Design Principles for Opportunistic Communication in Constrained Computing Environments

Earl Oliver, Srinivasan Keshav
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
{eaoliver, keshav}@uwaterloo.ca

ABSTRACT

Constrained computing environments, such as smartphones and embedded wireless devices, are becoming increasingly prevalent. Driven by the need to minimize power usage, these devices are characterized by their low-power CPUs, limited memory, slow yet vast amounts of persistent storage, and one or more wireless network interfaces. As the dominant form of future computing, understanding and adapting to the trade offs that exist between computing and communication resources and energy consumption will become increasingly important.

In this paper we consider the effect of a constrained computing environment on opportunistic communication. Drawing from our experiences with two existing mobile systems, we detail the constraints that inhibit opportunistic communication. We also show how these constraints can be satisfied using a set of design principles for systems that depend on opportunistic communication.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; D.1.0 [**General**]

General Terms

Design, Performance

Keywords

Opportunistic communication, constrained computing environment, design principles, mobile, KioskNet, MobiClique

1. INTRODUCTION

Resource constrained devices, such as smartphones and embedded wireless devices, are becoming increasingly prevalent. In 2007, smartphone sales rose 60% to nearly 115 million devices[1]. By 2010, sales are expected to surpass those of laptops[2]. Driven by decreasing hardware costs, small, embedded computers are increasingly found in vehicles, sensor networks, and other elements of daily life. As computing platforms, these devices are characterized both by their mode of operation and physical attributes. They operate in autonomous environments or as personal devices where minimizing heat and/or maximizing battery life is a key concern. To conserve energy these devices utilize low-power CPU(s), which are often underclocked to reduce heat. They typically contain a limited amount of RAM; roughly an order of magnitude less than an average personal computer. Persistent storage is provided by low-power, high-latency hard drives or solid state Flash memory. Short range wireless technologies such as WiFi and Bluetooth have also become standard. As the dominant form of future computing [9], understanding and adapting to the trade offs that exist between computing and communication resources and energy consumption will become increasingly important.

Although today's cellular networks provide nearly ubiquitous data service, devices are increasingly utilizing intermittent, short-lived, wireless *opportunistic* connections to exchange data with fixed infrastructure and other mobile devices. However, the capacity, or throughput, of these connections is handicapped by the *constrained computing environments* where they take place. In this paper we consider the effect of a constrained computing environment on opportunistic communication. Moreover, we argue that the conventional system model, where applications operate oblivious to their connectivity state, is inefficient in an opportunistic communication setting. Based on our experiences, we propose a set of design principles for systems that utilize opportunistic communication.

We derive our design principles from our experience with two real-world systems. The first system, KioskNet, uses opportunistic connections between a kiosk-based computer and an embedded device in a vehicle to transport data to and from kiosks in rural areas of developing regions to gateways on the Internet [3].The second system, MobiClique, exploits human mobility to provide decentralized store-and-forward communication between mobile devices [14, 6]. Although these systems solve different problems and cater to very different types of users, we believe that they encapsulate the problems that other systems would face in that their effectiveness is directly dependent on their ability to maximize communication during opportunistic connections.

This paper begins with a brief overview of KioskNet and MobiClique in Section 2. In Section 3 we define our system

[1] http://tinyurl.com/4l3f3z

[2] http://tinyurl.com/2dsnmn

WINS-DR'08, September 19, 2008, San Francisco, California, USA.

