

Design Principles for Robust Opportunistic Communication

S. Keshav

David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, Ontario, Canada
keshav@uwaterloo.ca

ABSTRACT

Several researchers have proposed the use of transient communication opportunities, that is, *opportunistic* communication, to reduce the cost of rural communication. We propose some design principles for robust opportunistic communication drawing from our experiences in developing and deploying several practical systems. We conclude with a sketch of some areas for future research.

1. INTRODUCTION

Rural communication using technologies such as dialup, VSAT, and long-range wireless networking tend to be low-capacity, unreliable, and expensive. The seminal DakNet approach [11] broke with tradition by using transport vehicles to carry data. This approach has been explored by many others, and has become increasingly relevant to the field with the proliferation of small, highly-functional wireless devices such as smartphones.

Such devices can establish an opportunistic wireless connection (between two devices, or between a device and a wireless access point) that lasts from a few tens of seconds to a few minutes. Surprisingly, even these short connection durations, if properly exploited, allow transfers of several tens of megabytes among the communicating parties [5]. This can enable innovative and useful applications, such as the distribution of audio and video content amongst people as they go about their daily routine—in effect, turning everyday social interactions into a low-cost and efficient content distribution network [13].

Although the potential for opportunistic communication has been known for a few years, building robust systems for opportunistic communication is surprisingly difficult. A system can be said to be robust to a fault if it can carry out its desired functionality, albeit with reduced performance, despite the fault. Robust system design, therefore, requires us to catalog a set of potential faults, and then prove, either by analysis or by actual test, that the system is robust to the fault. The designer of a robust communication system must anticipate and deal with many potential faults. The problem here is that, without actual deployment in the field, the set of possible faults is nearly impossible to determine.

Over the past four years, we have built a series of systems

for opportunistic communication that we have deployed in the field and tested under realistic conditions. Based on our experience, we have identified several failure modes for opportunistic communication. These allow us to prescribe some principles for designing robust opportunistic communications, which are the primary focus of this paper.

The paper is laid out as follows. Section 2 presents some motivating examples of applications that exploit opportunistic communication. Section 3 presents the requirements for a practical system for opportunistic communication. Section 4 describes the design principles in some detail, as exemplified by the Opportunistic Communication Management Protocol (OCMP). Finally, Section 6 presents our conclusions and directions for future work.

2. SOME APPLICATIONS

In this section, we present three examples of applications that can use opportunistic communication.

Our prime motivation is the use of opportunistic communication for rural communications. In this scenario, vehicles equipped with a wireless router and a small on-board computer exchange data with desktop computers in village kiosks [11, 16]. Packets transferred from the desktop to the vehicle are physically carried to other computers that provide a gateway into the Internet. The first such system was DakNet, which was built at MIT in in 2001 [11]. More recently, the VLink system from the University of Waterloo provides a similar service [19]. The use of opportunistic communication allows low-cost communication in areas where it is economically infeasible to deploy infrastructure. Even where such infrastructure exists, opportunistic communication augments traditional communication paths with a low-cost alternative.

The principles of opportunistic communication embodied in DakNet can be used even in the developed world. Consider a mobile device that has an on-board camera. Suppose that a user can take short videos using this camera and store it locally. Also assume that the device has a WiFi interface. The device could use this to opportunistically distribute content to similar devices. Specifically, each video clip could be annotated with a set of descriptive tags. On meeting another mobile, the devices could exchange tags of locally stored

content. If one device wanted to obtain content matching certain tags, it could request it from the other. Clearly, the design of this application requires opportunistic communication. This example can be extended to allow flooding of ‘want’ lists to a set of nodes, and when a ‘want’ request can be satisfied, the routing of content back to the requester along an opportunistic path. It can also be extended to allow a set of nodes to maintain a completely distributed and self-indexed content database, similar to the Huggle project [13, 12].

