number generators.

- **Interesting hosts** – A special (movement trace dependent) list of *interesting* hosts is read, these hosts are instantiated into the list of users that exist in the simulation.

- **User libraries** – All users libraries are created according to the process described in Section 5.2.4.

- **Connectivity trace** – As some trace sets are very large, possibly multiple gigabytes, the complete file could not be parsed upon start up. The relevant colocation trace file is opened, and five million lines of text are read. This text is then parsed into connection events, which are placed in the OMNet++ event queue. While running, whenever this event queue becomes empty, another five millions lines are read and parsed until the trace has been completely executed.

File Download

An equivalent process as described in Section 6.1.2 is performed when the system is running and hosts are active. Most important is the behaviour of individual hosts when they are trying to find peers and collect content from them. Though in actuality it is structured as an event-based system, the pseudo-code illustrated in Algorithm 6.1 gives a high-level approximation of the process. When a host has no available neighbours, it will sleep for 100 seconds ($Alone$) waiting on other co-interested peers joining it.

Wireless propagation effects are not modelled, though 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 this throughput reduction due to collisions, we use the analysis presented by Liu [Tin03], assuming all packets are DH5, thus dynamically reducing the throughput of a transfer according to the number of neighbours currently sending data.

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 speed 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 successfully completes, rather than performing a new search and selection, another download will be almost 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 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. A small delay is enforced before the next negotiation process (without peer selection), to account for any storage activity, such as what was observed in Section 6.1.3.

At the beginning of the simulation each host is classified as 'malicious' with probability $MalRate$. For each transfer that a malicious host provides, upon completion with probability $MalChance$ the file is identified as malicious. The threshold for deciding whether a peer is malicious ($TrustThresh$) is set to 0 (neutral trust). All statistics are gathered over the course of the whole experiment, there is no 'warm-up' or 'learning' phase. This means that all results are a worse-case scenario, with all hosts starting with no historical information. This was done to enable to longest possible operational phase and avoid the assumption of some hosts possessing a lot of historical information. Real performance is likely to improve upon the results presented

here, once patterns and behaviours have been learnt.

6.2.2 Parameters

To examine the operation of our system under varying conditions we changed a number of its parameters in a controlled fashion. Two of the most important aspects are the mechanisms used for source selection (*Chooser*) and item advertisement (*Matcher*). The Matchers were defined in Section 4.2. The implemented Choosers are:

- **Random** - An available neighbours with at least one matching interest is uniformly randomly chosen as the download source.
- **Slotted** - The available neighbour with the largest predicted outstanding colocation duration is selected as the download source. This prediction is made using the slotted temporal matrix, described in Section 3.3.2.
- **Oracle** - The available neighbour with the actual longest future colocation duration is selected as the source. This method is not achievable in reality as it involves knowledge of the future.

When labelling graphs the following format will be used to denote the varying peer selection and matching approaches being used: *Chooser*-[*Matcher*-]*Threshold*. Therefore, *Oracle-Random-0* represents the Oracle chooser being used conjunction with the Random matcher whilst employing no download threshold. When individual files are not being modelled (Section 6.3) the Matcher name will be omitted.

To ensure the simulations modelled reality as closely as possible, the timing and behaviour of our prototype was used to inform the selection of the simulation parameters. All experiments will use parameter values as defined in Table 6.2, unless specified otherwise. The full list of parameters used, together with an explanation of how we set them, is provided below.

- **Node number** – We have studied the effect of increasing the maximum 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 a real system. We chose to individually model the most frequently travelling passengers, as they stand to benefit the most from our system. This constraint was partially imposed by memory and processing limits.
- **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 $[1, 20]$ MB, to represent content from small music files up to larger video clips.

- **Transfer rate** – Indicative experiments we ran on underground trains showed achievable Bluetooth data rates at rush hour of over 100Kb/s, although the Bluetooth version v1.1 specification highest asymmetric transfer mode DH5, no Forward Error Correction (FEC), is rated at 723.2 Kb/s. In our simulations, the value for this parameter has a range of 100, varying from $[50, 150]$ to $[1500, 1600]$ Kb/s to cover a full range of Bluetooth specifications, 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.

- **Download threshold** – A parameter for how cautious hosts are about initiating a transfer. In general, the higher the value, the longer the predicted colocation must be, to actually start a download. Note that when this parameter is set to 0, downloads are always attempted if there is any available source.

- **Time slice** – This parameter only has an effect when using the Slotted Peer Selection technique, as described in Section 3.3.2. 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 2 hours (thus 12 slots in a given period). Using fewer slots requires less state being maintained about each familiar stranger. Hence, the times of groups could be dynamically learnt using a lightweight technique such as the one described in [DM07].

- **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.

- **Advert length** – The length (in tracks) of advertisement list sent between hosts during the negotiation phase. This parameter was to 20, as lists of any greater length did not give significant benefit.

Parameter	Distribution	Value
Node Number	n/a	2000
File Size	normal	μ=5MB, σ=1MB
Transfer Speed	uniform	[400,500]
Download Threshold	n/a	1
Time Slice Count	n/a	12
Library Size	uniform	[50,150]
GenreCount	uniform	[5,20]
Trust Threshold	n/a	0
MalRate, MalChance	n/a	0

Table 6.2: Canonical Parameter Values.

- **Malicious Rate and Chance** – To control the malicious behaviour of nodes in the system, two parameters are used: $MalRate$ and $MalChance$. $MalRate$ is the rate that normal hosts are classified as *malicious* and $MalChance$ is the probability that a malicious host will provide a file that is deemed to be malicious.

6.2.3 Metrics

To measure the behaviour of individual aspects of a parameter configuration, some metrics used to evaluate the system will now be defined:

- **Success** – The mean percentage of initiated downloads that completed successfully and contained non-malicious file content, i.e., $\frac{1}{n}\sum_{i=0}^{n} Success_n$.
- **Increase** – The mean number of files that hosts added to their libraries across the course of the whole simulation, i.e., $\frac{1}{n}\sum_{i=0}^{n} Increase_n$.
- **Efficiency** – After each simulation, each host will have spent X bits downloading non-malicious files successfully, and Y bits downloading either incomplete files, or malicious files. The *Efficiency* metric is thus defined as the mean of each host's $\frac{X}{X+Y}$.

The metric of success is not the same as efficiency: success is based on a binary value of whether a transfer succeeded, whereas efficiency measures how much useful work was performed. If an initiated download is destined to fail, it would be preferable if it failed sooner rather than later. This way less time will be spent on a fruitless attempt, freeing the hosts involved to pursue possibly useful downloads.

6.3 Colocation Prediction

The results from the ideas presented in Chapter 3 will now be depicted and discussed. Initially we focus on simply achieving complete data transfers, without adding the additional complication of file matching. Consequently, no specific files are modelled, and it is assumed everyone has a file that someone else wants. This facilitates testing how well the *Chooser* performs without the complication of the *Matcher*. Hosts are also all non-malicious unless stated otherwise.

6.3.1 Transfer Success

To examine how accurate the peer selection mechanisms are, Figure 6.2 shows how the transfer success rate changes in response to varying network speeds for the TfL traces. As would be expected, all source peer selection approaches perform better as the network speed increases and transfers complete quicker; so less transfers fail from early disconnections. The naive *Random* method acts as a baseline for any other considered approach, it performs extremely poorly at low speeds, yet can still lead to a complete transfer most of the time once speeds exceed 1Mbit/s. When only using the slotted mean (Slotted-0) to order possible neighbours, success is not improved, even using the oracle (Oracle-0) to only order neighbours does not give a significant improvement. There is a marked difference between the considered approaches that utilise a download threshold and ones that do not. In fact, when employing a download threshold (*Threshold*) of 1, a large majority of initiated downloads succeed, offering an obvious benefit for hosts that do not want to spend their battery power and network bandwidth performing useless work. The slotted mean approach (Slotted-1), while not performing as well as the oracle (Oracle-1), still achieves within 10% of successes across all tested speeds. In terms of transfer success, it can be seen that being discerning in which downloads are attempted can yield a near total reduction in disconnections. Furthermore, our slotted prediction approach when a threshold is also employed, is comparable to the Oracle-1 approach.

The effect on transfer success rate when varying the size of files being shared is demonstrated in Figure 6.3. Note that the canonical simulation parameters are being used, apart from *file size*. A separation can again be seen, where employing a download threshold gives a huge benefit to the success rate. All approaches reduce in their success percentage as files become larger and less likely to complete in a given colocation. There is obviously a close relationship between the parameters file size and transfer speed, and the resulting transfer duration. However, the file size is constant throughout the transfer and is known at the beginning, whereas the transfer speed will change during the transfer. Small files are nearly always transferred, no matter the peer selection technique, as they take such a short time to complete. However, as

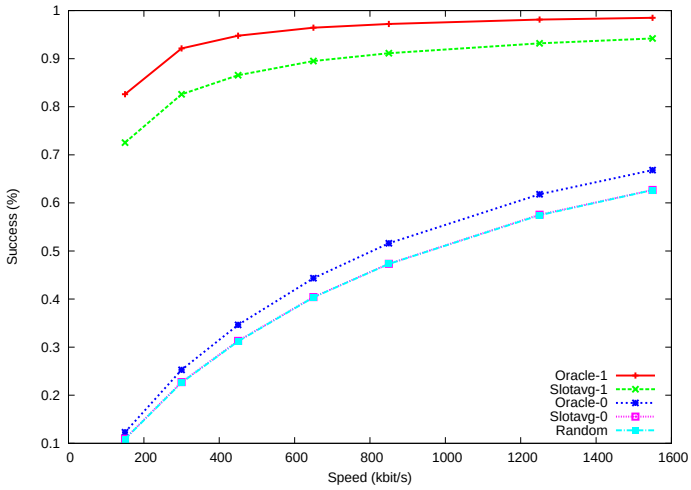

Figure 6.2: TfL Transfer Success vs Speed.

they increase in size, there is a marked divergence between the thresholded approaches (Oracle-1 and Slotted-1) and ones that employ no such judgement (Random, Oracle-0 and Slotted-0). Thresholded approaches maintain a high success rate (always over 70%), with a very low negative gradient as file size increases. The non-thresholded behaviour suffers significantly, with more than half attempts failing when files are larger than 3MB (a likely size for short/low quality music files). Thresholded approaches are able to achieve similar success rates at a wide range of transfer speeds and file sizes, a useful property for a flexible system.

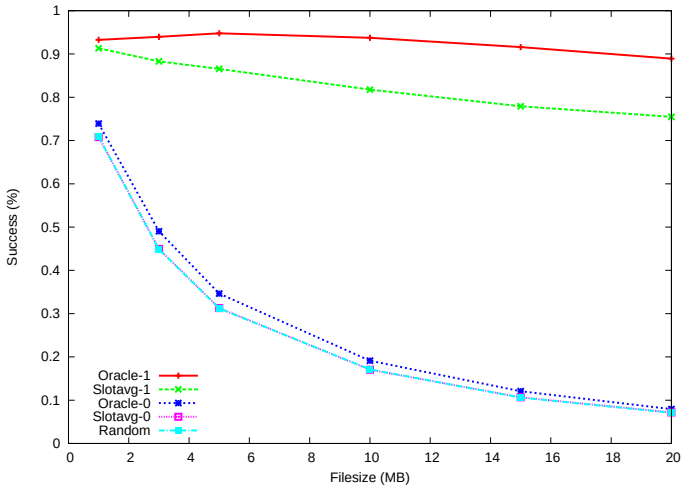

Figure 6.3: TfL Transfer Success vs File size.

6.3.2 Library Increase

The user-centric metric of successfully downloaded files is compared against network speed and file size parameters in Figure 6.4 and Figure 6.5, respectively. All approaches gather more files as the transfer speed increases and less as the file size increases. Both thresholded methods collect more files than non-thresholded, with Oracle-1 maintaining the best performance, a curious result if it is considered that the thresholded approaches are refusing to attempt some potential downloads. This is due to the reduction in contention that is caused by additional downloads, that are unlikely to complete, not being initiated. Also, the data that is transferred is much more likely to be *useful* to the recipient, rather than as part of transfers that will not complete. Hence, the self-limiting behaviour of thresholded peers actually causes an increase in system (and peer) file throughput, similar, in principle, to *congestion control*. It also allows hosts to wait for a promising source, rather than waste their potential downloading time on low duration contacts. Despite Oracle-0 achieving a higher success rate than Random and Slotted-0, it still procures loosely as many files as them. For the canonical configuration, thresholded approaches gain 35 files with 80% success and non-thresholded gain 20 files with 30% success rate.

When considering the impact of file size, it can be seen that for very small files the non-threshold methods just outperform the thresholded. When files take such a short period to complete, not attempting every possible download leads to missed opportunities for file collection. The effect of increasing file size has a large detrimental effect upon all methods' collection of files. For all approaches at particularly large file sizes (around 20MB), less than 20 files are on average gained by hosts while in the system.

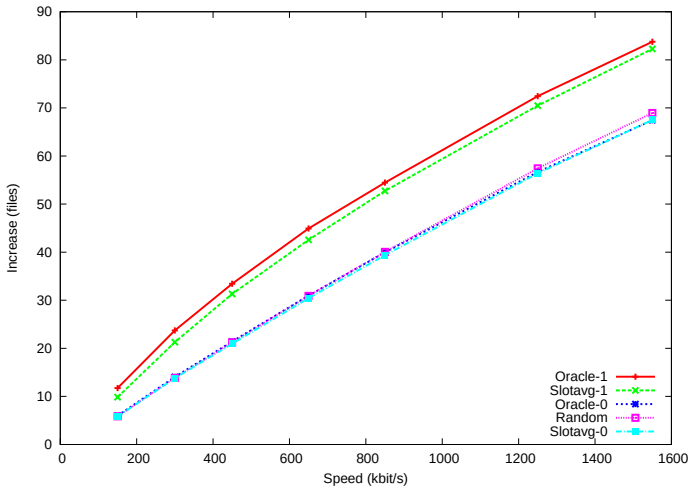

Figure 6.4: TfL File Increase vs Speed.

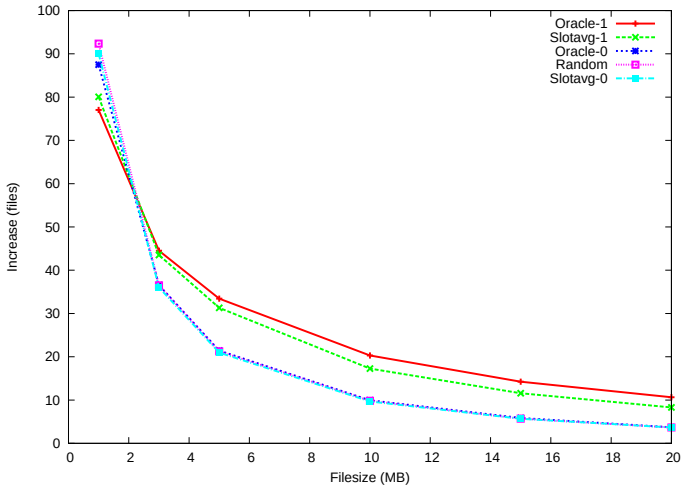

Figure 6.5: TfL File Increase vs File size.