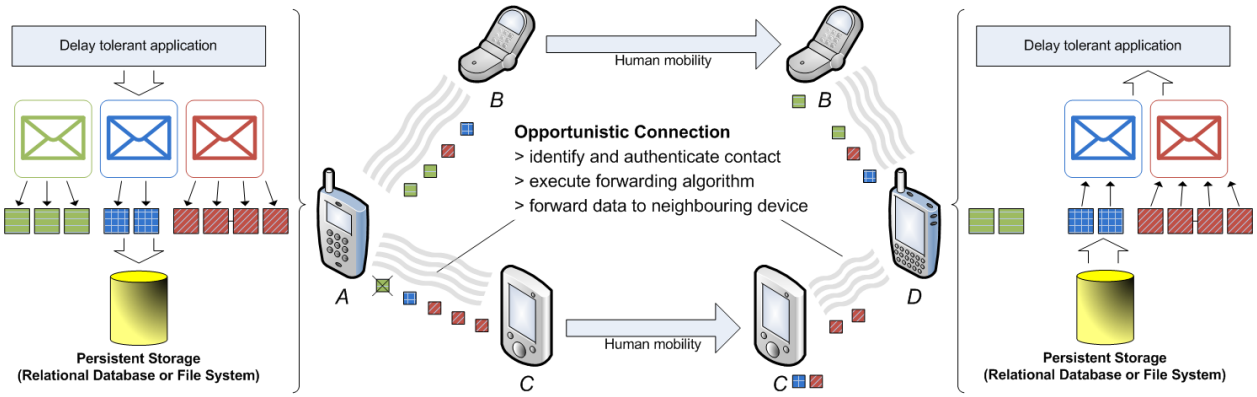

Figure 1: Example scenario: Mobile devices operating in a pocket-switched network.

model and outline the set of system constraints that adversely affect opportunistic communication. Based on our experiences, we define a set of design principles for opportunistic communication in Section 4. We conclude in Section 5.

2. OVERVIEW OF EXISTING SYSTEMS

We first give a brief overview of two existing systems that take advantage of opportunistic wireless communications to exchange data amongst wireless-enabled devices.

2.1 KioskNet

KioskNet is a mobile system that provides very low-cost Internet to developing regions [3]. Building on the pioneering lead of Daknet [11], KioskNet utilizes buses and cars as "mechanical backhaul" devices to carry data between rural village kiosks and Internet *gateways*. A village kiosk consists of a kiosk *controller*, a low-cost, low-power embedded computer that can be powered from an independent power source such as a solar panel. Villagers access the kiosk using a recycled PC connected to the controller. Data created by users, such as emails, information requests, videos, etc., is fragmented and stored on the controller as self identifying *bundles*[3]. A controller is assumed to have WiFi, and possibly a cellular or dial-up connection. Although controllers can communicate with the Internet using a variety of connectivity options, mechanical backhaul is the primary mode of communication. Mechanical backhaul is provided by cars, buses, motorcycles, and trains that pass by a kiosk and also an Internet gateway. Such entities are known as *ferries*.

During an opportunistic connection with a controller, which may last from 20 seconds to 5 minutes, bundles are transferred from the controller to the ferry. Similarly, bundles destined for a kiosk user is transferred from the ferry to controller. To minimize redundant bundle transfers during an opportunistic connection, each connection begins with a metadata exchange where each component exchanges a list of its existing bundles. Only bundles that have not present on the other component are transferred. Each component maintains a record of how many times each bundle was forwarded, which ensures that precedence is given to bundles that have been forwarded the fewest times.

[3]Terminology borrowed from the delay-tolerant networking community (http://www.dtnrg.org).

Similarly, ferries upload and download bundles opportunistically to and from an Internet gateway, which is a computer that has a WiFi interface, storage, and an always-on connection to the Internet. The gateways are likely to be present in cities having DSL or cable broadband Internet access. A gateway collects bundles opportunistically from ferries and stages them in local storage before uploading to a server on the Internet. It also downloads bundles from the Internet on behalf of kiosk users, and transfers them opportunistically to the appropriate ferry, governed by a routing protocol [4].

2.2 MobiClique

As a form of pocket-switched network [6], MobiClique exploits natural human mobility and opportunistic wireless connections to disseminate data from device to device across an otherwise disconnected network [14]. Unlike KioskNet, where data flows to and from rural kiosks and Internet gateway, each device running MobiClique is both a source and destination for data content. The system depends on intermediate devices to ferry data between source and destination. An example scenario is illustrated in Figure 1. Delay tolerant applications such as email pass data into MobiClique as opaque objects. The data is subsequently fragmented into smaller bundles and stored in persistent storage (a relational database). At the receiving device, bundles are stored, reassembled, and passed to the receiver's instance of the application.

Devices running MobiClique operate by continuously scanning for neighbouring devices through a combination of Bluetooth scans and beacons sent over WiFi. When another MobiClique device is detected, an opportunistic connection is established. During an opportunistic connection, each device exchanges a set of identifying/authenticating information and metadata about the user. In MobiClique, this metadata consists of the user's social profile: their friends and their interests. Each device uses this metadata to query a local relational database that returns a set of bundles that should be forwarded to the neighbouring device. Referring to our example figure, device A would have determined that device B and C were friends or shared interests with device D, and were thus likely to deliver data to D.

Like KioskNet, MobiClique's ability to efficiently disseminate content is dependent on its ability to maximize data transfer during an opportunistic connection.

3. COMPUTING ENVIRONMENT

With an overview of two existing systems as context, we now present a more detailed system description and consider the constraints that inhibit opportunistic communication.