A final example of opportunistic communication is to allow access to the Internet from moving vehicles [10]. Such vehicles could gain access to the Internet from roadside wireless access points. Symmetrically, content created by the user of a mobile device could be placed into the infrastructure. By avoiding onerous charges for data transfer imposed by cellular providers, this mode of transfer makes it possible for device users to access rich multimedia content at low cost. This communication is opportunistic because vehicles lose connectivity as they move past the access point.

In these examples the use of opportunistic communication enables applications that could not otherwise be implemented. Therefore, we believe that there is a need to support robust opportunistic communications for such future applications, both in the rural context, and elsewhere.

3. REQUIREMENTS

In this section, we describe the requirements for any system that provides opportunistic communication. These requirements are derived from a careful consideration of the system support needed by the three applications presented in Section 2.

We will assume, as a necessary prerequisite, that the applications are tolerant to both delay and delay variance. Otherwise, the applications would not be suitable for opportunistic communication in the first place. This is an important point: opportunistic communication is not feasible for all applications. It does not permit real-time communication or even interactive communication. It is best suited for moving large amounts of data where cost is a constraint and delay is not.

Here are the requirements for opportunistic communication:

- *Should not require human intervention:* to be useful, opportunistic communication should not require active participation of users who are likely to be engaged in other activities.
- *Should recover from disconnections:* Opportunistic communication necessarily implies that disconnections will be common. The system should recover from such disconnections, continuing existing data transfers from the point where they left off.
- *Should be low cost:* the system should make use of unlicensed spectrum when possible, to reduce costs.

- *Should be legacy compatible:* This allows easy deployment in legacy infrastructure.

In addition, we believe there are four secondary requirements which may not apply in all situations: to minimize device power usage, maximize use of communication opportunity, support both single and multi-hop communication, and provide over-the-air security.

These requirements cannot be met using standard TCP/IP. For example, on disconnection TCP goes into a series of progressively longer timeouts, and on reconnection, if the device acquires a new IP address, the communication state is completely lost. Moreover, TCP does not have any notion of communication cost. Therefore, it is as likely to use an expensive WWAN NIC as a free WLAN NIC. Given a choice of access points, TCP does not have any criterion by which to choose one. Of course, these requirements can be met by careful application design. However, this requires every application supporting opportunistic communication to implement the same functionality. Our goal instead is to design a standard set of primitives that can be used by broad class of applications.

It is difficult to meet these requirements primarily because of disconnections, which cause changes to every layer of the protocol stack. For instance, at the link layer, the presence of disconnections makes rapid WiFi association and agile selection of the data rate imperative. At the network layer, the routing protocol needs to take into account the fact that not all links are always available. At the transport layer, reliability cannot be achieved by timeouts and retransmission alone: every packet may need to be replicated for fault-tolerance. Finally, at the application layer, the API should encourage a send-and-wait programming style, rather than the current paradigm of request-reply that arises naturally from the socket API.

3.1 OCMP

These changes and challenges motivate the design of a purpose-built protocol architecture for opportunistic communication called Opportunistic Communication Management Protocol (OCMP) shown in Figure 1. OCMP is similar in spirit to the DTN Reference Implementation architecture [2] and Huggle [13] and is described in detail in [16, 9]. Over the last four years, we have implemented the OCMP architecture several different times. Our first implementation only dealt with disconnections and supported only end-to-end paths [14]. In our second implementation, we added support for hop-by-hop paths, in particular, where this path was provided by a bus or a car that carried a router and provided ‘mechanical backhaul’ [16]. We also added support for security [18]. Our latest implementation, dubbed VLink, supports program state persistence across crashes and COs for USB- and SMS-based paths[19]. In parallel, the architecture has been independently implemented by Bee Networkx Inc., a University of Waterloo spinoff, and is currently being used in the field. Therefore, we have accumulated a wealth