Other Traces

The transfer success percentage and library increase metrics will now be presented for tracesets other than TfL, to give an indication of how the system behaves in different environments. Similar overall behaviour is observed for all traces (Figures 6.6, 6.8 and 6.10 for Reality, Unitrans and Haggle respectively), as speed increases/file size decreases the success rate and file collection increases. Oracle-1 is able to perform near 100% download success for all of these traces: unlike TfL the devices are not as densely packed as on a train, and so transfer duration is predicted more accurately. Slotted-1 is able to achieve the next highest success rate for the Reality traceset, yet under performs for the other two. Their best observed library increase is still provided by Oracle-1 for all traces, however Slotted-1 does not improve over Random, unlike with TfL. This indicates the long period of the Reality traceset facilitates the learning of colocation patterns and consequently reduces the disconnection rate. In fact, Slotted-1 performs worst for Unitrans and Haggle, with all the non-thresholded approaches being similar to Random. Slotted-1 also scales worse than the other approaches when considering file increase (Figures 6.7, 6.9 and 6.11), its gradient is less, diverging from the rest. While all others (after Oracle-0's initial divergence) maintain a similar absolute increase in files received for each increase in transfer speed.

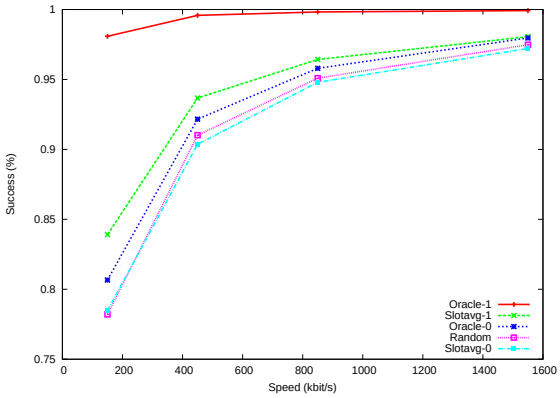

Figure 6.6: Reality Success vs Speed.

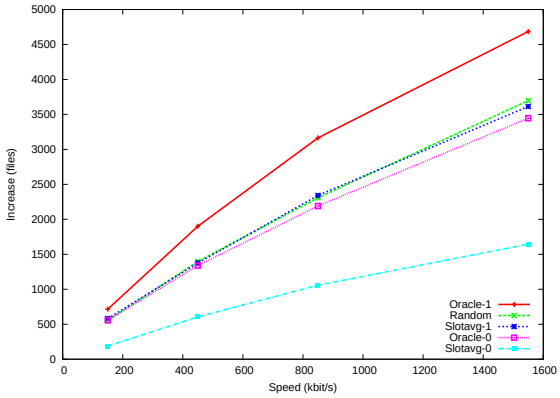

Figure 6.7: Reality Increase vs Speed.

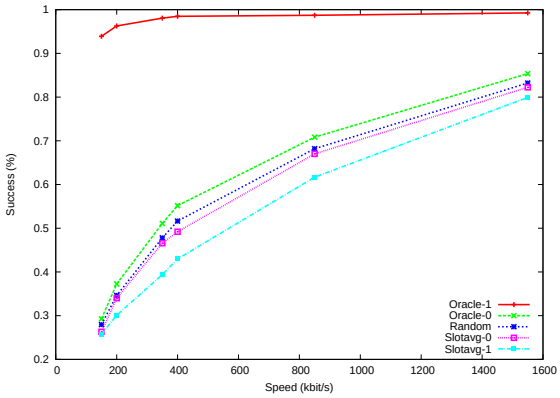

Figure 6.8: Unitrans Success vs Speed.

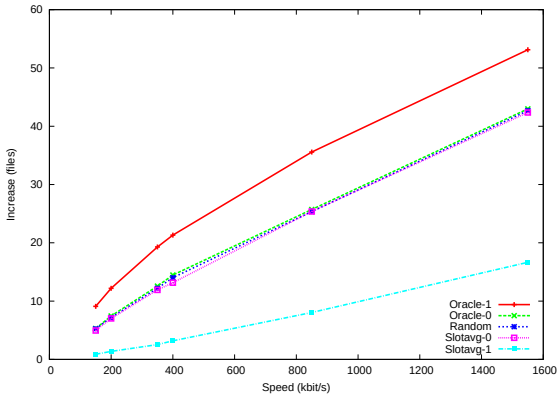

Figure 6.9: Unitrans Increase vs Speed.

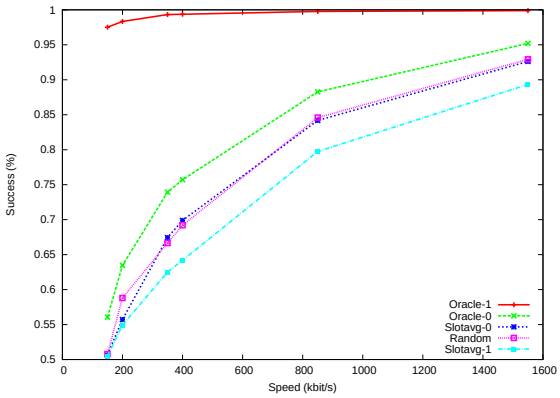

Figure 6.10: Haggle Success vs Speed.

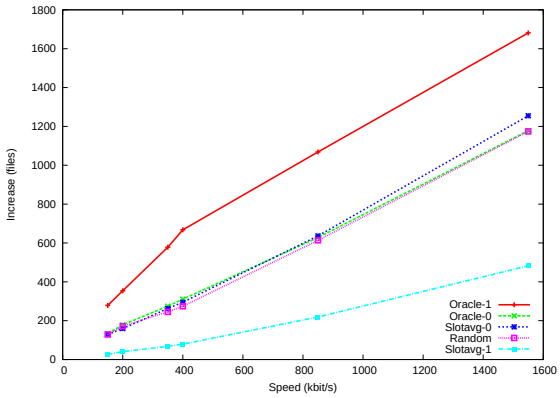

Figure 6.11: Haggle Increase vs Speed.

6.3.3 Efficiency

The network transfer efficiency for TfL is portrayed in Figure 6.12 and Figure 6.13 for speed and file size, respectively. The effect upon efficiency from the underlying success rate leads to similar behaviour seen in Section 6.3.1 between selection methods. Oracle-1 and Slotted-1 are both the most efficient approaches, with all mechanisms approaching perfect efficiency as the system tends to infinite network speed or zero sized files and maintaining 90% efficiency or more normally. Non-thresholded approaches waste around 40% of all their communication overhead when considering the canonical example, this becomes even worse if file sizes increase or network speeds fall.

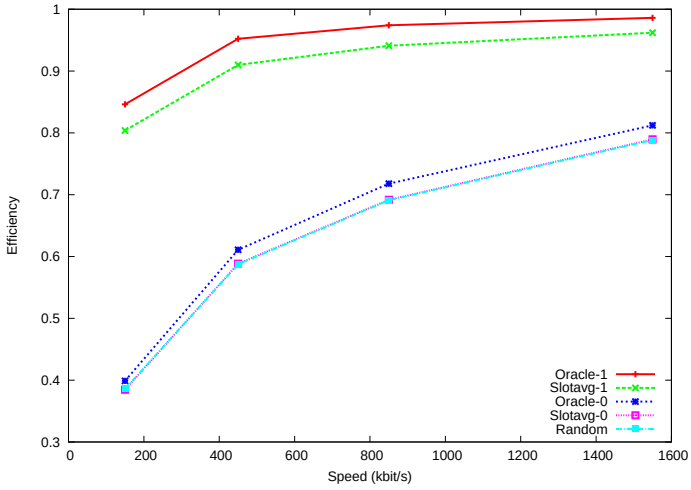

Figure 6.12: TfL Efficiency vs Speed.

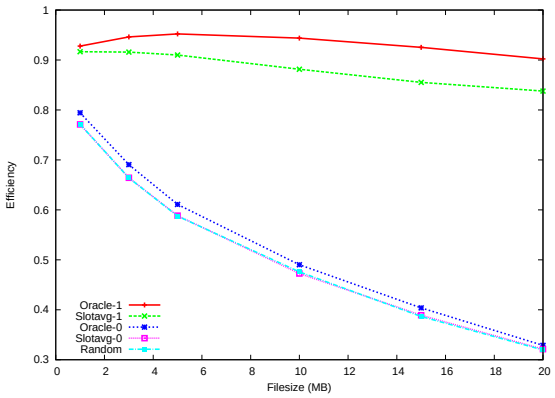

Figure 6.13: TfL Efficiency vs File size.

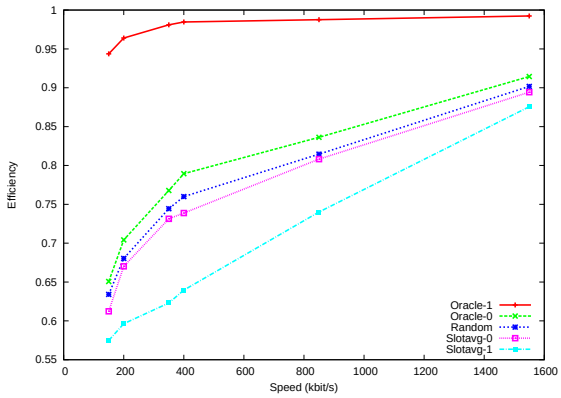

Figure 6.14: Unitrans Efficiency vs Speed.

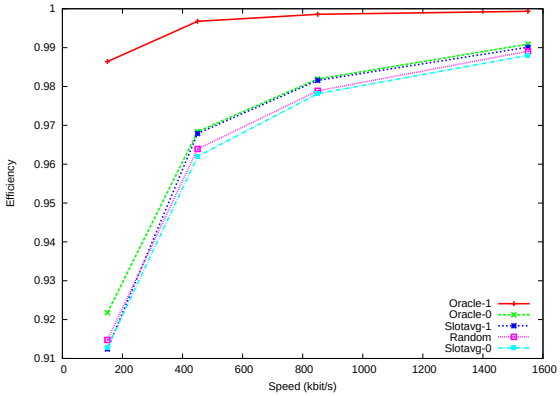

Figure 6.15: Reality Efficiency vs Speed.

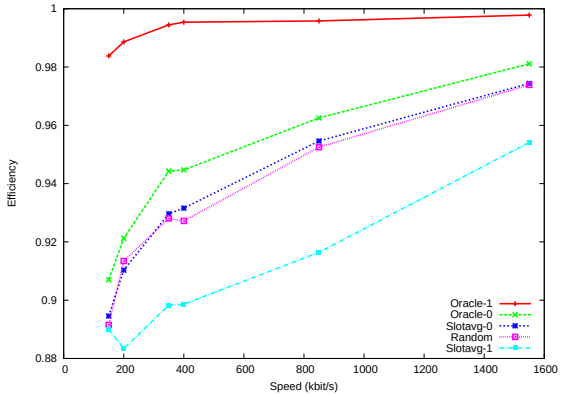

Figure 6.16: Haggle Efficiency vs Speed.

Other Traces

When considering the Reality Mining traces, all approaches (apart from Oracle-1) are very efficient (within 90%-100%) and perform very similarly. Both Unitrans and Haggle display a benefit from using Oracle, even without a threshold, and significant penalty from Slotted-1. Both these traces also span a greater range of potential efficiency, canonical Unitrans experiments (except Oracle-1) are all below 75%.

The differing nature of the trace files is apparent in all these results. This obviously stems from the variety of the colocation properties, e.g. mean duration, variance and regularity. The reality mining trace shows the highest amount of transfer success across all speeds, likely due to its very long durations, such as overnight ones, which do not occur in the other traces. The relatively small set of users also enables easier learning of other user's patterns. The TfL data shows the largest benefit from employing a threshold on the initiation of transfers, indicating that being careful when starting transfers was particularly important in this environment. This was because of the high density of the TfL traces, hence multiple overlapping transfers had a larger negative impact on system success rates. The comparatively poor behaviour of the Unitrans traces could be partially explained by the manner of the trace collection. Bluetooth scanners are fixed on buses and thus may not be making journeys at the same time, and thus fail to capture user regularity well. These observations show that the source prediction technique's success will depend on the environment they are used in. It can be extrapolated that environments with short colocations and dense user placement will benefit the most from the prediction step. The approach does at least outperform Random in all traces, which indicates that despite variance of tested environments it is always beneficial to use colocation prediction.

6.4 Content Selection

The advertising techniques that were described in Chapter 4, and their effect on the operation of our proposed system, will now be presented. The modelling of individual files is included in all following results, to allow the comparison of advertising strategies. The success rate and efficiency are both almost entirely unaffected by the advertising policy and are therefore also not presented in this section.

File advertising methods when used in conjunction with each peer chooser on the TfL traces are shown in Figure 6.17, Figure 6.18 and Figure 6.19. It can clearly be seen that both *Popular* and *Unpopular* matching techniques have a negative impact on the file increase metric, performing much worse than all other matchers. The best matching approaches are *Uncommon* and *Random* for all peer choosers, they have both perform equally well in terms of file increase. *Common* follows a similar trend at low speeds, but diverges when the network speed increases.

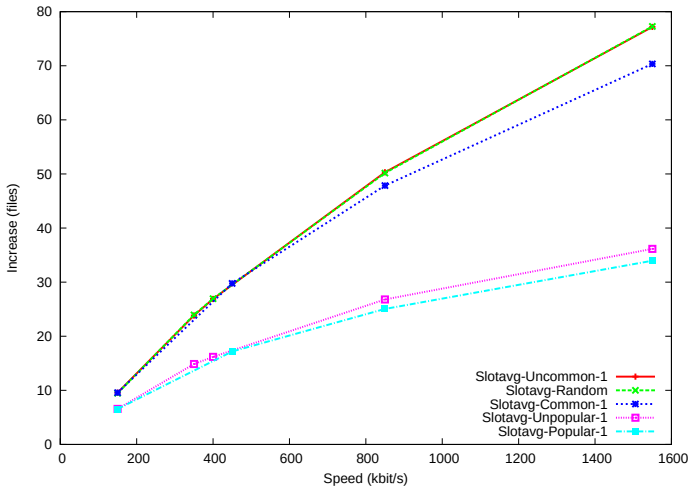

Figure 6.17: TfL Slotted Matcher Increase Speed.

Despite *Uncommon* and *Random* performing similarly when measuring file increase, they do perform differently when considering which actual files are replicated in the system. The impact of different advertising methods upon the replication frequency of individual tracks is shown in Figure 6.20. The X axis is the ranked list of tracks arranged by the amount of replication performed by the system, which is represented on the Y axis. The approach that replicates the least number of different files (but replicates them many times) is *Common*, which is intuitively understandable if only the files with many initial copies are replicated. This would occur due to hosts initially sharing commonly seen files, causing positive feedback on the files so they are replicated in the system as a whole. Random advertising of the tracks gives the middling result of the approaches, replicating many files, but still giving particular focus to

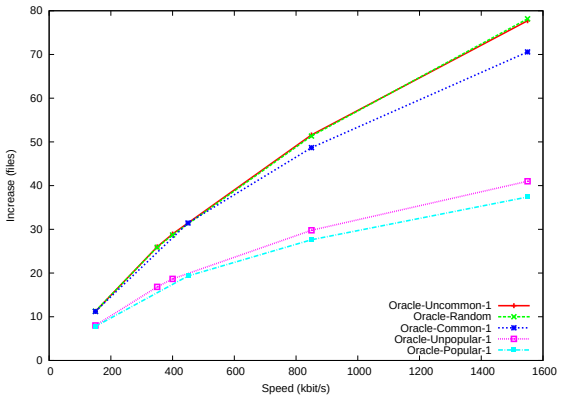

Figure 6.18: Oracle Matcher Increase Speed.

