Low-cost Communication for Rural Internet Kiosks Using Mechanical Backhaul

A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav
School of Computer Science, University of Waterloo
Waterloo, ON, Canada, N2L 3G1

ABSTRACT

Rural kiosks in developing countries provide a variety of services such as birth, marriage, and death certificates, electricity bill collection, land records, email services, and consulting on medical and agricultural problems. Fundamental to a kiosk's operation is its connection to the Internet. Network connectivity today is primarily provided by dialup telephone, although Very Small Aperture Terminals (VSAT) or long-distance wireless links are also being deployed. These solutions tend to be both expensive and failure prone. Instead, we propose the use of buses and cars as 'mechanical backhaul' devices to carry data to and from a village and an internet gateway. Building on the pioneering lead of Daknet [15], and extending the Delay Tolerant Networking Research Group architecture [24], we describe a comprehensive solution, encompassing naming, addressing, forwarding, routing, identity management, application support, and security. We believe that this architecture not only meets the top-level goals of low cost and robustness, but also exposes fundamental architectural principles necessary for any such design. We also describe our experiences in implementing a prototype of this architecture.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: *Store and forward networks, Wireless communication*

General Terms

Design, Economics

Keywords

System design, delay tolerant networks, mechanical backhaul, rural communication, low cost

1. INTRODUCTION

Rural kiosks in developing countries provide a variety of services such as birth and death certificates, bill collection, email, land records, and consulting on medical and agricultural problems. They are well-suited to this purpose because both the capital and operational expenses of the kiosk are spread among a fairly large user base, greatly reducing the per-user cost. Even with very low user fees (10-15 cents/transaction), an entrepreneur can make enough money to profitably provide government-to-citizen and financial services.

Kiosks are unproductive without reliable Internet connectivity. Today, kiosks connect primarily using dialup lines, Very Small Aperture (satellite) Terminals, or rarely, long-range WiFi. Unfortunately all three solutions suffer from insuperable problems. Dialup land lines are slow, unreliable, and subject to environmental extremes which degrade their availability. In rural areas, repair delays of three to four days are common. VSAT terminals are reliable, but require considerable up-front capital expenditure as well as expensive monthly fees. Their cost-per-bit is therefore high. Moreover, in case of breakdowns, spare parts are not widely available. Finally, long-range WiFi requires considerable planning for a large scale deployment. Line-of-sight is necessary for most technologies, and a constant power supply is needed at each relay tower. Early adopters such as N-Logue and Drishtee in India report unexpected problems such as a long-range link being overpowered by a laptop near one of the towers. In practice, tower heights of at least 18m have been reported to be necessary, which turn out to be quite expensive [11].

Looking to the future, it is likely that high-speed 3G cell phone technologies such as EvDO will eventually reach rural areas. However, rural areas are usually poor, so it is unlikely that high-speed data services from cellular providers will be offered any time soon.

Given this situation, our interest is in providing low-cost and reliable connectivity to rural kiosks. In this paper, we present an architecture that uses buses and cars ('mechanical backhaul') to ferry data to and from a kiosk, building on the pioneering work of Daknet [15]. This design decision has repercussions on every layer of the protocol stack, and indeed, the entire network architecture. We therefore present principles for naming, addressing, forwarding, and routing that are needed by any system that uses mechanical backhaul. We also report on our experiences with implementing this architecture in the context of the Delay Tolerant Networking architecture proposed by the IRTF DTN Research Group [24]. Currently, we have implemented most aspects of our design (as detailed in Section 14), and we deployed a first prototype in the field in May 2006. The focus of our presentation is on the opportunity, challenges, and architecture, rather than on performance measurements, which we plan to report in future work.

The paper is laid out as follows. We present our goals in Section 2 and outline in Section 3 why we cannot use existing research to address this problem. We then describe in Section 4 the technologies available to reach the goals, and enumerate in Section 5 fundamental principles on which our work is based. We give an overview of our architecture in Section 6, and describe details of location management, routing, security, and protocol design in Sections 7 through 11 respectively. Section 12 outlines an approach to scaling our system. An implementation overview and status is presented in Section 14. Finally, we describe related work in

MobiCom'06, September 23–26, 2006, Los Angeles, California, USA.
Copyright 2006 ACM 1-59593-286-0/06/0009 ...$5.00.