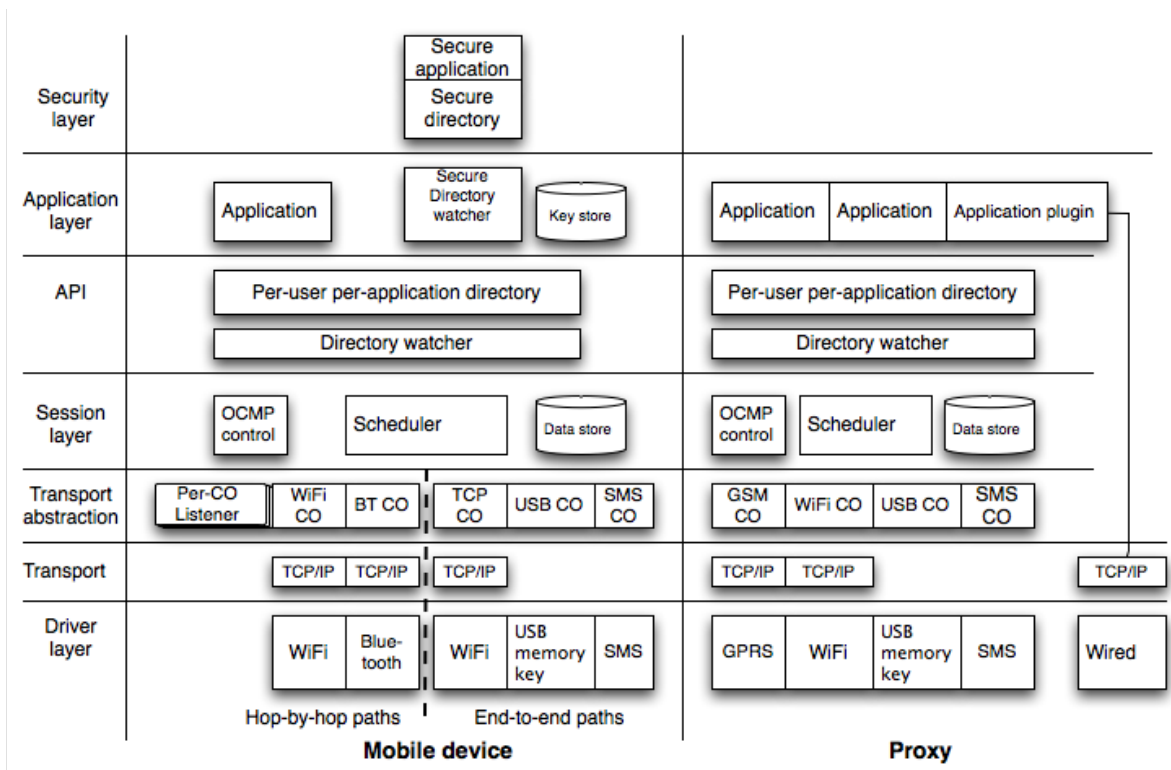


Figure 1: OCMP software architecture

of experience in the use of OCMP in the field. In the next section, we present some design principles for robust opportunistic communication based on this experience.

4. THE DESIGN PRINCIPLES

The principles are categorized according to the corresponding OCMP layer (Figure 1).

4.1 Driver layer

Avoid the wireless fringe.

Opportunistic communication frequently uses wireless links such as WiFi and Bluetooth. These short-range technologies offer good throughput when the received signal to noise ratio is high. However, in their so-called “fringe”, the signal-to-noise ratio is low and frame loss rates are high.

During opportunistic communication, two nodes that are initially out of range of each other come into range, exchange frames, and then move out of range again. Control frames—such as association and authentication request frames—sent when the nodes just enter each other’s communication range are likely to be lost, triggering retransmissions and wasted communication capacity [20, 5]. The problem is exacerbated by the use of fixed multi-second timeouts in most device drivers. These timeouts nearly guarantee the loss of precious seconds of communication opportunity because of lost control frames.