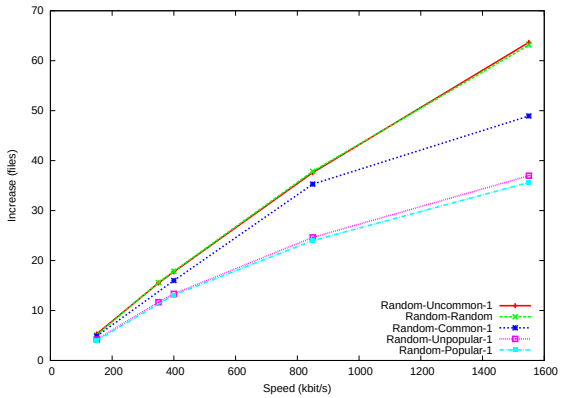

Figure 6.19: Random Matcher Increase Speed.

some. This is due to the files that initially exist on many hosts being more likely to be selected for replication by one of the owners, even though each owner is selecting randomly. This graph is comparable to the model's genre distribution graphs, presented in Section 7.4.

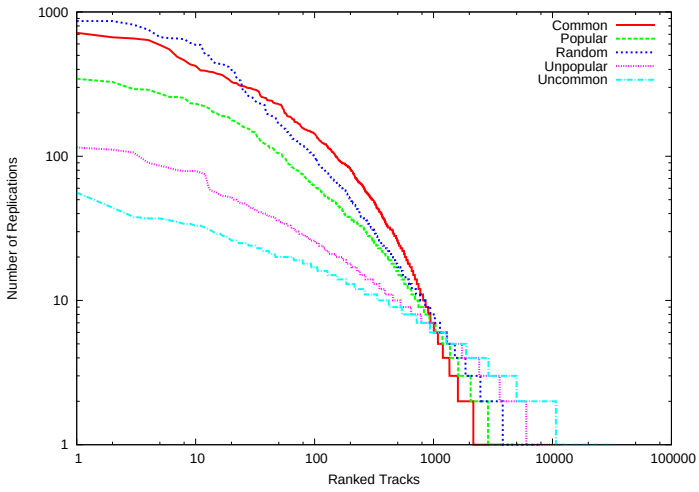

Figure 6.20: Track Replication.

Figure 6.21 shows progression of the total amount of files in the system over time, with normal configuration. Distinctive bursts of increase can clearly be seen, which correspond to the peak travels times, identified in Section 5.1.3, equivalently flat areas correspond to weekends.

Other Traces

The effect of file advertisement is particularly different for Reality Mining (Figure 6.23) as the trace lasts for so long. Uncommon matching outperforms all other approaches, even when the threshold is disabled. Unpopular is shown to be of poor quality, still performing significantly less library increases than Random. Uncommon and Random perform similarly on the

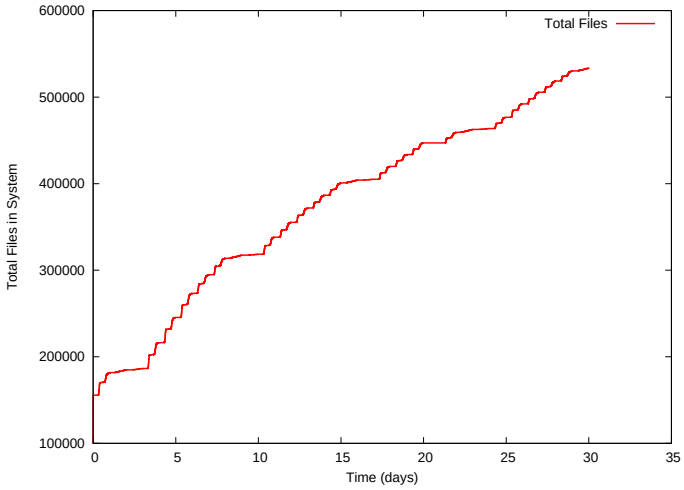

Figure 6.21: TfL File Increase.

Haggle traces Figure 6.24, and again Unpopular is the worst choice. Random outperforms all approaches on Unitrans, though Uncommon closely trails it (Figure 6.22).

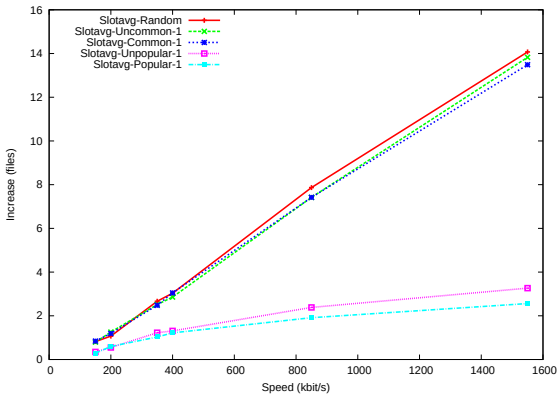

Figure 6.22: Unitrans Matching Increase.

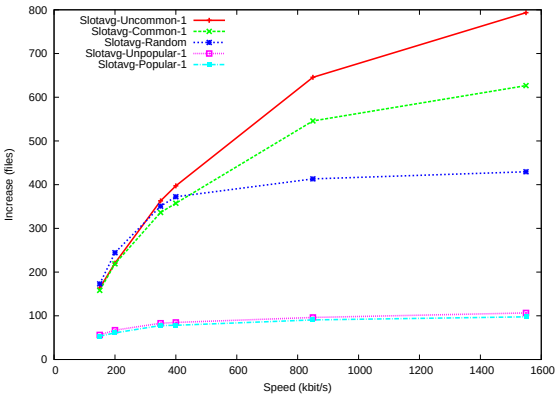

Figure 6.23: Reality Matching Increase.

File Receipt Behaviour

A graphical representation of how a single particular track (and each of its copies) spreads through the TfL network can be seen in Figure 6.25. The file size was set particularly small for this simulation run (0.1MB), to allow greater spreading and facilitate analysis of the connection network. At the beginning of the simulation there were three versions of the file, belonging to a popular genre (rock), which spread to 1837 other hosts. The three trees formed from the three initial files (root nodes) are each very different. The small one being a three link chain, the medium one being a relatively balanced tree and a third large tree that dominates the replications. It can be seen that the file in the large tree spread to many hosts, often through

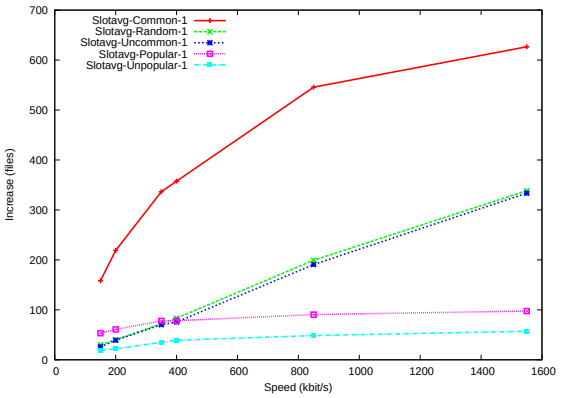

Figure 6.24: Haggle Matching Increase.

intermediaries that gain the file elsewhere. This shows how even with hosts only focusing on single hop dissemination, large complex emergent flows can still be created; with tracks spreading along paths of common interest.

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 6.26 shows the effect on the genre occurrence distribution in the network from using our system. The 'Initial distribution' and 'Gained distribution' are combined into 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 ones become so widely available that they are not looked for as much as at the beginning of the simulation. Note also how the final distribution still has a heavy tail: we are thus not radically 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.

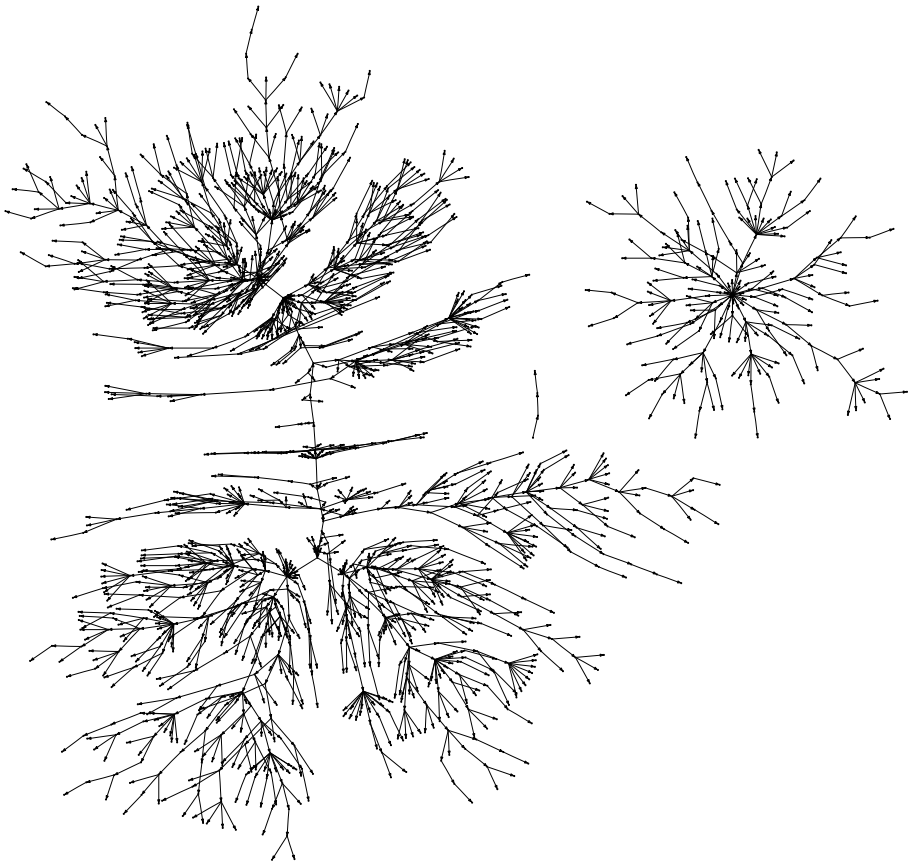

Figure 6.25: Individual Track Dissemination.

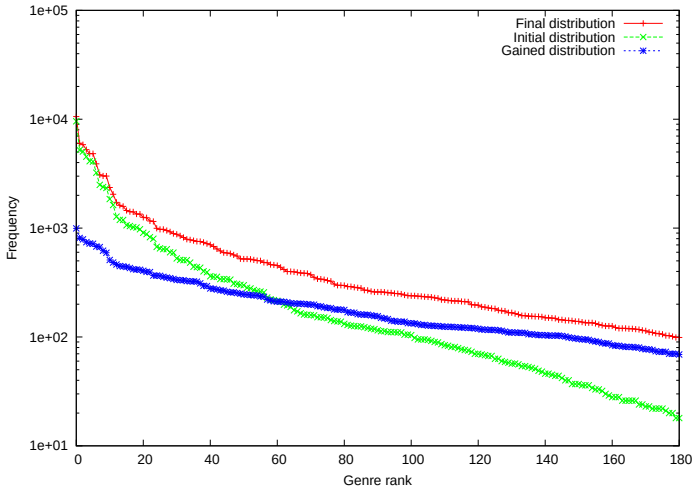

Figure 6.26: TfL Genre Distribution.

6.4.1 Temporal Behaviour

The number of files a user has available to share will increase throughout the lifetime of the system, as more and more is downloaded from others. The path that a file follows through the system will be of varying length, with a length of 0 indicating that the owner started the simulation with the file in their library, and a length of 10 indicating that the file was previously handled by 9 intermediaries before being downloaded. The graph shown in Figure 6.27 visualises the distribution of a file's path length at the end of the canonical simulation run, when using Slotted-1 chooser, comparing the Uncommon and Random matchers. It should be noted that files are counted multiple times, as a file shared twice will register a length of 0 for the initial owner, a 1 for the second and a 2 for the third. The longest replication path that any initial file achieved was 31, by only one file. The Uncommon matcher causes long path lengths to occur more often, and short ones much less. This shows how Uncommon is having peers that a gain a file to redistribute it, rather than allowing the initial owner to be the spreader of its data.

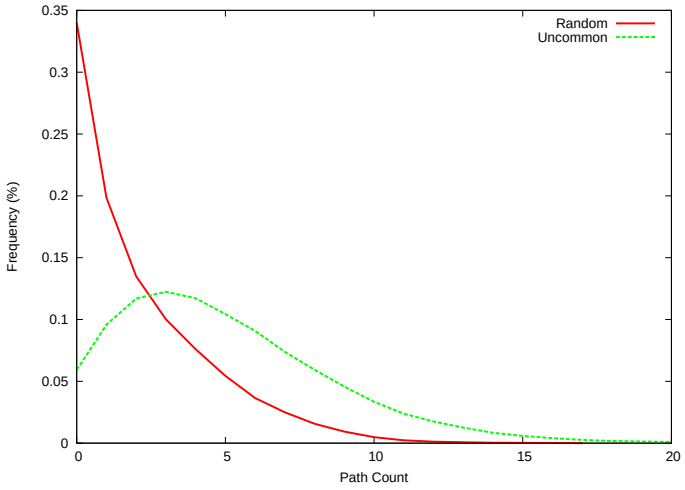

Figure 6.27: Frequency of Path Length.

6.5 Trust

In all previous experiments, hosts were assumed to be non-malicious. However, to analyse the impact of the presence of malicious/faulty nodes in the opportunistic network, nodes which exhibit non-cooperative behaviour must be introduced. When receiving data from a malicious host, the transfer completes as normal; it is only after receipt of the complete file that recipient realises that the data is malicious. The interaction outcomes are recorded in a trust repository, which is then consulted as part of the source selection algorithm. The canonical parameter values for $MalRate$ (proportion of malicious hosts) and $MalChance$ (chance a malicious host

Setup	$MalRate$	Good Files	Bad Files	Useful (MB)	Wasted (MB)
Naive	0	31.3	0	155.3	15.3
Beta	0	30.8	0	147.6	14.9
Naive	0.3	20.7	9.2	102.8	60.3
Beta	0.3	21.2	7.8	105.1	53.3

Table 6.3: Malicious Effects.

misbehaves) are now set to 0.3 and 1, respectively. Having 30% of all nodes being malicious, represents a large proportion for the scenario we are considering.

When reasoning about trust, a major restriction of the actual user traces is their limited duration: a matter of days is not enough time to build a reliable trust repository. In the Unitrans traces, hosts only meet a mean of 5.9 times, and with the Haggle traces just 3.7. Note that this is not the average number of times a pair of hosts interacts, but purely the number of times they have met, so the actual chances of building an accurate trust valuation from completed downloads is very low. The TfL data lasts for a more reasonable length of time, but the large size of the network and large number of complete strangers also makes it challenging. The Reality Mining traces are ideal for testing the potential of a trust system as it is very long, and a tight-knit community.