3.1 System Model

Opportunistic communication is typically provided at the OSI network session layer. Applications access this layer through a conventional set of APIs for sending and receiving data. Data received from applications is fragmented into bundles. These bundles are then stored in persistent storage to provide robustness to power failure during periods of long disconnection. Bundles received by other devices are also stored in persistent storage. When all of the bundles composing a data item are received, the data is passed to the application layer.

Associated with each bundle is a set of metadata. Metadata is application dependent; however, it typically consists of the source and destination of the bundle, a creation timestamp that is used to expire bundles, a globally unique identifier, and value identifying its structure or purpose.

We make three fundamental assumptions regarding the data and metadata handled by this layer.

- **Metadata fits in memory**: Although pathological metadata schemes can be defined that consume vast amounts of memory, we assume that metadata can always fit into main memory. This assumption is easy to satisfy even in low memory environments; because by doubling the size of a data bundle, we can usually reduce the total metadata size by half. Metadata per bundle is about 100 bytes at most.

- **Application data bundles cannot fit in memory**: For these systems to scale to a large number of users, they must accommodate large amounts of data for a wide range of users. For example, two hours of music recorded at 192 kbps (typical) occupies about 170 MB. Twenty minute videos can grow from 20 MB to several hundred MBs. We therefore assume that the amount of data contained on any device for the purpose of forwarding exceeds the amount of main memory and must reside on persistent storage.

- **Every opportunistic connection needs different data**: To disseminate large amounts of data, devices cannot transfer the same bundles during each connection. For example in KioskNet, connections between ferry and rural kiosk must exchange bundles that have been forwarded the fewest number of times. In MobiClique, the bundles exchanged during a connection dependent on the identity and social relationship with the neighbouring device. Each opportunistic connection must therefore identify the neighbouring device and making a subsequent bundle selection decision.

The primary function of the opportunistic communication layer is to establish and participate in opportunistic connections with other devices. These connections are generally defined by the following four phases:

- **Scanning phase**: Each device continuously scans for neighbouring contacts. The scanning interval may be static or dynamic depending on the user's previous mobility and current context. Scanning may be performed passively by listening for beacons or actively by probing for neighbours.

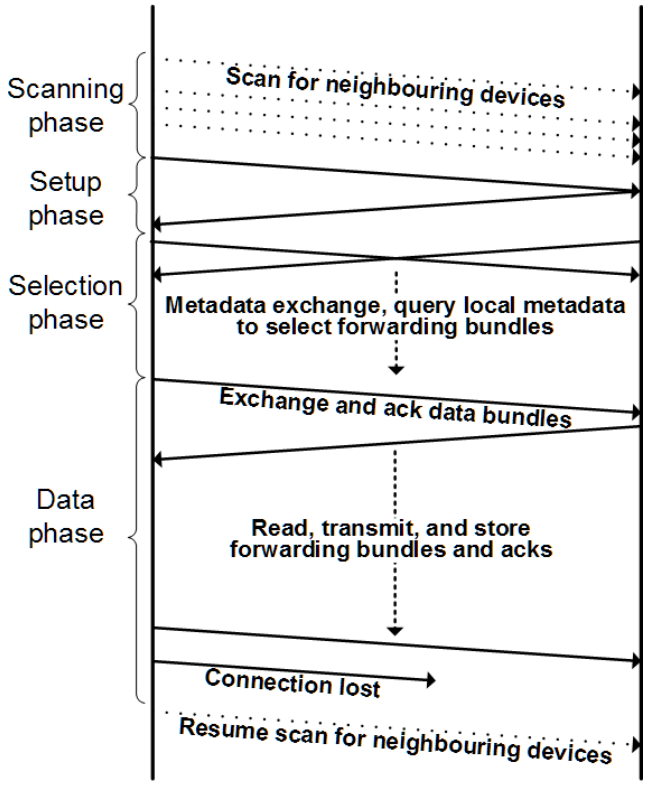

Figure 2: Phases of an opportunistic connection.

- **Setup phase**: Upon detecting and connecting to an opportunistic contact, each device must first identify the contact, and in some cases, authenticate and setup a secure connection.