These problems can be avoided by initiating communication only when the the received signal strength is high. Specifi-

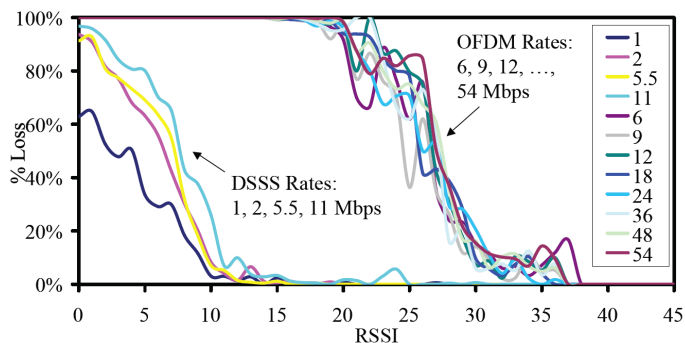


Figure 2: Loss rate as a function of RSSI

cally, we found that loss rates during opportunistic communication decrease dramatically when the signal-to-loss ratio is greater than a threshold (which depends on the data rate), as shown in Figure 2.

The rule, simply, is to initiate transmission only when the signal strength exceeds this threshold. We found that, with this simple rule, throughput achieved during a communication opportunity can nearly double [5]. Note that this rule cannot be used in static scenarios, where the SNR may never increase past the threshold. In such cases, auto rate fall-back is a better solution. But, with opportunistic communication, where mobility is a given, and the SNR is likely to

improve as nodes move into range, deferring transmission in the hopes of encountering better SNR is the right decision.

Avoid performance coupling.

Performance coupling [6] refers to the dramatic decrease in performance across a good-quality wireless link from a mobile device to an access point because of the presence of a nearby mobile device that communicates with the same access point over a poor quality link. The problem arises due to 802.11's automatic selection of data rates. Nodes distant from an access point (AP) use a lower data rate to compensate for a lower signal-to-noise ratio on their communication path. This means that they take longer to transmit a single bit - up to 54 times slower than the best-possible bit transmission time in the case of 802.11g. Consequently, the overall performance of a BSS where some nodes have a lower communication capacity than others is dominated by the slower communication paths. This results in a single 'bad' node reducing performance for all other nodes.

To avoid this problem, we proposed that an AP service only those nodes that have good signal strength. This strategy, called MV-MAX [4], results in considerable performance improvement with only minor changes to the driver layer. The obvious problem with this solution is that it is unfair to those devices that are situated in a region with poor signal-to-noise ratio. However, because devices are mobile, it is likely that over time every device finds itself in a 'good' region. This is particularly true when dealing with vehicular opportunistic communication from a roadside access point, where every vehicle on the road goes through a poor quality region as well as a good-quality region, so that approximate fairness is attained.

4.2 Transport layer

Use hop-by-hop TCP.

In a network with opportunistic communication, end-to-end TCP is not always feasible [3]. Instead, we use hop-by-hop TCP over wireless links. The advantage of using TCP over UDP or raw IP is that provides flow control and reliability, which make it much more likely that transfers from one node to another succeed. Moreover, it exercises a heavily optimized and debugged path, which adds robustness. We did consider using our own implementation of an erasure code over UDP as an alternative to TCP, but the gains from this approach are yet to be proven.

4.3 Transport abstraction layer

The transport abstraction layer wraps an end-to-end or hop-by-hop path over a single NIC with a software Connection Object (CO).

Avoid wireless networks if possible.

Wireless networks, especially when used by mobile nodes, tend to be both unreliable and hard to debug. We found that the same device, moving through the same region of space in the vicinity of an access point, can experience dramatically different performance depending on the presence or absence

of other devices and the unpredictable effects of multi-path interference. It is difficult to build a reliable system when the capacity of the wireless link can vary by nearly two orders of magnitude.