To simulate the behaviour of malicious hosts the parameter $MalRate$ is used, to change the proportion of hosts that act malicious. The impact on the quality of received files from the proportion of constantly malicious hosts is explored. Though not presented here, to avoid verbosity, if malicious hosts only misbehave probabilistically ($MalChance < 1$), then it will be harder to learn which perform poorly in aggregate. This allows examination of both simplistically malicious hosts (such as advertising shills) that misbehave every time and either probabilistically malicious devices or normal users with mistagged libraries. The ability to avoid bad sources that sometimes provide good interactions is important for a trust system, so they are not fooled by hosts that occasionally masquerade as good, but are detrimental to the system overall. When experimentally tested, as would be expected, if malicious hosts only provide malicious files 50% of the time, then all hosts in the system receive less malicious files; however the proportional improvement of using a trust system is reduced.

The performance when using the trust system (Beta) or not (Naive) in the presence of both malicious users and non-malicious users is shown in Table 6.3. It should be noted that *Wasted* bandwidth also includes data from incomplete transfers as well as malicious transfers. It can be

seen that when no hosts are behaving maliciously ($MalRate$ is 0) both Naive and Beta receive no malicious files, while gaining a number of good files similar to each other. However, when 0.3 of all nodes are malicious, Beta only received 85% of the malicious files that Naive does, a figure that is likely to grow as the system operates and hosts learn the good sources from the malicious ones.

The graphs in this section present a slightly modified version of 'successful transfer'; we are only considering the ratio of non-malicious received files, thus only analysing the ability to avoid malicious hosts, and not ensure complete transfers. The effect on the ratio of non-malicious files collected when varying the proportion of malicious files is shown, with curves employing the trust reasoning appended with 'Beta'. Only the Oracle and Slotavg choosers are presented, with both configurations using the Uncommon file matcher. When applied to the TfL traces (Figure 6.28) the use of a trust system has a good improvement, which grows as the rate of malicious hosts grows. When 30% of hosts behave maliciously, the trust reasoning reduces malicious file collection by nearly 8%. The benefit for Reality Mining is marked (Figure 6.29), the trust system enables the hosts to accurately identify malicious hosts extremely effectively. Relatively, barely any malicious files are received, less than 5%, even with 30% malicious nodes. This is due to the long duration of the experiment, all malicious entities can be identified and subsequently avoided. Both Unitrans (Figure 6.30) and Haggle (Figure 6.31) also show a large improvement from the utilisation of trust valuations.

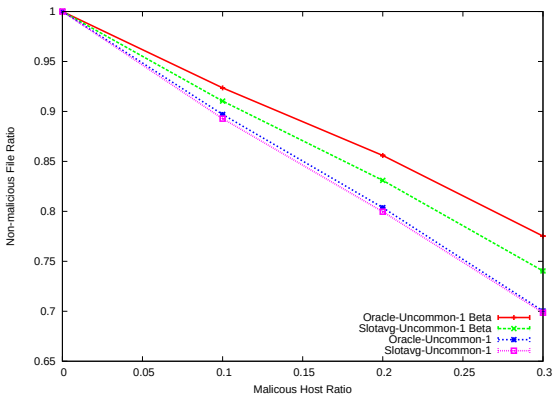

Figure 6.28: TfL Trust Comparison.

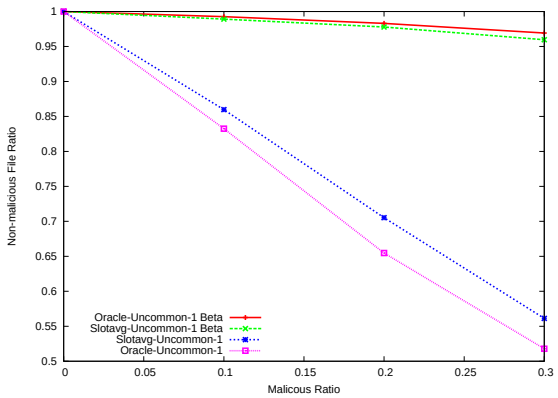

Figure 6.29: Reality Trust Comparison.

Despite not lasting for long periods, as with the Reality Mining traces, both traces have much smaller populations, again leading to the ability to learn who the malicious entities are. Interestingly, the Oracle chooser out performs the Slotavg chooser, despite the fact we are only considering completed transfers. This is caused by the Oracle approach making more stable

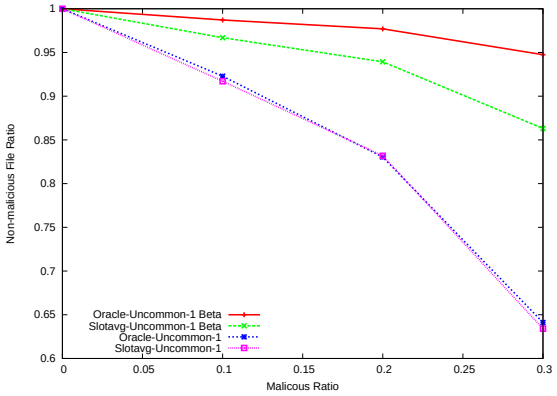

Figure 6.30: Unitrans Trust Comparison.

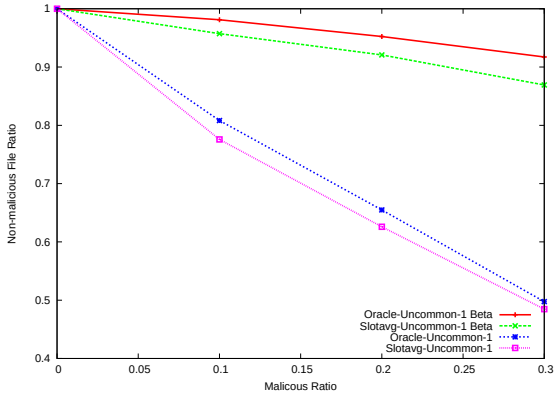

Figure 6.31: Haggle Trust Comparison.

choices, i.e., it selects the same source more often, allowing better learning of other host's behaviour.

The trust values for other hosts evolves over time, becoming more accurate with additional interactions. Therefore, our Beta trust repository should improve over time, reducing the amount of malicious failures as the simulation progresses. This progress can be seen in Figure 6.32, which depicts the percentage of completed transfers in the canonical simulation configuration. A curve is shown for the stateless configuration where no trust values are used, and the Beta trust approach. The lines are not completely smooth, likely due to weekly variation in the amount of new people met. It can be seen that the Beta trust is gradually improving throughout the experiment, starting with just below 29% failure rate and ending the simulation with a 25% failure rate.

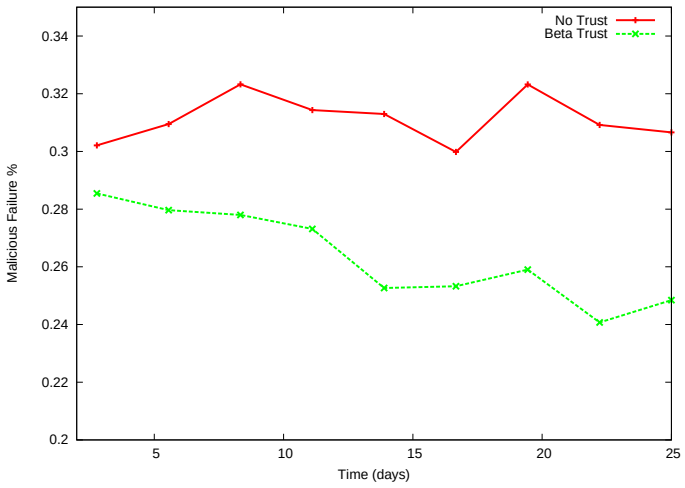

Figure 6.32: Evolution of Malicious Failures.

6.6 Summary

This chapter presented two methods of investigating our proposed system, a prototype implementation and simulation of the content distribution system. First we described the application that was developed to examine the feasibility of operating a wireless content distribution system. It was tested on relatively modern devices in the envisaged scenario environment of a rapid transport system, both off-peak and on-peak. Despite the huge storage possibilities, the limitations of the devices still required extra consideration; they could not be treated as simply small desktops.

Next we presented the findings from the simulations of user movements that were used to demonstrate how our system would perform in the selected environments. Results were illustrated from our simulation experiments analysing the component features of our proposed system. The impact on file transfer success and efficiency from the source selection procedure, as described in Chapter 3, was discussed first, followed by the file advertising techniques (from Chapter 4) and how that modifies which files are successfully replicated by the content distribution system. We saw how disconnections can be avoided using informed source selection. Careful selection of which files are downloaded could also have an important effect on the behaviour of the system, and how it replicates files of different popularities. Finally, the benefits of using a trust system to avoid malicious hosts in untrusted environments was presented. It was apparent that experiments that lasted for long periods of time or involved a small population could gain very large reductions in the amount of malicious files received. Even small scale or short simulations also showed some benefit, an improvement that seemed to increase over time, as hosts learnt about others behaviour. Some analysis of how the network of spreading files changes, the routes files take through the network and how the system performs over time was also briefly discussed.

7

Content Spreading Analysis and Evaluation

The main evaluation aspect of this thesis focuses on using experimentally collected data (e.g. human movement traces and music website crawls) to study the feasibility of our proposed method of content sharing. In addition this chapter offers a concise conceptual representation of the content spreading process, to facilitate simplified analysis of the effects of sharing data items. This chapter discusses how the pairwise content selections that occur in a peer-to-peer system can have large effects upon the system as a whole. Specifically, we evaluate and discuss how the choice of which data items to share can drastically change the availability and distribution of the files in an idealised system, particularly when no source selection process is used. Policies that have an analogue to ones presented in Section 4.3 have their effects investigated in idealised networks.

The rest of the chapter is arranged as follows: Section 7.1 presents some related research in modelling data exchange networks, Section 7.2 lists some simplifying assumptions that were made for our model, which is defined in Section 7.3. The results of this modelling is then presented Section 7.4, followed by some conclusions that can be drawn, Section 7.4.3.

7.1 Related Work

Modelling a network of states has been pursued for many years in fields as varied as ferromagnetism to anthropology. Though it may seem natural in opportunistic network research, when considering the spreading of data items, to employ epidemic modelling techniques, as much previous work does [KMM34, ZAZ04]. This formulation of the data spreading problem is not actually suitable for the scenario we are considering. Epidemic modelling is usually concerned with the spread of one entity through a population, whereas we are considering the distribution of many. If our system were viewed as many epidemic processes superimposed on top of each other, it would miss the very important limitation of bandwidth. If a node has an opportunity to interact with another, the choice of the type of interaction they perform has an impact on the spreading of all others, i.e. if the pair share data item j during one period, it means they do not share i during that same period. In fact, the usage of many superimposed epidemic networks would imply infinite bandwidth, and so the lack of one type of interaction would not impact whether another type was performed.

Delay Tolerant Network (DTNs) research often implicitly assume an infinite bandwidth between peers. The DTN field focuses on the potential spreading and reachability of messages. More precisely, which intermediate nodes should be handed a message in order for it to eventually reach its intended destination. This is useful when a host wishes to spread a data item to a certain subset of the population, but is less useful when a host simply wants to pull certain categories of data from its neighbours. Sophisticated models have been used to model the average age of content updates being collaboratively distributed across a network [ICM09]. They rarely include finite bandwidth [ILY07] and multiple different items being shared in the network.

A slightly more suitable formulation of sharing state over a network has been considered in the field of cultural studies. The application of complex network theory to social developments was performed by Axelrod [Axe97]. It can be very useful for analysing how culture spreads through a social network along relationships between people. It can also help identify structurally important people [BKA09] or their influence on a network of friends [ORT09]. Axelrod's work, rather than concerning itself with the particulars of a culture or how people of a particular culture interact with another, models the way people exchange culture with their neighbours. The focus being what processes cause convergence/polarisation of cultural values between sets of people. Nodes in the system represent statically connected villages. Each village has a vector of cultural *features*, which could represent language spoken, food eaten or architectural styles. Each node's language feature would be set to a particular value or 'trait', such as *English* or *Spanish*. Over time nodes affect each other's cultural features, and so in-

fluencing a neighbour to adopt one's own trait. It was discovered that even with relatively simple rules local convergence can lead to global polarisation, a behaviour observed in real life. The static connections on the network are not a requirement of this type of model. Additional work [PH06] describes models where network connections and node qualities are dynamic. Attempting to explain how new connections can be influenced by a host's attributes as well as how the attributes can equally be modified by the connected hosts.

Axelrod's work was further extended with the description of Networked Urn Processes [CCH+08]. Urn processes have each node represented by an urn filled with multiple coloured balls (e.g. multiple genre files). An urn can then select one of its neighbour's balls to duplicate into its own urn, and thus gain a ball of the selected colour. This models data sharing better than Axelrod, as balls are never replaced or removed only duplicated. These cultural models are created with a view to informing us about what processes may be occurring with real people. The analysis in [CCH+08] shows how social influence and selection interact differently in different scenarios, specifically Wikipedia and LiveJournal[1]. The justification for our model, is just to understand a similar system, rather than replicate the processes of an existing one. We are not trying to make its behaviour match reality, we will input some real data, but the model is purely used as a tool, not a realisation.

The scenario laid out in previous work differs from this body of work in a number of significant ways. Nodes do not have their tastes changed by neighbours, they simply add transfers to their library of items. Hence 'culture' can not be destroyed, however categories could be neglected. The connectivity network is a much more dynamic arrangement, changing many times over the course of one day. We require a model that uses many interconnected elements, able to share data items between each other. Each element needs to be interested in multiple categories from a population of data categories. They should be able to consecutively acquire items of these categories from their neighbouring elements. This formulation is flexible and generic, yet still captures enough features of our scenario to behave similarly and highlight some considerations for our system.

7.2 Assumptions

Many assumptions have been made for purposes of analysis and to achieve a generalised approach.

- No network effects are considered, there is a perfect transmission environment, with no signal attenuation or contention. Downloads take a constant time and are not affected by

[1] Website URLs – Wikipedia: http://wikpedia.org and http://livejournal.com

V	Set of nodes in the system: $V = \{u_1, u_2, ..., u_n\}$.
G	Set of possible interests that a node may possess: $G = \{g_1, g_2, ..., g_e\}$.
$Pr(g)$	The probability of selecting the genre g out of the population.
$Int(u_i)$	The subset of interests that a user has: $Int(u_i) \subseteq I$. $h_i = \|Int(u_i)\|$
$Libsize_g(u_i)$	Size of user's library genre: $libsize_g(u_i) = \|Q\|, Q \subseteq D, \forall q \in lib(u_i)$.
$Libsize_g$	Size of the system's collection of a certain genre: $Libsize_g = \sum_{i=0}^{n} Libsize_g(u_i)$

Table 7.1: Notation and Terms.

other transmissions happening in the surrounding environment.

- Users have an immutable set of interests $Int(u_i)$ that they desire over the course of the whole experiment. The likelihood a user enjoying a genre is entirely independent of the probability that they like another genre. Note, this loses covariance of occurrence between genres.