- **Selection phase**: After establishing a connection with the neighbouring contact, each device must select a subset of its local bundles to forward. In sparse networks, where opportunistic connections are infrequent, this selection may be simple. For example, in KioskNet, bundles travel either upstream to the Internet, or downstream to a rural kiosk. The selection phase therefore only considers the class of contact (i.e. kiosk or gateway) and selects bundles in the order of least times sent. In dense networks, where devices are in frequent contact, more complex queries are necessary. The selection phase in MobiClique selects forwarding bundles based on their destination and social relationship with the contact. To prevent redundant bundle transfers, MobiClique maintains a record of each bundle successfully transferred to each device in persistent storage. Experimentation has shown that the resulting selection phase can quickly grow to the order of tens of seconds because of the need to access persistent storage [14].

- **Data phase**: After selecting a set of bundles to be forwarded, each device begins the forwarding process. Ideally, this phase should occupy the largest portion of the opportunistic connection. During this phase, bundles are read from persistent storage, transmitted

to the neighbouring device, and stored persistently on the receiver. At this time, the receiver either explicitly or implicitly acknowledges (depending on the transport protocol) the receipt of each data bundle. The data phase terminates when both parties have finished transmitting data or when the connection is disconnected (either intentionally or due to signal loss).

The four phases of an opportunistic connection are illustrated in Figure 2.

Scanning and setup phases have been extensively studied in the literature [16, 5]. Scanning more aggressively reduces the expected delay in detecting a nearby device, at a cost of increased energy consumption. The trade off between scanning periods and missed opportunities is detailed in [16]. Similarly, the setup phase offers little opportunity for improvement and has been studied in great detail [5]. In existing systems, contacts are identified by their MAC address or an exchange of globally unique identifiers. Techniques for authentication and setup of a secure channel range in complexity; however, these are typically CPU-bound operations with a constant running time.

We therefore focus, for the remainder of this paper, on minimizing the duration of the selection phase, and maximizing the quantity of data transferred during the data phase. Unfortunately, these phases are inhibited by a series of system constraints.

3.2 System Constraints

Opportunistic communication is inhibited by the following constraints.

- **Need to limit energy consumption**: The need to limit energy consumption is a constraint that fundamentally defines constrained computing environments. On mobile platforms, conserving battery life requires low-power and sometimes underclocked CPUs, wireless technologies that cannot operate at full data rates and frequently enter power saving modes [1], and low-power, high-latency forms of persistent storage. These hardware requirements also exist in embedded environments that depend on passive cooling and must minimize heat.

  This constraint also affects decisions made when implementing an opportunistic communication system. As mentioned above, the chosen scanning frequency has a direct effect on battery life. Excessive computation causes the CPU to be active when it may otherwise be off. Finally, the choice of wireless interface(s) can significantly affect power consumption [12, 13].

- **Underpowered CPU**: As discussed, mobile devices are commonly designed with an underpowered and underclocked CPU to save energy and reduce heat. Other devices, such as Apple's iPhone [8], contain ample CPU power; however, the CPUs are disabled or placed in a low-power mode when the user is not interacting with the device. Underpowered CPUs manifest themselves in three phases of an opportunistic connection. (1) For systems that must authenticate contacts or establish a secure connection, an underpowered CPU increases the duration of the setup phase. (2) In the selection phase, each device must execute a forwarding algorithm to select a subset of bundles to forward. In KioskNet and MobiClique, this selection is computationally simple; however, other forwarding schemes have been proposed that demand significant computation [17]. (3) Finally, in both the selection and data phases, underpowered CPUs have been shown to be the primary bottleneck when performing wireless network I/O, and significantly reduces throughput during an opportunistic connection [10]. Measurements in KioskNet show that an underpowered CPU can reduce the maximum throughput of a 54 Mbps capable WiFi card to 12 Mbps or less.

- **Poor and intermittent communication**: The mobility of devices operating in an opportunistic setting introduces inherent communication problems. Devices may be detected on the fringes of communication range, which causes subsequent communication to be unreliable and problematic; particularly for control packets that initiate the connection [5, 18].

  Moreover, devices that are in communication range will eventually move out of range. Depending on the wireless technology used, communication interrupts may cause random backoffs, inordinately long delays, and failures that can hinder other connection opportunities. Devices need to somehow determine that the communication opportunity has ended, and cleanly close open descriptors. Moreover, they need to deal with cases where the opportunistic connection resumes after a brief pause.

- **Slow persistent storage**: Low-power persistent storage takes two forms: disk drives that operate with a low disk RPM and spin down when not used, and Flash memory. Slow persistent storage affects both the selection and data phases. Slow (random) I/O causes the retrieval of metadata to take order of magnitude longer than if the metadata was in memory. Slow I/O also serves to reduce the overall throughput of a connection. We have found that reading and writing to slow persistent storage (hard disk and Flash memory) during an opportunistic connection reduces wireless throughput by approximately a factor of two [10].