After considerable effort in trying to make wireless links reliable, we ultimately decided to provide, in addition to wireless links, a USB-memory key based communication path, where transfer between nodes occurs by physically carrying a USB-memory key from device to device. This eliminates problems with wireless networks. Because of the clean separation of functionality in our architecture, adding this solution required us to only implement a USB CO. The upper layers are indifferent as to whether communication uses wireless COs or the USB CO. Indeed, the USB memory key can be thought to hold 'frozen' packets, that are 'thawed' at the receiver[19].

This solution is far more cumbersome than one that relies entirely on wireless communication. However, in the context of rural communication, the added delay and loss of efficiency is minimal. Moreover, we no longer need a computer in a vehicle, greatly reducing costs.

4.4 Session layer

Use multi-copy routing.

A natural consequence of multi-hop opportunistic communication is that it is impossible to guarantee that a specific path exists between a source and a destination node. Moreover, node-to-node communication capacity is usually not a constrained resource. Therefore, instead of sending only a single copy of a bundle from a source to a destination, it is better to send multiple copies along more than one path - an approach called multi-copy routing [7]. Note that this does not apply for opportunistic communication between a mobile device and an infrastructure node, such as an access point.

The first version of our system used reverse path forwarding [16]. This is single-copy routing, and turned out to be efficient, but not sufficiently robust. The simplest approach to multi-copy routing is flooding. Here, a node gives a copy of every bundle in its data store to every other node that it meets. By exploring all possible paths in parallel, flooding-based routing is very robust. The current version of our system uses this approach.

The problems with naive flooding is that it is wasteful. To reduce its overhead, either the number of flooded copies, or the time-to-live of a packet can be adjusted [17]. Optimally choosing these parameters depending on the degree of connectivity in the underlying system is an open problem. Nevertheless, the use of multi-copy routing has been widely studied [8] and greatly increases the robustness of the system.

Use death certificates.

When the mobility pattern is unknown, naive flooding, although costly, is often the only robust solution, and indeed, the one we have adopted in our work. When node-to-node

communication is cheap, the primary cost of flooding is in the storage of flooded bundles at intermediate nodes. There needs to be some way to eliminate these stored copies. One way is to remove bundles after some period of time, that is, on a timeout. The problem with this approach is that it requires setting the timeout value, which can be difficult to determine. If storage is not constrained, then choosing a conservative value for the timeout is a reasonable solution. However, there is a better solution.

Instead of timing out flooded packets, the destination of a bundle can acknowledge it by sending ‘death certificates’ [1]. These small bundles are also flooded and result in the removal of the corresponding bundle from the data store of a receiving node. Death certificates elegantly solve the problem of when to remove bundles. Moreover, when a sender receives a death certificate, it also gets a delivery acknowledgement, which is useful.

The use of death certificates leads to the recursive problem of how to remove the death certificates themselves. This is easily solved by keeping the size of a death certificate small and the time-to-live for a death certificate conservatively large.

Give higher priority to less-replicated data items.

When implementing flooding-based routing, we found that the same bundle could be transmitted during several communication opportunities. For instance, the arrival of every bus to a kiosk would result in the same bundle being transferred from the kiosk to the bus and *vice versa*. This is correct behavior, but results in a subtle denial-of-service problem: bundles that have not been sent could be queued behind bundles that have already been sent before, and may not be transmitted before the end of the communication opportunity, leading to persistent starvation.

Our solution is to prioritize those bundles that have been replicated the least. This avoids starvation and makes it more likely that they will make it to the destination. In practice, we store a transmission count with each bundle in the node database. We then query the database (using SQL) for all bundles that have been transmitted zero times, one time, and so on, using the results of these queries to determine the transmission order.

Use databases to store volatile state.

Both mobile devices and computers in developing regions can lose power, due to the battery running down, or grid power outages. On power loss, a typical session layer loses all program state and cannot know which bundles have been sent, how often they have been sent, and which bundles need to be acknowledged. To avoid this problem, the OCMP session layer maintains all data and state in a database, treating in-memory tables as a cache. Specifically, every in-memory table corresponds one-to-one with a database table. Access methods to the tables are then modified so that the in-memory table and the database table are updated simultaneously.