- The likelihood of a user liking a single music genre is Zipf distributed, and the probability of two genres of interest are not specifically covariant events, i.e., if a user likes classic rock it is no more probable that they will enjoy stadium rock. However, if user has two genres of interest they will not be the same.

- For the modelling of this chapter, we will assume that there is no source selection process, involving colocation prediction (such as in Chapter 3), neighbours uniformly randomly select the peer they will download from. This is a simplification to ease the reasoning content spreading without being affected by the colocation prediction.

7.3 Model Definition

To aid in the description of our model and its behaviour, some notation is presented in Table 7.1. Similar to the previously described research [Axe97], a simple regular lattice arrangement of nodes will be employed to provide a starting point for further analysis. This allows the model to be examined on a basic network, without any of the properties of real wireless networks, such as movement, clustering and regularity. The effects of the data item selection phase can thus be seen without complicating factors. As depicted in the Figure 7.1, there is a two dimensional grid where each cell represents a node in the system. Each node has four neighbours, hence the center node in the diagram is neighbouring with the four shaded nodes. Nodes on the edge and corners have three and two neighbours, respectively. The model progresses in time along sequential *ticks*, which can simply be thought of as seconds. Each tick of time, a node that is

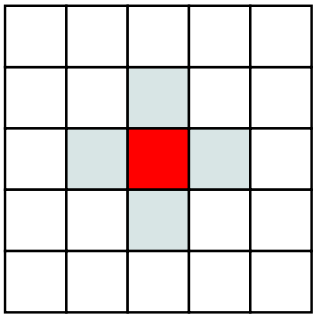

Figure 7.1: Simple lattice of nodes.

not currently downloading may randomly select one of its neighbours to act as a source for a download. The order that hosts choose their neighbours is always sequentially from a corner row by row, the different random seed for each run will remove the effect of any structural deadlocks. Only neighbours that are *available*, and thus not already acting as sources may be selected. When the trace files from Chapter 5 are employed, if a source loses connectivity with the downloader during the course of the transfer, the download is stopped and a new source selected next tick.

Content Distribution

If we are assuming that a genre g_n of rank n has a Zipf distributed likelihood of occurrence, then its frequency can be characterised as: $Pr(g_n) \sim 1/n^a$. The larger the exponent a, the more skewed the distribution is to the top ranked genres. This leads to the probability mass function having a linear plot on a log-log graph. This follows from the analysis presented in Section 5.2.2, where genre and artist had their distribution of popularity discussed.

If each host in the system chooses a single genre of interest, then the resulting total population of genres in the system will also be Zipf distributed. If each host selects a subset of the available genres, the resulting total population of genres will not be Zipf distributed. Consider a given host u_i, with h_i non-repeating genres of interests (out of all possible G). The probability the host is interested in a specific genre g_x is not simply $Pr(g_x).h_i$. Due to the lack of replacement in the selection process the probability of choosing g_x increases each time it is not chosen (as other possibilities are removed). For example, if a host u_i has to select 2 genres from a population of 3 i.e., $G = \{g1, g2, g3\}$ and $h_i = 2$. We are interested in whether they select *g1* i.e., if $g1 \in Int(u_i)$. Each genre has an associated frequency of occurrence, or equivalently associated probability of individual selection $\Pr(g_x)$. The host has two genres of interest and can either select *g1* first with probability $\Pr(g1)$, or select $g1$ for its second genre.

However the probability of selecting it the second time is not $\Pr(g1)$, as the previous selection has reduced the possible options to only two. The first choice will influence the probabilities for the second choice. If $g2$ was chosen first, then only $g1$ or $g3$ can be chosen, with probability $Pr(g1)/(1 - Pr(g2))$ and $Pr(g3)/(1 - Pr(g2))$, respectively. Hence, the likelihood of choosing $g1$ in the second selection is $Pr(g1)/(1-Pr(g?))$, where $g?$ is the previously chosen genre. To generalise this reasoning to arbitrary sized interest populations and user interest sets, the following equations can be used. Equation 7.1 gives the chance of choose $g1$ for a given interest slot n. Equation 7.2 is just the definition of all the previously made choices ($ProbPrev$), where $Prev$ is the set of all already chosen genres. Equation 7.3 gives the chance of choosing $g1$ when the host has h slots, .

$$Select(g1, n) = \frac{\Pr(g1)}{1 - ProbPrev} \tag{7.1}$$

$$ProbPrev = \sum_{x \in Prev} Pr(g_x) \tag{7.2}$$

$$PrSelect(g1, h) = \sum_{i=0}^{h} Select(g1, i) \tag{7.3}$$

The equations can be represented more naturally as a procedural algorithm show in Algorithm 7.1 and 7.2. In fact this process can not be described well with simple equations as no closed form exists, evidenced by Wallenius' noncentral hypergeometric distribution [Wal63]. Using this algorithm we calculate the expected initial distribution of genres across the network when users each select a subset. Figure 7.2 shows *lines* from 400 runs of the experimental system and the *points* from the analytical calculations. The number of possible genres is 100 and all genres have a Zipf likelihood. Three separate configurations are presented, with the user's genre subset size (h) being set to 1, 5 and 7. The experimental and analytical results match almost perfectly in Figure 7.2. The curve labelled *1 Genre Experiment* shows the relative proportion of genres that are liked in the system, it is a simple Zipf distribution. The other two curves (for 5 user genres and 7) also have a set of points overlaid on them. The curves show the experimental results from using our model and the points show the analytical results using our formula. It appears that as users enjoy more genres the distribution gains a heavier tail, and the popular genres are less popular. Actually, as the size of total interests present in the system increases, the popular genres simply represent a smaller proportion. This is because users with many genres of interest will probably chose a popular genre, but will then be able to choose

more, possibly less popular ones as well. This causes a flattening of the top end of the total system genre distribution.

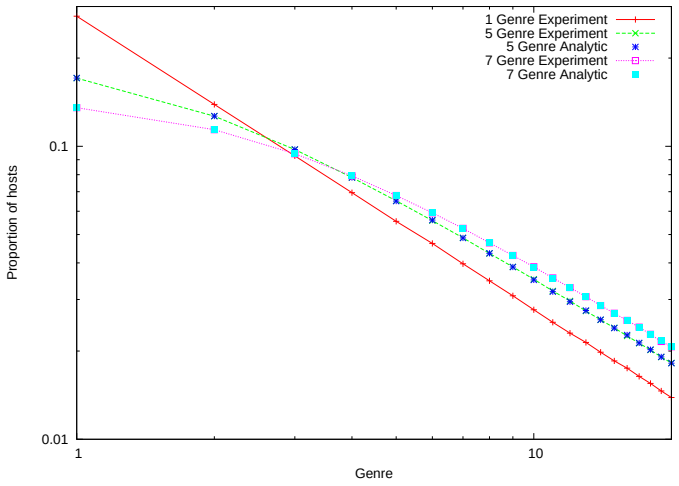

Figure 7.2: Analytic vs Experimental.

result = 0;
hasGenre(gN,g,$Chosen$):
if $g = 0$ **then**
  return
$current$ = random($Genres \setminus Chosen$)
$Chosen$ = $Chosen \cup current$
if $Chosen = gN$ **then**
  result = addProb(Chosen)
else
  hasGenre(gN,$g - 1$,$Chosen$)
return result;

Algorithm 7.1: Likelihood of Genre Occurrence.

To populate the libraries, each host is dealt with independently. Choosing a host's interest genres $Int(u_i)$ is a simple act of selection without replacement. For each genre g selected, the user's library size $LibSize_g(u_i)$ is set as a uniform random variable $[1 : 10]$. The absolute range of initial interest does not fundamentally change how the model performs, it merely impacts the rate with which new transfers affect the global distribution of genre volume.

At each time tick, all non-downloading hosts will choose a random available neighbour, that shares a genre of interest, in order to initiate a download. A download is initiated and the source host is set as *unavailable*. If no neighbours are available, the downloader does nothing. The choice of genre for the transfer is chosen according to the download selection policy that is

addProb($Chosen$):
$acc = 0$
$rv = 1$
for $n \in Chosen$ **do**
  $rv* = \Pr(n)/(1 - acc)$
  $acc+ = \Pr(n)$
return rv

Algorithm 7.2: Calculate Probability.

being used. If the neighbour loses connectivity (only applicable to the trace based experiments) the download is cancelled, and the initiation search is performed again. Upon completion, the host's relevant genre magnitude is incremented.

The connection traces also modify the structure of the network. For the simple grid structure the inter-host connections do not change over the course of the experiment. The trace based experiments all follow the same initialisation procedure, each modelled host is mapped to a trace host. Each line of the trace file is then read and the relevant (dis)connection is applied to the model. Each second in the trace file equates to a single tick in the model.

7.3.1 Advertising Policies

The policies that hosts use when negotiating which genre will be downloaded between peers, will now be presented. Each advertising policy is based on local information during the file negotiation phase, no state is maintained.

Random – The Genre is chosen uniformly randomly between matching genres.

Popular – The matching genre that the source has the most interest in is selected as the genre to share.

Unpopular – The matching genre that the source has the least interest in is selected as the genre to share.

Popular Proportional – The matching genre is chosen with a probability equal to the relative interest the source shows in that genre (compared to all other matching genres).

Unpopular Proportional – The same as above, but with a probability inversely proportional to the relative interest compared to all other matching genres.

These policies are analogues of the mechanisms presented in Section 4.3, except for the difference that they do not maintain state, and operate purely with local knowledge from the source.

Parameter	Symbol	Default Value
Hosts	d	2500*
Experiment Duration	s	10,000*
Download Duration	D	200
Global Genres	G	100
Local Genres	h	5
Zipf Exponent	a	1

Table 7.2: Model Parameter Values.

Although each host has an i.i.d. value for each of its genres of interest, some order will still arise out of the initial conditions. If a genre is more likely to be of interest to the population, then independently of the individual values it will still be more likely to be a possible matching interest between any pair. If there is a Popular based policy being used, then the most popular and shareable files will be shared more, promoting their popularity even further in a positive feedback loop. The Unpopular approaches are trying to always share the currently least populous, and so it approaches a system where all genres are shared by an equal amount. Though this would not count hosts that are isolated from any others with similar tastes.

7.3.2 Parameters

The major parameters that control the behaviour of the model are defined in Table 7.2 together with their default values. The time it takes to perform a download D is set to 200 seconds, equivalent to 5MB files transferred at 200Kbit/s. The file popularity exponent a is set to 1, this is 5% different from the majority of the Last.fm data (ignoring most popular few), but was chosen to be generic, the effect of varying this parameter is investigated. The parameters with a star are dictated by the network structure that is being used. The possible network styles used are Grid, TfL, Reality and Haggle, relating to the idealised grid network and each of the traces presented in Chapter 5. The canonical values are modified for each trace file, so that the number of hosts and length of experiment match the trace file.

The results gathered from a particular trace set are relatively specific to that trace environment and collection instance. To enable the testing of the model using a wider variety of different yet still comparable traces, we have also generated some alternate versions of the trace files previously used. A simple mechanism was used to preserve the aggregate colocation density and durations of these alternate traces. Each colocation is iterated over, with the participating nodes being swapped with another randomly chosen colocation. The time of occurrence

Trace	Hosts	Length (1000 seconds)
TfL	4,000	258
Reality	100	31,608
Haggle	3,500	1,018
Unitrans	25	436

Table 7.3: Trace specific parameters.

and duration are unchanged, leading to preservation of many trace properties.

7.3.3 Metrics

To judge the effect that each advertising policy has on the distribution of file types in the system as a whole, we will use the Kullback–Leibler divergence (KL) [KL51]. It is a non-symmetric measure of how two probability distributions *P* and *Q* differ (Equation 7.4). KL measures the expectation of the number of extra bits required to code samples from *P* when using a code based on *Q*. Hence, it is loosely a measure of how much more information is contained in the distribution *Q* compared to *P*, or how different it is. It is not strictly a true 'metric' in the statistical sense.

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{7.4}$$

If *P* is defined as the starting distribution of the genre frequencies (i.e., the initial $Libsize_g$ across all G) and *Q* is the distribution of genres at the end of the simulation. Hence, for our purposes, a KL value of 0 signifies that the resulting proportional distribution of files is exactly the same as the initial one. Note that this says nothing about the number of files in the system, only their relative proportions.

7.4 Results

The results from our model are presented in this section. First the grid structure is tested, to allow details specific to the sharing process to be examined separately from the movement effects. The trace files are then presented to confirm that a similar behaviour is exhibited even when using a variety of movement scenarios. The model uses Mersenne twisters [MN98] to generate random numbers required in each run of the model. To gather accurate results, the model is run with a different random seed at least 100 times. In all experiments, relative stability of results was achieved after 50 runs.

7.4.1 Grid

The most basic result from our model, when each host only likes 1 genre ($h = 1$) out of a possible 10, is portrayed in Figure 7.3. The genres are all distributed according to a Zipf distribution with exponent 1. Hence the most popular genre $g1$ is nearly an extra 20% as popular as the second most $g2$. There are three stacked bars, each representing the distribution of genres, i.e., each sector of the bar is a single genre. The first bar is the absolute number of each genre in the *Initial* model creation. The second bar is the absolute number of each genre at the *End* of the experiment, there is then a third bar of the initial distribution scaled up to the *End* magnitude so that a proportional visual comparison can be made. Note that a genre selection policy would have no effect in this formulation of the model, as hosts only enjoy one genre and so have no choice about what they download. Even if hosts were interested in multiple genres, the total number of files shared would not significantly change. In fact, the expected E number of transfers (throughput of the network) is invariant across all advertising policies. This is because the policy does not affect the source node selection, it is purely selected by the random number generator.

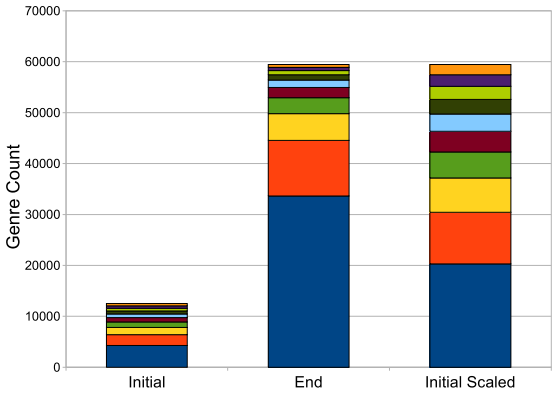

Figure 7.3: Grid: Genre Count.

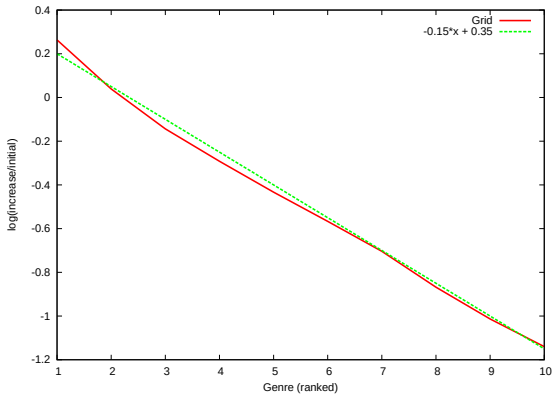

Figure 7.4: Initial vs Increase Relationship.