Section 15, and our conclusions and future work in Section 16.

2. GOALS

We first present the goals of our work.

- **Low cost**: To be widely deployed, our system has to be low-cost. We would like to keep the capital cost per kiosk to be under US $250 and the operational costs to be as low as possible. For example, we would like a kiosk supporting a user base of about 500 users to charge no more than US$0.15/month for email service.

- **Reliable**: To be useful, the system has to be reliable, tolerating power outages at kiosks, ferry failures, packet loss, and disk corruption. Moreover, this reliability has to be designed in, because the system will be operated by technically untrained users.

- **High bandwidth**: The system has to be scalable to support applications that require large amounts of data to be transferred to and from the Internet. For example, digital photographs uploaded from a kiosk can be more than 2 MB in size, and cannot be easily uploaded over a dialup connection. Video clips are even larger.

- **Disconnection tolerant**: We require communications to be disconnection tolerant for two reasons. First, this allows the system to work reliably despite power outages, which are endemic in developing countries. Second, it allows us to trade delay for cost. That is, we can reduce costs by tolerating message delays of up to a few days. Because of disconnections, both ends of a transport connection may not be simultaneously present [7], precluding the use of standard TCP/IP to provide end-to-end connectivity.

- **Support both kiosk and laptop/PDA users**: We envision that some users will be using shared PCs in a kiosk to access networked services, while other users may have their own device; the latter users will typically be government or NGO employees who may have a PDA or laptop, or even a cellular smartphone that is out of coverage area in villages. We would like to support both classes of users.

- **Allow user mobility**: Field studies show that many villagers routinely move from village to village within a 15-20km radius [11]. We would like to allow such villagers (or health care workers having laptops/PDAs, who also move from village to village) to access their data from the closest kiosk.

- **Data privacy**: Banking is a critical need in rural areas. Lacking banks, farmers are forced to pay usurious rates to money-lenders. To break this cycle, banking communications, which need to be private and reasonable secure, must be provided. We cannot use a standard solution such as IPSEC because PKI does not work well in disconnected environments [18].

- **Interoperability**: Clearly, applications such as VoIP are incompatible with mechanical backhaul. Nevertheless, to the degree possible, we would like the users to be able to access Internet services on legacy servers *without modification to these servers.*

- **Policy based use of all available networks**: We would like a kiosk to be able to use all channels of communication (including cell phone and dialup) based upon intelligent policies. For example, some applications might require immediate data transfer and dialup may be the only option possible, as opposed to email or land-record applications that are inherently tolerant to delays and more suited to use a mechanical backhaul communication mechanism.

3. DO WE NEED A NEW SOLUTION?

Before describing our solution, it is worth considering if the goals presented in Section 2 can be achieved using existing research. First off, it is clear that a legacy solution, that is, naming nodes with DNS names, addressing them with IP addresses, and using TCP end-to-end, will not work, because the two-ends of a connection, i.e. kiosk users and legacy servers, are never simultaneously present [7]. Moreover, standards such as SSL for security do not cope well with long end-to-end delays. Nevertheless, recent research presents solutions which on surface appear to meet all our goals. For instance, past work has addressed disconnection tolerance [20], support for mobile users [13, 16, 22], interoperability with legacy servers using proxies [14], and use of multiple networks and NICs [5].

However, all these solutions have three problems. First, they are point solutions that do not form a single coherent system. It is not possible to simply mix and match the solutions to achieve our goals. Second, they have been designed in the context of laptops and PDAs that are almost always connected, with short disconnected periods, and, when connected, are connected at relatively high speeds. Finally, they do not focus on low cost and reliability. Introducing these design constraints greatly changes the problem.

This motivates us to seek a custom-built solution to our problem, using, where possible, design principles advocated in existing research. We return to a more detailed evaluation of past work in Section 15.

4. AVAILABLE TECHNOLOGIES

We leverage the following fundamental technologies to meet our goals:

- **Cheap storage**: Storage today is cheap, costing less than US$1/GB, and rapidly getting cheaper. Therefore, we envision several tens of GB of storage at a kiosk, on a bus, and anywhere else needed in the network to store data in transit. To make this storage reliable, we use RAID 1, which is both simple and cost-effective.