In case the session layer crashes or the mobile device reboots, it refreshes all in-memory from the database. This

makes it robust to power failures. We found this approach to be necessary in dealing with mobile devices and with deployments in developing areas.

This degree of robustness comes at a cost: writes are very expensive. Moreover, the amount of data that can be sent during an opportunistic communication is bandwidth limited by the database access throughput. We believe, nevertheless, that given the deployment environment, this is the right tradeoff to be made. In cases where the node is known to have access to reliable power, we can simply modify the access methods to not make database writes: a minor change in the code.

4.5 API layer

Use directory-based APIs.

Applications communicate with OCMP using a directory-based API. Files that need to reach a particular destination are placed in an ‘upload’ directory along with a metadata file that contains information that would normally be in a packet header, such as the destination, whether the file should be encrypted, and its priority. Any application-specific control information is also placed in the metadata file.

By using a file system as the communication API, application developers do not attempt to use a request-reply paradigm, which does not work in opportunistic communication networks. Instead, they adopt a ‘send-and-wait’ approach, which is more appropriate. We found that these hints to application programmers made it easy for them to write robust programs in a disconnection-tolerant environment.

4.6 Overall

Choose simpler solutions.

The first version of our system used Hierarchical Identity-Based Cryptography, flat names, Distributed Hash Tables, and a very general routing protocol [16, 15]. Through a process of simplification, we replaced these with PKI, hierarchical names, DNS, and bus-and-kiosk-specific routing protocols respectively. This made our solution far more robust, though not so ‘exciting’ from a research perspective. In general, there is a tension between what we call ‘full buzzword compliance’ and building a system that actually works. We have learned to choose the latter.

5. RELATED WORK

The field of low-cost rural communication has grown from the seminal work on DakNet[11]. Instead of surveying this broad field, we focus on protocol architectures most closely related to OCMP.

OCMP is a practical implementation of the concepts first presented by Fall [3]. The work here is most closely related to two other implementations of the DTN concept: the DTN Reference implementation [2] and Haggie [13]. Both these implementations, like OCMP, store bundles in a local data store and offer both single-hop and multi-hop routing between nodes. They also incorporate ways to encapsulate

different connection paths and offer support for experimenting with different routing and scheduling strategies.

Our work differs from the DTN Reference implementation in three significant ways. First, we bind connection objects to paths. In contrast, the equivalent to a CO, called a ‘convergence layer’, is bound to a protocol type, such as TCP or UDP. This prevents fine-grained control over scheduling and routing policies. Second, the reference implementation uses a socket-like API, instead of the directory-based API. Finally, the security model for the reference implementation offers the equivalent of link-level encryption unlike the seamless end-to-end encryption provided by our architecture. Moreover, it does not support a disconnection-tolerant mechanism for key distribution and management. Nevertheless, many of our ideas are derived from our long experience with using the DTN reference implementation and we owe it many insights.

Our work also differs from Haggles in some critical aspects. Haggles uses a non-layered architecture, where different agents collaborate with each other to accomplish the forwarding task. In contrast, our approach uses traditional layering. Second, Haggles only supports multi-hop paths between devices and has no support for proxies or the Internet infrastructure. Finally, Haggles manages all data on behalf of applications: applications never own data. Again, we take a more traditional approach, where OCMP is responsible only for data transfer, not for application-level data management.

6. DISCUSSION

Robust opportunistic communication is essential for rural communications. Unfortunately, it requires careful attention to every layer of the protocol stack, as exemplified by our experiences with OCMP.