As expected all genres increase in frequency due to performing the transfers (i.e. End is larger than Initial), however the proportion of genres does not remain constant. The more popular genres are increased disproportionately, genre $g1$ (the most popular) represents 39.4% of all transfers but only 18.8% of hosts. This is due to an *availability bias* caused by the tastes of a host's neighbours. If a given host likes the most popular genre, it is more likely to be able to find an available neighbour that shares an interest, and is thus able to share files more often. Hosts with very niche tastes may not have any neighbours sharing an interest, and thus be unable to gather more files. This pathological case is a problem mostly limited to the grid

network as hosts maintain a short static set of neighbours.

This availability bias can be quantified if the increase is considered with reference to the amount of other hosts that are interested in a given genre. The average proportional increase for hosts of a given genre can be calculated through dividing the proportion of a genre that is shared during the simulation by the proportion of hosts that like that genre. If a logarithm of the average proportional increase is plotted (Figure 7.4) a linear relationship is revealed using least squares fitting, with Pearson's correlation coefficient $R = 0.9956$. This indicates that there is an exponential relationship between the popularity of the genre and the average proportional increase. This underlying process will be present in all subsequent experiments, even when hosts like multiple genres and are using a selection policy. In fact, this underlying process is something that should be actively combated if it is desired that the system not disproportionally favour the popular genres.

Figure 7.5 allows a more precise display of the effect of the running experiment, with a curve showing the initial genre distribution (as in Section 7.3) and the distribution at the end of the experiment, with small confidence intervals. The genres are not strictly continuous values, but they are shown as lines rather than bars to enable easy comparison. Note the log-log nature of the axis, the gradient of the lines corresponds to the exponent of a Zipf distribution. A gradient close to 0 reflects an even distribution of genres, the more negative the greater availability is skewed to the popular genres. If a more even distribution of files is desired, running the model should lead to an increase in the gradient of the distribution curve. When running with a uniform distribution of genres there is no change in the distribution across the system. All genres are increased by the same amount.

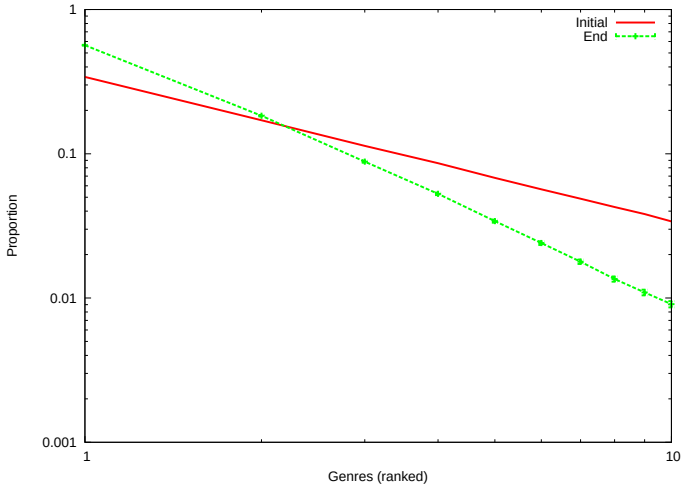

Figure 7.5: Grid: Genre Increase.

To investigate the policy effects, we now introduce hosts with multiple genre interests ($h = 5$) and a larger population of possible genres ($|G| = 100$). The system will now exhibit the effect from the lack of replacement in the genre selection and will not be exactly Zipf distributed in the Initial construction. The effect of varying the selection policy can be seen in Figure 7.6. All policies show a mostly similar behaviour for each genre, with popular tastes being shared much more that niche ones. There is a familiar decrease in the gradient of all resultant curves compared to the Initial distribution. Each policy has a different gradient, with Unpopular $>$ Random $>$ Popular. This shows that the Unpopular policy shares more of the less frequent genres and less of the popular than the other policies. They also all perform near identically for the genre of rank two. The logarithmic scale should be noted, as the difference between policies for the top ranked genre is significant. Out of the total number of exchanges, the Popular policy has 49.5% being of type $g1$, compared to the Unpopular policy's 40.0%. This represents an almost 10% increase in total transfers for genres of rank lower than two, when using the Unpopular policy.

If files have a more pronounced Zipf distribution, i.e., the exponent is larger, the more the population is skewed to the more popular ones. Figure 7.7 shows only the distributions from the outcome of the experiments, the impact on the resulting distribution when using the Unpopular policy and varying a. Setting a to 0 obviously results in a uniform popularity across all genres, and thus a uniform resulting distribution. When $a = 2$ the most popular genre is vastly more popular than the rest. This leads to an even greater flattening for the most popular genres at the expense of the less popular. Though the simple *Popular* and *Unpopular* policies give clearer results, all of the following results in this chapter will now use *Proportional Popular* and *Proportional Unpopular*.

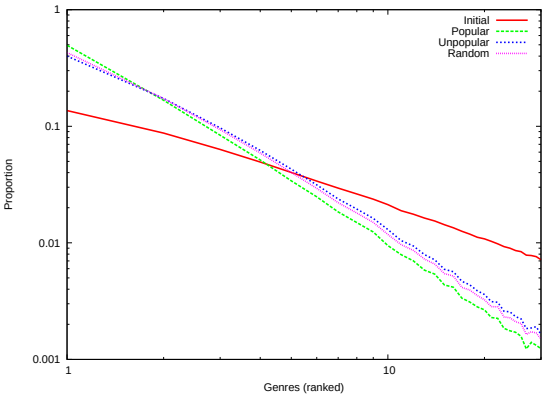

Figure 7.6: Grid Policy Effects (Five Genres).

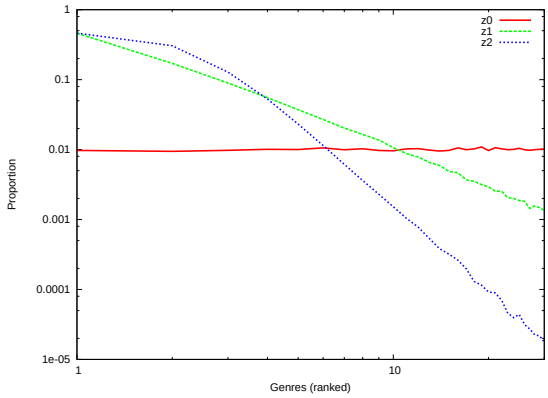

Figure 7.7: Grid Zipf Effects (Five Genres).

7.4.2 Traces

With the behaviour of the grid investigated, we will now briefly demonstrate the effect of applying the model to the trace files. The hosts will now experience transfer disconnections, as they are now dynamic and could change their connection state at any point. The effect of file disconnections is not considered, as all transfers last for the same duration, and the source selection procedure is the same for all experiments in this chapter. First, the model is run on the TfL trace set, as shown in Figure 7.8, using its default parameters. It exhibits the same behaviour as the grid, even though hosts are constantly changing their connections and they possess different levels of connectivity (i.e., some hosts are super-peers). The Initial global library content distribution is unchanged. Interestingly, all the distributions for the *Increase* in genre types have an even lower gradient than the grid. The most important aspect from the graph is that the gradients of the policies follow the same ordering of Unpopular $>$ Random $>$ Popular.

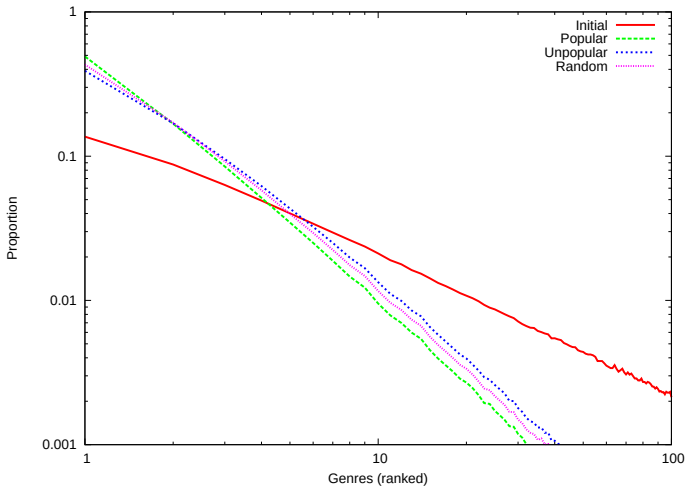

Figure 7.8: TfL: Genre Increase.

Both Haggle and Unitrans trace sets (Figure 7.9 and Figure 7.10) behave similarly, with Unpopular giving a more even distribution for the increase in genre availability. There is more noise in these results compared to the previous results, likely due to regularity of the grid, and high mixing of the TfL data. The gradient of the policy plots are also less than when using the grid structure.

To distill the distribution change that occurs to the model into a scalar value, the KL distance is shown in Table 7.4. A visual representation is also given in Figure 7.11. The results only have true comparability between ones operating on the same trace files, due to the variable experiment size and duration. The table is organised with the policies in descending order of KL distance from their initial distribution. To calculate these, the KL distance was taken from

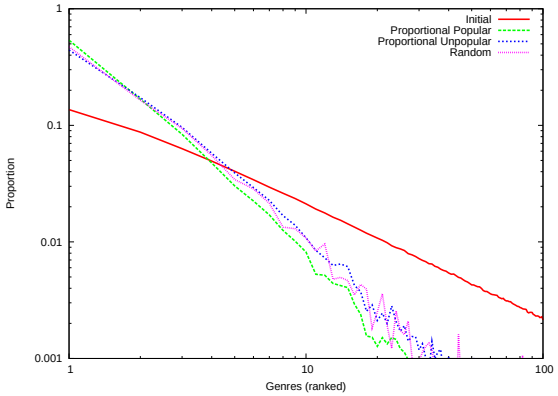

Figure 7.9: Haggle: Genre Increase.

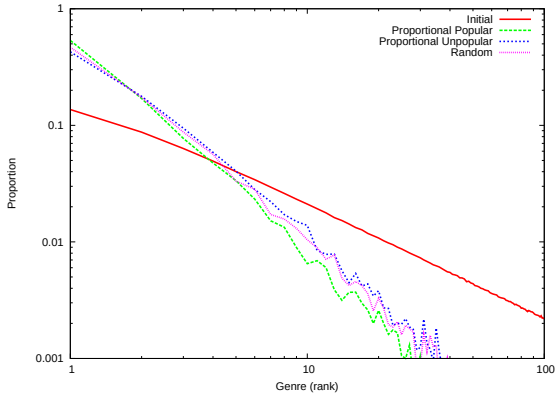

Figure 7.10: Unitrans: Genre Increase

Policy	Grid	TfL	Haggle	Unitrans
Popular	0.2430	0.4881	0.18243	0.0231
Popular Proportional	0.2212	0.4286	0.16578	0.0197
Random	0.2014	0.3576	0.14299	0.0162
Unpopular Proportional	0.1882	0.3061	0.13562	0.0147
Unpopular	0.1846	0.2735	0.13271	0.0148

Table 7.4: Kullback–Leibler Distance.

each run of an experiment and the mean of all runs calculated.

7.4.3 Summary

This chapter has defined a constrained model for accumulated item transfer in a network. It presented some possible file negotiation policies for hosts to follow and the effect they had on which file categories were shared. We showed that the absence of specific intent in the file sharing process will lead to a disproportionate amount of replication for genres of a popular nature. Furthermore, we showed that using a local stateless mechanism for estimating a categories popularity and striving to share the less popular files could lead to a reduction in the reinforcement of the most popular file types. However, even when explicitly only trying to share the least popular genres, popular files were still shared more, due to an availability bias.

The conceptualisation of a basic model of file sharing, based on networked urn processes, has shown some possible implications for a deployed system. Most importantly, we have shown that localised decisions on which content to share can have a large effect on what types of content gets shared by the network as a whole. The regular grid provided a simplified version that still exhibits features of the more complicated networks. Applying the structure of human car-

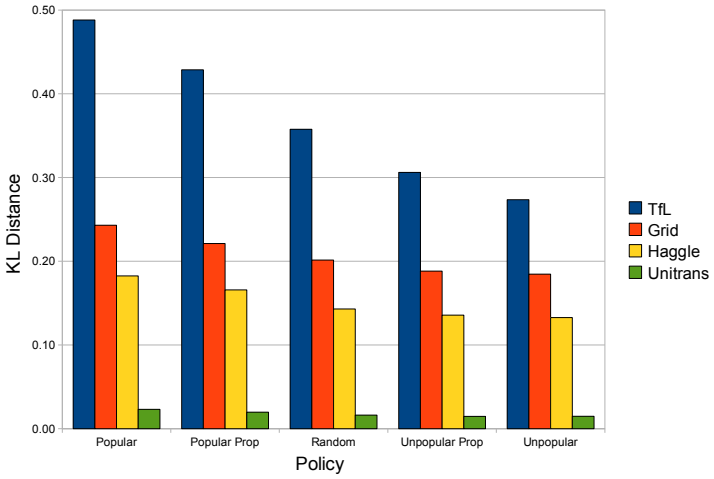

Figure 7.11: KL Distance vs Policy

ried device networks did add some noise to the results, but the same effect of the policies was comparable to the grid. As has been mentioned in other literature [Axe97], there appears to be an tendency for the homogenisation of taste, where popular categories are made even more popular. If random spreading of data was the sole process acting in a network, as would be possible in our scenario, then it appears that mainstream tastes would naturally dominate. The traces files also showed an even greater preference for popular files. This is due to the traces have not segregating users as in the static minimally connected grid structure. Hence, small islands where only niche tastes can ever be shared are never created. Even when a localised attempt is made to reduce the popularity skew, the popular tracks are still shared more. Though there is an improvement in fairness when attempting to share unpopular genres. This indicates that a more sophisticated technique would have to be used to achieve equality in sharing frequency for all genres.

8

Conclusion

This chapter provides a summary and critical evaluation of the work presented in this thesis, together with some possible future directions.

8.1 Summary of Thesis

This thesis proposed the idea that wireless device-to-device connectivity can provide data of interest to users. Indeed, the volume of data content surrounding a typical urbanite during their day is vast, and accessibility to it would provide significant utility for people. Furthermore, we examined the state of technology that facilitates such interactions and saw that many components have long been available. However there has only recently been widespread public and corporate engagement with the use of this technology. The scenario on which we focused our consideration, was in the sharing of music files between people using mass transit systems in urban environments. Some background investigation was required into how the categorisation of music data was performed by users, and more importantly how different categories and artists vary in popularity. An equally important concern was the movement of users when on public transport systems. Another consideration was how to limit the effect of peers with incorrectly encoded/labelled files or even intentionally corrupt files.

A solution was devised that employed automatic 1-hop peer-to-peer file transfers. This allows the system to gradually accrue files of interest when a suitable neighbour was present. Only files of a category that a user had already shown enough interest to acquire would be collected. Not all potential transfers were initiated, in fact it was found to be useful to be conservative in the initiation of downloads. The decision of whether to initiate communication was based on predictions of the neighbours future colocation duration. This prediction was informed by the previous length of colocations. Neighbour and temporal specific historical information took precedence over the aggregate history of all other neighbours. Any neighbours that were deemed to have misbehaved were remembered and avoided in subsequent interactions. The devices were assumed to be sharing their data over short range Bluetooth networks; however, any short ranged wireless technology could operate the connection.