- **Limited RAM**: Despite the continuous decrease in memory costs, RAM continues to be a highly constrained resource on mobile devices. Constraints due to limited RAM manifest themselves in two forms and inhibit both the selection and data phases. First, lack of RAM prevents applications from storing application data in memory and thus requires accessing persistent storage during an opportunistic connection. Second, querying, reading, and transmitting data during an opportunistic connection typically results in a rapid series of memory allocation requests that often consume all of the available memory and cause virtual memory swaps to slow persistent storage. Subsequent allocation requests during the opportunistic connection must block waiting for the virtual memory swap to finish.

  The problems stemming from limited memory are magnified by virtual machines such as Java and .NET, which are becoming increasingly common on mobile devices. Both Java and .NET rely on garbage collection to reclaim memory. Garbage collection is known

Constraint	Design principle
Need to limit energy consumption	Minimize redundant wireless data transfer
Underpowered CPU	Explicitly distinguish between periods of connection and disconnection
Poor and intermittent communication	Use hysteresis
Slow persistent storage	Cache metadata
Limited RAM	Maximize the use of available memory and adapt to low memory conditions

Table 1: Design principle satisfying each system constraint.

to be computationally expensive [19]. Although garbage collection can be performed eagerly during periods of inactivity, it is typically performed lazily and reclaimed on demand. Performing garbage collection during an opportunistic connection can significantly reduce throughput.

4. DESIGN PRINCIPLES

With an understanding of the parameters that inhibit opportunistic communication, we now present a set of design principles. The relationship between system constraints and design principles is summarized in Table 1.

- **Cache metadata**

  Storing metadata in main memory rather than persistent storage provides a quick remedy to the effect of slow persistent I/O. Measurements in KioskNet show that even an in-memory naïve metadata implementation can reduce the duration of the selection phase by approximately a factor of five. Unfortunately, while we assume that metadata can always fit in memory, we cannot guarantee that memory will not be consumed by other applications or persist after a device reset. Applications should therefore treat in-memory metadata as a cache that provides hints during the selection phase [15]. An invalid hint may have the side effect of forwarding redundant data; however, successful hints will significantly increase performance by avoiding persistent I/O.

- **Maximize the use of available memory and adapt to low memory conditions**

  This principle follows naturally from the need to cache and minimize interaction with slow persistent storage during an opportunistic connection. The ability to sample the level of available memory exists in nearly all smartphone and embedded OSes. Applications should proactively populate their in-memory cache when memory is unused.

  While opportunistic communication systems should consume as much main memory as available, they must be equally ready to release it back to the underlying operating system. In virtual memory environments, failure to release memory back to the OS will eventually result in frequent page swaps. In environments without virtual memory, memory allocations will either fail or emergency garbage collection will take place[4]. We have found that low memory problems frequently occur during opportunistic connections when large amounts of memory are requested.

  On recent smartphone platforms, including BlackBerry [7] and Windows Mobile [2], applications can register for low memory events from underlying OS and proactively release memory. While every non-trivial application should utilize these OS functions, their use is critical in an opportunistic communication setting.

- **Minimize redundant wireless data transfer**

  The most effective means of minimizing redundant data transfer is to avoid retransmitting bundles. Although this principle may seem obvious, its design and implementation is non-trivial and must be considered in the early stages of software design. In KioskNet, redundant data transfer is reduced through the use of a metadata exchange. Each connected component exchanges a list of bundle IDs identifying the bundles that it contains. Each component then requests only bundles that it does not have. In our current implementation, this exchange causes the selection phase to last approximately 30 seconds for a 100 MB workload and nearly zero bundles are redundantly transferred. MobiClique also minimizes redundant transfers without a metadata exchange; the selection phase consists purely of a complex database query and no network I/O. Each MobiClique device maintains a record of which bundles have been successfully delivered to each device; only unsent bundles are forwarded. This operation requires a four-way database join and can take over 30 seconds to select less than 5 MB worth of bundles. Clearly, the design of this optimization can have a significant impact on the performance of the system.

- **Explicitly distinguish between periods of connection and disconnection**

  Exploiting periods of disconnection is an unexplored topic in opportunistic communication. We strongly believe that exploiting this fundamental property can yield substantial performance gains. Although none of our systems exploit this property to its full potential, we foresee using disconnection periods to (1) refresh the metadata cache, (2) compress/decompress data bundles, (3) pre-compute forwarding strategies, and (4) performing explicit garbage collection.