There are still many open problems in designing, implementing, and deploying systems based on opportunistic communication. At the link layer, there is a need for agile data-rate selection algorithms that can exploit the predictable increase and decrease in signal-to-noise ratio that is characteristic of opportunistic communication. At the transport layer, there is a need for adaptive erasure codes that are optimized for opportunistic communication. The interaction of erasure codes with routing and flow control algorithms is likely to be both a challenging and an fruitful area of research. At the session layer, a critical problem is to develop optimal algorithms for multi-copy routing over temporal graphs, that is, graphs whose topologies are time-dependant. Finally, at the application layer, there is the need for non-trivial applications that can exploit opportunistic communication in the rural context.

One non-technical aspect of this work worth mentioning is that the barriers to adoption of any technology in rural areas are formidable. Although robust opportunistic communication may not solve these problems, *lack* of robust communication will certainly hinder adoption. It is in this spirit that we hope share our experiences in the design of robust

opportunistic communication for use by other researchers in the field.

Acknowledgments

Figure 2 is due to David Hadaller and is from his unpublished research. This work draws from the talents of a host of graduate and undergraduate students at the University of Waterloo. In particular, I would like to thank Aaditeshwar Seth, Darcy Kroeker, David Hadaller, Matei Zaharia, Shimin Guo, Hossein Falaki, Usman Ismail, Earl Oliver and Mohammad Derakhshani for their many contributions over the years.

7. REFERENCES

- [1] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. PODC*, 1987.
- [2] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, and R. Patra. Implementing delay tolerant networking. *Intel Research, Berkeley, Technical Report, IRB-TR-04-020*, 2004.
- [3] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. SIGCOMM*, 2003.
- [4] D. Hadaller, S. Keshav, and T. Brecht. MV-MAX: Improving Wireless Infrastructure Access for Multi-Vehicular Communication. In *Proc. CHANTS*, 2006.
- [5] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular opportunistic communication under the microscope. *Proc. MOBISYS*, 2007.
- [6] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *Proc. IEEE INFOCOM*, 2003.
- [7] S. Jain, M. Demmer, R. Patra, and K. Fall. Using redundancy to cope with failures in a delay tolerant network. *ACM SIGCOMM CCR*, 35(4):120, 2005.
- [8] E. Jones and P. Ward. Routing strategies for delay-tolerant networks. http://www.ieice.org/explorer/ITC-CSCC2008/pdf/p1577_P2-46.pdf, 2006.
- [9] S. Keshav. Design Principles for Robust Opportunistic Communication. *University of Waterloo, Technical Report, CS-2009-35*, 2009.
- [10] J. Ott and D. Kutscher. A Disconnection-Tolerant Transport for Drive-thru Internet Environments. In *Proceedings of IEEE INFOCOM*, 2005.
- [11] A. Pentland, R. Fletcher, and A. Hasson. Daknet: Rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.
- [12] A. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. MobiClique: Middleware for Mobile Social Networking. In *Proc. SIGCOMM WOSN*, 2009.
- [13] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggles: A Networking Architecture Designed Around Mobile Users. In *Proceeding of WONS*, 2006.
- [14] A. Seth, S. Bhattacharyya, and S. Keshav. Application Support for Opportunistic Communication on Multiple Wireless Networks. In <http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/05/ocmp.pdf>, 2005.
- [15] A. Seth and S. Keshav. Practical Security for Disconnected Nodes. In *Proc. First Workshop on Secure Network Protocols*, 2005.
- [16] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost Communication for Rural Internet Kiosks Using Mechanical Backhaul. In *Proc. MOBICOM*, 2006.
- [17] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proc. WDTN*, 2005.
- [18] S. Ur Rahman, U. Hengartner, I. Ismail, and S. Keshav. Practical Security for Rural Internet Kiosks. In *Proc. NSDR*, 2008.
- [19] VLink project, Tetherless Computing Laboratory, University of Waterloo. <http://blizzard.cs.uwaterloo.ca/vlink>.
- [20] Z. Zhuang, T. Chang, R. Sivakumar, and A. Velayutham. A 3: application-aware acceleration for wireless data networks. In *Proc. MOBICOM*, 2006.