Automatic sharing of content between users' personal devices on urban transport was shown to be a plausible scenario. The selection of whom to use a download source was also shown to be important to the efficiency of the system. Furthermore, prior knowledge of colocation history could be used to gain confidence in the likelihood of a data transfer completing. In dense, high churn networks, this knowledge can be approached using tractable, low state techniques.

Even when people have extremely varied interests in the available content, they can still have their niche tastes satisfied. Attention to the priority of which files are shared is essential to ensure that replication of tracks is not focused upon a select few popular tracks from a subset of users. This protection against the complete homogenisation of category distribution (even between items within a particular category) is important for ensuring diverse user libraries. People can independently choose which advertising strategy their device uses, and so can impact which files replicated by the system as a whole. They could also choose to employ a trust system, which, dependent on their feedback can enable the avoidance of hosts that have sourced bad files in the past.

Following from Metcalfe's Law [BOT06], if there are very few users of a network, it is of negligible value. Our proposed system requires that other users running the software are actually present in the environment and also share some interests for any utility to be gained. As with most wireless systems, there is also an upper bound on how dense the network can be, while still performing useful communication, due to contention. Our approach mitigates the effect of excess network load by not indiscriminately initiating downloads. Specifically avoiding them if the transfer seems unlikely to complete in the predicted available time. The tube's Oyster traces that comprised roughly a quarter of passengers (according to Transport for

London). A penetration of 25% would be very high, even assuming every person possesses a device. Our simulations also only considered the most frequently travelling users, generally at peak times. Leading to an environment with users of the system frequently seeing other users, sometimes in dense configurations. A scenario that presumes moderate yet large scale adoption by the public at large. However, even at high densities, we can still maintain system capacity, though the per user capacity will decrease. Thankfully, the node density can not increase unrestricted, even on the tube, due to the minimum space required for the actual human beings.

8.1.1 Contributions

The contributions of this thesis were as follows:

Colocation Prediction – Proposal and analysis of the efficacy and computability of a range of mechanisms for estimating the length of a neighbours colocation. We used a variety of wireless computer traces to study human movement in urban transport environments. Particularly when using mass transit systems in a large metropolitan city. It was shown that with minimal storage requirements, devices are able to decide whether to initiate transfers that are more likely to complete.

Advertising Strategies – The order that users advertise subsets of their library significantly effects which tracks are downloaded. The advertisement ordering process and its impact on the movement of files through the network is examined, specifically how to satisfies the different interests of users and ensure a variety of content is available throughout the network.

Music Taste Modelling – A measurement study of users' music tastes from a popular social music website was performed. Leading to the construction of a realistic and configurable model for users' digital libraries and music tastes. This modelling allows the investigation of various advertising policies and file selection procedures, providing results on how they affect system performance.

Malicious Peer Avoidance – We utilised an existing trust system to prevent the accumulation of files that a user does not want. We saw that even in networks with very high churn, the effect of badly performing devices could be reduced through learning from previous bad experiences.

Content Distribution Model – A simplified model of categorised item distribution was defined and the effect of using different advertising policies was presented. It was demon-

strated that the informed download selection of data items can be used to affect the systemic distribution of files of different types.

System Evaluation – An application for mobile phones that implements the ideas and algorithms presented in this thesis was created, together with results of its behaviour in an actual urban transport environment. The application was tested in a real environment on a city subway system, where its actions were timed. These timings were then used to inform a city-scale simulation of the proposed system. This simulation was used for the bulk of our analysis and was able to simulate a large portion of users from the London Tube system for a month. The results show that the careful selection of a download source and the downloaded file is a feasible and useful strategy for content source selection.

8.2 Critical Evaluation

Despite the use of real music listening habits, we have not considered how the proposed system would be accessed by real users. People may desire a system of the type proposed to mostly collect music they are unfamiliar with, that they would not be exposed to from their usual sources of music information (e.g., friends, the music press and public radio).

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. Even in large metropolitan cities, the regularity of peoples' movement can still be leveraged to identify familiar strangers, and exploit the learnt colocation patterns. Though our results have mostly been validated in the domain of public transport movement, and some related movement patterns, it was not applied to other human activities, such as places of work, coffee-shop or pub visiting. Our dataset allowed the modelling of a large metropolitan city, and displayed expected properties, with encouraging regularity of movement. The application of our selection scheme allowed passengers to identify other promising candidates for consistent connection. 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 selection) may initiate more transfers, but lead to more incomplete transfers, increasing the burden on the network (reducing neighbours' throughput), wasting time that could be spent on successful downloads and needlessly draining device battery-life. This can still be the correct approach in environments with few neighbours and long term colocations, such as in the Reality Mining traces. Our approach significantly reduces failed transmissions, saving energy and time to be used on successfully receiving content, when hosts only initiate a download they predicted would succeed,

it leads to significantly greater performance. The download threshold parameter can be tuned to effect download initiation likelihood, and optionally reduce the amount of download failures (at the expense of some successes from false negatives). Its value could 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. With additional context-awareness about the environment it should be possible to dynamically adjust the download threshold and achieve the best of both approaches.

Though the focus has been on sharing music files, apart from the Last.fm based interest distribution in the evaluation, other aspects would hold true when applied to other types of bulk data. The main possible difference with sharing other types of bulk data would be if it was of significantly larger size, such as high quality videos. This would make the accurate prediction of link duration even more important, as it would be more likely that transfers would not complete in time. Non-music data would also likely have a different distribution of popularity between categories and items. If the category popularity distribution was extremely heavily skewed to the popular end, then it would be likely that an opportunistic wireless data sharing system would almost totally marginalise the less popular content; as users with niche tastes would almost never meet. In such circumstances, it would require a different approach to the problem of file distribution, transfers would likely not be independently negotiated by pairs of devices.

If it was not bulk data being shared, but rather a type of data that was useful when only partially complete (e.g., news) the source selection would not be as important. It would be possible to transmit a majority of the utility of the data quickly (i.e. headline and story text) less important information (e.g., associated pictures, audio or video) could then be transmitted, with any termination of the transfer not significantly harming the quality of the received item.

Another possible modification to the system would be using a different wireless technology. A high bandwidth technology would also reduce the importance of colocation prediction for data transfers. If the increased data rates only acted to increase the size of files that people wanted to share, then the problem of neighbour colocation prediction would persist. Also, if the types of activity being performed was not dependent on data volume but time (such as peer-to-peer games) then the need for selecting a long-term neighbour would still be of large importance.

8.3 Current Research Directions

The work in this thesis provides a framework of analysis for the developing of wireless peer-to-peer content sharing. The field could be further advanced in a number of ways, including:

- The inclusion of partial file transfers, allowing peers to resume downloads that are disconnected during their transfer, similar to BitTorrent. This behaviour should complement our colocation prediction, allowing transfers to complete more frequently and be resumed if they are interrupted.

- Deployment of the implementation, to enable usage, interaction and opinions from real human users and their mobile devices. This would allow testing of their satisfaction with the system behaviour. Also, it would allow analysis of how the system performs with other unmodelled user actions such as manual addition/deletions of tracks from the library and some real feedback in to the trust system that was presented.

- Deeper analysis of the file advertising policies and their effect on data that has more metadata available. For instance, giving preference to newly released tracks, enabling people to quickly receive copies of the popular tracks in the charts at that period of time.

- Consideration of the use of automatic file sharing between users that know each other, and the possible effects this has on a users taste. Potentially even the use of friends music tastes to decide which files should be selected, as a form of mobile recommender system.

- Development of a more realistic content spreading model, that includes file level modelling. The addition of network level behaviour, such as contention and variable link speeds, would also reduce its level of abstraction. This would then allow the inclusion of source selection behaviour, an ability that has been shown to have a significant impact on system performance.

As a final comment, it should be mentioned that device-to-device wireless technologies are inexorably improving in throughput, range and power costs. These technologies, in tandem with the ballooning of device and data ownership, will only lead to a flourishing in the desire for flexible and easy to manage data sharing paradigms. The involvement of industry in this sphere of technology would be stimulating, though not necessary. The reducing costs of owning a capable device, and of running a system such as proposed is also becoming almost trivial. The proposal that mobile devices will improve in their ability to store reams of data, consume content and transmit it on short-range wireless links should be taken for granted. The only question is whether connectivity through infrastructure will come to dominate how people access data of importance to them. While the promise of cloud computing is attractive, it will never be the sole or best way to access data in all circumstances. Particularly for people wishing to avoid control of the data they wish to communicate and acquire. If opportunistic distributed data dissemi-

nation systems are to become widely deployed, their efficient, robust and useful functioning is paramount. We have analysed one of the most challenging (and possibly rewarding) environments to operate such a system: on a large urban subway system. This thesis goes someway to investigating the total problem space and delivers some important design considerations for the creation of a functioning, deployable software system.

Bibliography

[ABR07] 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, October 2007. ACM.

[AG07] Donovan Artz and Yolanda Gil. A Survey of Trust in Computer Science and the Semantic Web. *Web Semantics*, 5(2):58–71, June 2007.

[ARH00] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*, page 6007, Washington, DC, USA, January 2000. IEEE Computer Society.

[Axe97] Robert Axelrod. The Dissemination of Culture: A Model with Local Convergence and Global Polarization. *Journal of Conflict Resolution*, pages 203–226, 1997.

[BBM+07] Arianna Bassoli, Johanna Brewer, Karen Martin, Paul Dourish, and Scott Mainwaring. Underground Aesthetics: Rethinking Urban Computing. *IEEE Pervasive Computing*, 6(3):39–45, July 2007.

[BKA09] Eytan Bakshy, Brian Karrer, and Lada Adamic. Social Influence and the Diffusion of User-Created Content. In *10th ACM Conference On Electronic Commerce*. ACM, July 2009.

[BMA04] Arianna Bassoli, Julian Moore, and Stefan Agamanolis. tunA: Synchronised Music-Sharing on Handheld Devices. In *Adjunct Proceedings of Ubicomp*, September 2004.

[BOT06] Bob, Andrew Odlyzko, and Benjamin Tilly. Metcalfe's Law is Wrong - Communications networks increase in value as they add members-but by how much? *IEEE Spectrum*, 43(7):34–39, 2006.

[BWG+07] Liam J. Bannon, Ina Wagner, Carl Gutwin, Richard H. R. Harper, and Kjeld Schmidt. Gifts from Friends and Strangers: A Study of Mobile Music Sharing. In *ECSCW: Proceedings of the 10th European Conference on Computer-Supported Cooperative Work*, pages 311–330. Springer London, September 2007.

[CABP05] Rajiv Chakravorty, Sulabh Agarwal, Suman Banerjee, and Ian Pratt. MoB: A Mobile Bazaar for Wide-Area Wireless Services. In *International Conference on Mobile Computing and Networking (MobiCom)*, pages 228–242. ACM Press, August 2005.

[CCH+08] David Crandall, Dan Cosley, Daniel Huttenlocher, Jon Kleinberg, and Siddharth Suri. Feedback Effects Between Similarity and Social Influence in Online Communities. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 160–168, New York, NY, USA, 2008. ACM.

[CDV+02] Fabrizio Cornelli, Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Implementing a Reputation-Aware Gnutella Servent. In *Revised Papers from the Networking 2002 Workshops on Web Engineering and Peer-to-Peer Computing*, pages 321–334, London, UK, 2002. Springer-Verlag.

[CGD+09] Jilin Chen, Werner Geyer, Casey Dugan, Michael Muller, and Ido Guy. Make New Friends, but Keep the Old - Recommending People on Social Networking Sites. In *International Conference for Human-Computer Interaction. (CHI)*. ACM, April 2009.

[CH97] Bruce Christianson and William S. Harbison. Why Isn't Trust Transitive? In *Proceedings of the International Workshop on Security Protocols*, pages 171–176, London, UK, April 1997. Springer-Verlag.

[Cre09] Creative Commons. http://creativecommons.org/licenses/, November 2009.

[Dar06] Dartmouth College. CRAWDAD: Community Resource for Archiving Wireless Data. Available at http://crawdad.cs.dartmouth.edu/, November 2006.

[DDB04] Prashant Dewan, Partha Dasgupta, and Amiya Bhattacharya. On Using Reputations in Ad hoc Networks to Counter Malicious Nodes. In *International Conference on Parallel and Distributed Systems (ICPADS)*, page 665, July 2004.

[DHGS07] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu. Characterizing Residential Broadband Networks. In *The 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 43–56, New York, NY, USA, October 2007. ACM.

[DM07] Vladimir Dyo and Cecilia 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.

[Dou02] John R. Douceur. The Sybil Attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, March 2002. Springer-Verlag.

[Eam06] Eamonn O'Neill and Vassilis Kostakos and Tim 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)*, September 2006.

[Edw06] W.K. Edwards. Discovery systems in Ubiquitous Computing. *IEEE Pervasive Computing*, 5(2):70–77, 2006.

[EP05] Nathan Eagle and Alex (Sandy) Pentland. CRAWDAD trace mit/reality/blueaware/devicespan (v. 2005-07-01). Downloaded from http://crawdad.cs.dartmouth.edu/mit/reality/blueaware/devicespan, July 2005.

[Fal03] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 27–34. ACM New York, NY, USA, 2003.

[Fun07] UNFPA United Nations Population Fund. State of World Population 2007 - Unleashing the Potential of Urban Growth, 2007.

[Gam88] Diego Gambetta. Can We Trust Trust? In *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Basil Blackwell, April 1988.

[GBNQ06] Joy Ghosh, Matthew J. Beal, Hung Q. Ngo, and Chunming Qiao. On profiling mobility and predicting locations of wireless users. In *REALMAN '06: Proceedings of the 2nd International Workshop on Multi-hop ad hoc Networks: from Theory to Reality*, pages 55–62, New York, NY, USA, May 2006. ACM.

[GK00] Piyush Gupta and P.R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46:388–404, March 2000.

[GM03] Chao Gui and Prasant Mohapatra. Efficient Overlay Multicast for Mobile Ad Hoc Networks. In *Wireless Communications and Networking (WCNC) 2003*, volume 2, pages 1118–1123. ACM, March 2003.

[Gro97] IEEE 802.11 Working Group. IEEE 802.11-1997: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 1997.

[GS00] Tyrone Grandison and Morris Sloman. A Survey of Trust in Internet Applications. In *IEEE Communications Surveys and Tutorials*, volume 3, pages 2–16, September 2000.

[GT02] Matthias Grossglauser and David N.C. Tse. Mobility Increases the Capacity of Ad Hoc Wireless Networks. *IEEE/ACM Transactions on Networking (ToN)*, 10(4):477–486, 2002.

[Guh03] Ramanathan Guha. Stanford University Technical Report: Open Rating Systems, 2003.

[Har04] Joona Hartman. Implementing a Service Discovery Mechanism for Mobile IPv6 Environments. Master's thesis, Institute of Communications Engineering at Tampere University of Technology, August 2004.

[HCS+05] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe 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.

[HCY08] Pan Hui, Jon Crowcroft, and Eiko Yoneki. BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks. In *Proc. of 9^{th} ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 241–250, Hong Kong, May 2008.