- **Use hysteresis**

  Poor and intermittent communication can cause connections to frequently come and go. Reacting to lost and re-established connections requires each device to

[4]The default behaviour in most Java VMs is to wait until free memory drops below a predefined threshold before triggering garbage collection.

enter the setup and selection phases before it can resume sending data. A system that uses opportunistic communication should not immediately react to the loss of a connection; the connection may resume after a short delay. When a connection is thought to be lost, we have found that waiting 10 to 20 seconds before declaring it closed works well.

5. CONCLUSION

Drawing from our experiences with two existing systems, KioskNet and MobiClique, we have discussed how the dominant form of future computing both supports and inhibits opportunistic communication. By examining each constraint in a mobile device and its effect on opportunistic communication, we have enumerated a set of design principles. Although these principles may seem like common sense, they are not obvious when initially designing a system. We have learned them only by examining the flaws in our own work and have re-implemented parts of both systems to correct these problems.

In summary, we believe that developing applications in a resource constrained environment requires careful consideration of how and when memory is used and storage is accessed, when a connection should be made and eventually closed, and most importantly, how energy consumption can be minimized; developing in these environments requires a sense of minimalism. These issues rarely concern developers in a PC or server development environment and merit further study.

6. ACKNOWLEDGEMENTS

We would like to thank Nilam Kaushik, Aaditeshwar Seth, and Hossein Falaki for their comments and suggestions on this paper.

This research was supported by grants from the National Science and Engineering Council of Canada, the Canada Research Chair Program, Cisco Research, and Nokia Research.

7. REFERENCES

[1] C. Andren, T. Bozych, B. Rood, and D. Schultz. Prism power management modes. *Application Note*, February 1997.

[2] Microsoft Corporation. Available from: `http://www.microsoft.com`.

[3] S. Guo, M.H. Falaki, E. Oliver, S. Ur Rahman, A. Seth, M. Zaharia, U. Ismail, and S. Keshav. Design and implementation of the kiosknet system. In *Proceedings of the International Conference on Information and Communication Technologies and Development (ICTD 2007)*, pages 300–309, 2007.

[4] Shimin Guo and Srinivasan Keshav. Fair and efficient scheduling in data ferrying networks. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2007. ACM.

[5] David Hadaller, Srinivasan Keshav, Tim Brecht, and Shubham Agarwal. Vehicular opportunistic communication under the microscope. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 206–219, New York, NY, USA, 2007. ACM.

[6] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 244–251, New York, NY, USA, 2005. ACM.

[7] Research in Motion Limited. Available from: `http://www.rim.com`.

[8] Apple Inc. Available from: `http://www.apple.com`.

[9] S. Keshav. Why cell phones will dominate the future internet. *SIGCOMM Computing Communications Review*, 35(2):83–86, 2005.

[10] E. Oliver and H. Falaki. Performance evaluation and analysis of delay tolerant networking. In *MobiEval '07: Proceedings of the 1st international workshop on System evaluation for mobile platforms*, pages 1–6, New York, NY, USA, 2007. ACM Press.

[11] A. Pentland, R. Fletcher, and A. Hasson. Daknet: rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.

[12] T. Pering, Y. Agarwal, R. Gupta, and R. Want. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232, New York, NY, USA, 2006. ACM.

[13] A-K. Pietilainen and C. Diot. Connectivity management for opportunistic communications in psn. Technical Report CR-PRL-2008-03-0001, Thomson Paris, 2008.

[14] A-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, J. Crowcroft, and C. Diot. Experiments in mobile social networking. Technical Report CR-PRL-2008-02-0003, Thomson, February 2008.

[15] D.B. Terry. Caching hints in distributed systems. *IEEE Trans. Softw. Eng.*, 13(1):48–54, 1987.

[16] W. Wang, V. Srinivasan, and M. Motani. Adaptive contact probing mechanisms for delay tolerant applications. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 230–241, New York, NY, USA, 2007. ACM.

[17] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure-coding based routing for opportunistic networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 229–236, New York, NY, USA, 2005. ACM.

[18] Zhenyun Zhuang, Tae-Young Chang, Raghupathy Sivakumar, and Aravind Velayutham. A 3: application-aware acceleration for wireless data networks. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 194–205, New York, NY, USA, 2006. ACM.

[19] B. Zorn. *Comparative performance evaluation of garbage collection algorithms*. PhD thesis, University of California, Berkeley, 1989.