- **Wireless networks**: Wireless (802.11x) network cards are ubiquitous, cheap, and, for the most part reliable. Wireless allows us to make opportunistic use of buses and cars that happen to drive past a kiosk, and then exchange data as they drive past a server that has a persistent Internet connection.

- **Delay Tolerant Networking overlay**: The DTN Research Group architecture provides a delay and disruption tolerant bundle-forwarding architecture. At its core, the architecture describes how a set of DTN routers form an overlay to cooperatively store and forward *bundles* of information [7]. DTN routers are connected by links that are sometimes, or often, down, and can be persistent, scheduled, or opportunistic. The DTN architecture is ideally suited to our needs because it supports the opportunistic and scheduled links provided by buses. Although DTN routing schemes are yet to be defined, its naming and addressing conventions are simple and extensible, and the bundle forwarding engines are available as open source. We have therefore built our design as an extension to this architecture.

- **Cellular overlay**: Unlike 3G data services, which are expensive to deploy, GPRS-like data services at low bit rates (4-8 kbps) are nearly ubiquitous worldwide, even in rural areas. It appears to be straightforward to use recycled cell phones as GPRS modems. We therefore seek to exploit this connectivity, where available, to provide a cheap and reliable control plane.

5. PRINCIPLES

Before embarking on any architectural design, it is useful to identify the principles embodied in the architecture. These principles allow us to intelligently navigate the infinite space of possible designs. We believe these principles are applicable to any realistic architecture that uses mechanical backhaul and has goals substantially similar to ours. We hasten to add that we do not claim these principles to be novel; instead, we claim that these are the principles *relevant* to our context.

- **Lowered cost through shared infrastructure**: Low cost can only be achieved by sharing every component of the architecture. Therefore, we need to share not only the Internet gateway, but also the storage on the bus, and every element in the kiosk. Of course, unrestricted sharing can be both insecure and unfair. Therefore, all shared elements need to be appropriately managed.

  Note that the proposed One Laptop Per Child project [25] does not share end-systems. Therefore, we do not believe that this project can achieve the cost targets achievable using shared kiosks.

- **Store and forward of self-describing data**: This is necessary for tolerating disconnections and disruptions. Store and forward allows a node to deal with failed links. Moreover, by making bundles self-describing, in the same way that an email message is self-describing, allows easy recovery from power failures at nodes and bad routing decisions.

- **Decoupling location and addressing**: Because users are mobile, their identifiers must be location-independent. This means that we need some way to map from a user's ID (which is the eventual destination of a data packet) to the ID of his or her current location (i.e. address).

- **Opportunistic link use**: The opportunistic use of links increases sharing of infrastructure nodes such as kiosks and buses and thus reduces cost. Moreover, this principle dictates that we should attempt to use every available NIC at a kiosk based upon intelligent policies. Therefore, kiosks should exploit not only buses, but also dialup links, and VSAT connections, whenever required.

- **Separate data and control plane**: The clean separation of the data and control planes allows us to use almost-always available cellphone links for the control plane, and opportunistic WiFi or Bluetooth links for the data plane. By using costly cell phone links for low-bandwidth communication, we optimize the usage of the data plane. In particular, using the lightweight control plane for routing updates allows us to overcome pathological routing problems that arise when routing or location updates are delayed.

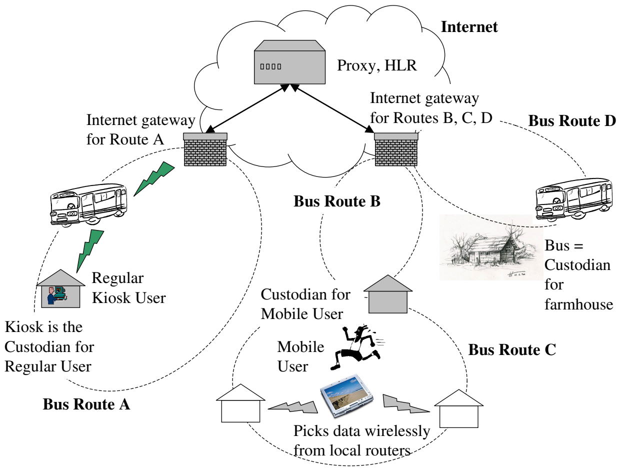
Internet
Proxy, HLR
Internet gateway for Route A
Internet gateway for Routes B, C, D
Bus Route D
Bus Route B
Bus = Custodian for farmhouse
Regular Kiosk User
Kiosk is the Custodian for Regular User
Custodian for Mobile User
Mobile User
Bus Route C
Bus Route A
Picks data wirelessly from local routers