[HFL+04] Andreas Haeberlen, Eliot Flannery, Andrew M. Ladd, Algis Rudys, Dan S. Wallach, and Lydia E. Kavraki. Practical Robust Localization over Large-scale 802.11 Wireless Networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 70–84, New York, NY, USA, September 2004. ACM Press.

[HGPC99] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A Group Mobility Model for Ad Hoc Wireless Networks. In *MSWiM '99: Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 53–60, New York, NY, USA, 1999. ACM.

[HGRW06] Tobias Heer, Stefan Gotz, Simon Rieche, and Klaus Wehrle. Adapting Distributed Hash Tables for Mobile Ad hoc Networks. *Pervasive Computing and Communications Workshops. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 173–178, March 2006.

[HJSS06] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information Retrieval in Folksonomies: Search and Ranking. In *The Semantic Web: Research and Applications, volume 4011 of LNAI*, pages 411–426. Springer, 2006.

[ICM09] Stratis Ioannidis, Augustin Chaintreau, and Laurent Massoulie. Optimal and Scalable Distribution of Content Updates over a Mobile Social Network. In *Proc. of IEEE Conference on Computer Communications INFOCOM*, April 2009.

[ILY07] Yin-Ki Ip, Wing-Cheong Lau, and On-Ching Yue. Forwarding and Replication Strategies for DTN with Resource Constraints. In *IEEE 65th Vehicular Technology Conference, 2007. VTC2007-Spring*, pages 1260–1264, 2007.

[Int09] International Telecommunication Union - http://www.itu.int. World Telecommunication/ICT Indicators Database 2008, February 2009.

[Jas06] Jason LeBrun and Chen-Nee. Chuah. CRAWDAD movement data set ucdavis/unitrans (v. 2006-11-01). Downloaded from http://crawdad.cs.dartmouth.edu/ucdavis/unitrans, November 2006.

[JLC+07] Sewook Jung, Uichin Lee, Alexander Chang, Dae-Ki Cho, and Mario Gerla. BlueTorrent: Cooperative Content Sharing for Bluetooth Users. *IEEE International Conference on Pervasive Computing and Communications (Percom)*, pages 47–56, March 2007.

[JLM01] Douglas S. J. De Couto Hu Imm Lee Jinyang Li, Charles Blake and Robert Morris. Capacity of Ad Hoc Wireless Networks. In *Proceedings of the 7th annual International Conference on Mobile Computing and Networking*, page 69. ACM, 2001.

[JRHH05] Mattias Jacobsson, Mattias Rost, Maria Håkansson, and Lars Erik Holmquist. Push!Music: Intelligent Music Sharing on Mobile Devices. In *In Adjunct Proceedings of UbiComp 2005, the 7th International Conference on Ubiquitous Computing*, pages 11–14, September 2005.

[JX07] Hai Jin and Jie Xu. TRES-CORE: Content-Based Retrieval Based on the Balanced Tree in Peer to Peer Systems. In *Parallel Computing Technologies (PaCT)*, pages 215–229. Springer, September 2007.

[KBV07] 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, Sepember 2007. ACM.

[Kha06] Jeffrey P. Kharoufeh. Bluetooth Inquiry Time Characterization and Selection. *IEEE Transactions on Mobile Computing*, 5(9):1173–1187, September 2006. Brian S. Peterson and Rusty O. Baldwin.

[KL51] Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.

[KMM34] William Ogilvy Kermack, Anderson Gray McKendrick, and PL McKinlay. Death-Rates in Great Britain and Sweden: Expression of Specific Mortality Rates as Products of Two Factors, and Some Consequences Thereof. *The Journal of Hygiene*, 34(4):433–457, 1934.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, pages 640–651, New York, NY, USA, May 2003. ACM Press.

[las09] Last.fm Website - http://www.last.fm, Audioscrobbler data source - http://www.audioscrobbler.net, November 2009.

[LC06] Jason LeBrun and Chen-Nee Chuah. Bluetooth Content Distribution Stations on Public Transit. In *MobiShare '06: Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking*, pages 63–65, New York, NY, USA, September 2006. ACM Press.

[LC08] Glenn Lyons and Kiron Chatterjee. A Human Perspective on the Daily Commute: Costs, Benefits and Trade-offs. *Transport Reviews*, 28:181–198, March 2008.

[LDDG09] Anders Lindgren, Avri Doria, Elwyn Davies, and Samo Grasic. Probabilistic Routing Protocol for Intermittently Connected Networks: draft-irtf-dtnrg-prophet-02. RFC 1, March 2009.

[LHSC05] Anthony LaMarca, Jeffrey Hightower, Ian E. Smith, and Sunny Consolvo. Self-mapping in 802.11 location systems. In *International Conference on Ubiquitous Computing (Ubicomp)*, pages 87–104, September 2005.

[LI04a] Jinshan Liu and Valérie Issarny. Enhanced reputation mechanism for mobile ad hoc networks. pages 48–62. February 2004.

[LI04b] Jinshan Liu and Valerie Issarry. QoS-Aware Service Location in Mobile Ad-Hoc Networks. *International Conference on Mobile Data Management (MDM)*, page 224, January 2004.

[LI07] Jinshan Liu and Valérie Issarny. An incentive compatible reputation mechanism for ubiquitous computing environments. *International Journal on Information Security*, 6(5):297–311, August 2007.

[LLS+06] Jérémie Leguay, Anders Lindgren, James Scott, Timur Friedman, and Jon Crowcroft. Opportunistic Content Distribution in an Urban Setting. In *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged Networks*, pages 205–212, New York, NY, USA, September 2006. ACM.

[LZZ+06] Shan Lin, Jingbin Zhang, Gang Zhou, Lin Gu, John A. Stankovic, and Tian He. ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks. In *SenSys '06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pages 223–236, New York, NY, USA, October 2006. ACM.

[Mas06] Paolo Massa. A survey of trust use and modeling in current real systems. In R. Song, L. Korba, and G. Yee, editors, *Trust in E-Services: Technologies, Practices and Challenges*. Idea Group Publishing, November 2006.

[MHM05] Mirco Musolesi, Stephen Hailes, and Cecilia Mascolo. Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks. In *Proceedings of the IEEE 6th International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM 2005). Taormina, Italy.* IEEE press, June 2005.

[MK90] Ryoichi Mori and Masaji Kawahara. Superdistribution: The Concept and the Architecture. *Transactions of The Institute of Electronics, Information, and Communication Engineers*, page pp.11331146, July 1990.

[MLC01] Miller Mcpherson, Lynn S. Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[MLKW07] Martin May, Vincent Lenders, Gunnar Karlsson, and Clemens Wacha. Wireless Opportunistic Podcasting: Implementation and Design Tradeoffs. In *CHANTS '07: Proceedings of the second ACM workshop on Challenged networks*, pages 75–82, New York, NY, USA, September 2007. ACM.

[MM06] Mirco Musolesi and Cecilia Mascolo. A Community based Mobility Model for Ad Hoc Network Research. In *Proceedings of the 2nd ACM/SIGMOBILE International Workshop on Multihop Ad Hoc Networks: From Theory to Reality (REALMAN'06)*, pages 31–38. ACM Press, May 2006.

[MMC06] Liam McNamara, Cecilia Mascolo, and Licia Capra. Trust and Mobility Aware Service Provision for Pervasive Computing. In *Pervasive 2006 Workshop Proceedings: 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI)*, pages 603–610. Springer-Verlag, May 2006.

[MMC07] Liam McNamara, Cecilia Mascolo, and Licia Capra. Content Source Selection in Bluetooth Networks. In *Proc. of International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous)*, pages 1–8, August 2007.

[MMC08] Liam McNamara, Cecilia Mascolo, and Licia Capra. Media Sharing based on Colocation Prediction in Urban Transport. In *Proc. of ACM 14th International*

Conference on Mobile Computing and Networking (Mobicom08), pages 58–69, San Francisco, CA, September 2008.

[MN98] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[MS02] Natalia Marmasse and Chris Schmandt. A User-Centered Location Model. *Personal Ubiquitous Computing*, 6(5-6):318–321, December 2002.

[NDGG07] Erik Nordström, Christophe Diot, Richard Gass, and Per Gunningberg. Experiences from measuring human mobility using bluetooth inquiring devices. In *MobiEval '07: Proceedings of the 1st international workshop on System evaluation for mobile platforms*, pages 15–20, New York, NY, USA, 2007. ACM.

[NN08] Anthony J. Nicholson and Brian D. Noble. BreadCrumbs: Forecasting Mobile Connectivity. In *MobiCom '08: Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, pages 46–57, New York, NY, USA, September 2008. ACM.

[Obr04] Philipp Obreiter. A case for evidence-aware distributed reputation systems - overcoming the limitations of plausibility considerations. In *Trust Management (iTrust)*, volume 2995 of *Lecture Notes in Computer Science*. Springer, March 2004.

[Old89] Ray Oldenburg. *The Great Good Place: Cafes, Coffee Shops, Community Centers, General Stores, Bars, Hangouts, and How They Get You Through the Day*. Paragon Books, 1989.

[ORT09] Jukka-Pekka Onnela and Felix Reed-Tsochas. The spontaneous emergence of social influence in online systems. Dec 2009.

[Peh07] Jon M. Pehal. The Benefits and Risks of Mandating Network Neutrality, and the Quest for a Balanced Policy. *International Journal of Communication*, December 2007.

[Pen07] Alex Sandy Pentland. Automatic Mapping and Modeling of Human Networks. *Physica A: Statistical Mechanics and its Applications, v. 378*, pages 59–67, May 2007.

[PG04] Eric Paulos and Elizabeth Goodman. The Familiar Stranger: Anxiety, Comfort, and Play in Public Places. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 223–230, New York, NY, USA, April 2004. ACM Press.

[PH06] Mark E. J. Newman Petter Holme. Nonequilibrium Phase Transition in the Coevolution of Networks and Opinions. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 74(5):37–55, 2006.

[PKGO09] Mikko Pitkänen, Teemu Kärkkäinen, Janico Greifenberg, and Jörg Ott. Searching for Content in Mobile DTNs. In *7th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10, March 2009.

[QHC06] Daniele Quercia, Stephen Hailes, and Licia Capra. B-trust: Bayesian trust framework for pervasive computing. In *Proceedings of the 4th International Conference on Trust Management, LNCS*, pages 298–312. Springer-Verlag, May 2006.

[QHC07] Daniele Quercia, Stephen Hailes, and Licia Capra. TRULLO - Local Trust Bootstrapping for Ubiquitous Devices. In *MobiQuitous*, pages 1–9, August 2007.

[RNGT05] Christian Rohner, Erik Nordström, Per Gunningberg, and Christian Tschudin. Interactions Between TCP, UDP and Routing Protocols in Wireless Multi-Hop Ad Hoc Networks. In *4th ACM/IEEE Conference on Mobile Computing and Networking: From Theory to Reality (REALMAN05)*, 2005.

[SDPG06] Sarafijanovic-Djukic, Michal Piórkowski, and Matthias Grossglauser. Island Hopping: Efficient Mobility Assisted Forwarding in Partitioned Networks. In *Proc. of the 3rd IEEE Communications Society Conference on Sensor, Mesh and Ad hoc Communications and Networks (SECON)*, pages 226–235, September 2006.

[SGC+06] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD trace cambridge/haggle/imote/cambridge (v. 2006-01-31). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote/cambridge, January 2006.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based cCollaborative Filtering Recommendation Algorithms. In *WWW '01: Proceed-*

ings of the 10th international conference on World Wide Web, pages 285–295, New York, NY, USA, May 2001. ACM.

[SKR99] J. Ben Schafer, Joseph Konstan, and John Riedi. Recommender Systems in E-Commerce. In *EC '99: Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 158–166, New York, NY, USA, 1999. ACM.

[SLC07] Mohamed Sordo, Cyril Laurier, and Oscar Celma. Annotating Music Collections How Content-based Similarity Helps to Propagate Labels. In *8th International Conference on Music Information Retrieval*, pages 531–534, Vienna, Austria, September 2007.

[SLG00] William Su, Sung-Ju Lee, and Mario Gerla. Mobility Prediction in Wireless Networks. In *IEEE Military Communications Conference (MILCOM)*, pages 491–495, October 2000.

[SMLN+03] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networks*, 11(1):17–32, 2003.

[SMM07] Giuseppe Sollazzo, Mirco Musolesi, and Cecilia Mascolo. Taco-dtn: A time-aware content-based dissemination system for delay tolerant networks. In *MobiOpp: Proceedings of the 1st International MobiSys workshop on Mobile Opportunistic Networking*, pages 83–90, New York, NY, USA, June 2007. ACM.

[SPGR08] Rossano Schifanella, André Panisson, Cristina Gena, and Giancarlo Ruffo. MobHinter: Epidemic Collaborative Filtering and Self-Organization in Mobile Ad-hoc Networks. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 27–34, New York, NY, USA, October 2008. ACM.

[SS05] Jordi Sabater and Carles Sierra. Review on Computational Trust and Reputation Models. *Artificial Intelligence Review*, 24(1):33–60, September 2005.

[SW04] Subhabrata Sen and Jia Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. *IEEE/ACM Transactions on Networks*, 12(2):219–232, April 2004.

[Swa02] Aaron Swartz. Musicbrainz: A Semantic Web Service. *IEEE Intelligent Systems*, pages 76–77, 2002.

[TG05] Cristian Tuduce and Thomas Gross. A Mobility Model based on WLAN Traces and its Validation. *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 1:664–674 vol. 1, March 2005.

[Tin03] Ting-Yu Lin and Yu-Chee Tseng. Collision Analysis for a Multi-Bluetooth Picocells. *IEEE Communications Letters*, 7:475–477, October 2003.

[TJK04] Muhammad Mukarram Bin Tariq, Ravi Jain, and Toshiro Kawahara. Mobility Aware Server Selection for Mobile Streaming Multimedia Content Distribution Networks. In *Web Content Caching and Distribution: Proceedings of the 8th International Workshop*, pages 1–18, October 2004.

[Tru05] Trusted Computing Group Whitepaper - http://www.trustedcomputinggroup.org. Trusted Platform Modules Strengthen User and Platform Authenticity, January 2005.

[UK 05] UK Government, Department for Transport - http://www.dft.gov.uk. Regional Transport Statistics: 2005 Edition, November 2005.

[UK 06] UK Government, Department of Transport - http://www.dft.gov.uk. Regional Transport Statistics: 2006 Edition, November 2006.

[Var01] Andras Varga. The OMNeT++ Discrete Event Simulation System. In *Proc. of the European Simulation Multiconference (ESM'01)*, pages 319–324, June 2001.

[VPKL08] Alex Varshavsky, Denis Pankratov, John Krumm, and Eyal De Lara. Calibree: Calibration-free Localization using Relative Distance Estimations. In *Proceedings of the 6th International Conference on Pervasive Computing*, pages 146–161. Springer, May 2008.

[Wal63] Kenneth T. Wallenius. *Biased Sampling; The Non-Central Hypergeometric Probability Distribution*. Stanford University, CA, Applied Mathematics and Statistics Labs, November 1963.

[ZAZ04] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198, New York, NY, USA, 2004. ACM.

[Zha02] Yilin Zhao. Standardization of mobile phone positioning for 3G systems. *IEEE Communications Magazine*, 40(7):108–116, 2002.