Figure 1: Example topologies

- **Proxies for legacy support**: We use disconnection-aware proxies to allow applications on a kiosk to interoperate with existing Internet applications.

- **Replication for reliability**: We replicate data on disks (RAID 1) and contemplate replicating bundles on the network to increase reliability at the expense of wasted resources. Disk is cheap enough that this waste is unremarkable. However, it is not clear whether, and to what degree, packet replication is needed. Finally, we envisage that ferries can carry a set of spare parts that can be used, as needed, by kiosks along its path.

6. ARCHITECTURE OVERVIEW

This section presents an overview of our architecture outlining the data and control flow in our system.

Kiosks play a central role in our architecture. A kiosk consists of a *kiosk controller*, a server that provides network boot, a network file system, user management, and network connectivity by means of dialup, VSAT, or mechanical backhaul. A kiosk controller is assumed to have a WiFi NIC, and very likely a GPRS/EDGE or dialup connection. Our prototype uses headless and keyboard-less low-power single-board computers from Soekris Corp. as kiosk controllers, although the functionality can be implemented in any commodity PC. Our choice of a Soekris device was due to that fact that it only draws 7W, and can therefore be powered by a battery-backed solar cell. Moreover, the lack of I/O devices discourages tampering.

The kiosk is expected to be used by two types of users, shown in Fig. 1. Most users would use a public access terminal that boots over the network (using RAM disk) from the kiosk controller, and can then access and execute application binaries provided by the kiosk controller over NFS. Recycled PCs that cost around $100 are ideally suited to function as public access terminals, and spare parts are widely available worldwide too. Moreover, as a shared resource, they are an order of magnitude cheaper than any dedicated resource.

A second class of users, such as wealthier villagers, government officials, or NGO partners, could access one or more kiosks, or a bus directly, using their own devices, such as smart phones, PDAs, and laptops. Such users would use the kiosk-controller or bus essentially as a wireless hotspot that provides store-and-forward access to the Internet. Identity management and mobility support for hotspot users is a key requirement supported by our architecture.

Although kiosk controllers can communicate with the Internet using a variety of connectivity options, our focus is on the use of mechanical backhaul. This is provided by cars, buses, motorcycles, trains, and even bullock carts that pass by a kiosk and also an internet gateway (which is described in more detail below). We call such entities *ferries.* A ferry has a small, rechargeable, battery-powered computer with 20-40GB of storage and a WiFi card. It opportunistically communicates with the kiosk controllers and internet gateways on its path. During an opportunistic communication session, which may last from 20 seconds to 5 minutes, anywhere from 100KB-50MB of data can be transferred in each direction. This data is stored and forwarded in the form of self-identifying *bundles.*

Ferries upload and download data 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 data opportunistically from a ferry and stages it in local storage before uploading it to the Internet. It also downloads bundles on behalf of kiosk users, and transfers them opportunistically to the appropriate ferry, governed by a routing protocol. In our implementation, we use a Soekris device both for ferries and for gateways.

We use the term *DTN router* to refer to any device that is connected to more than one other device either on different NICs, or at different points in time. In this sense, the ferry is just a mobile DTN router, and Internet gateways and kiosk controllers are examples of fixed DTN routers.

We expect that most communication between a kiosk user and the Internet would be to use existing services such as Email, financial transactions, and access to back end systems that provide government-to-citizen services, such as land record management, birth and death certificates, and various sorts of licenses. Systems that provide such services are typically unable to deal well with delays and disconnections. Therefore, we propose the use of a disconnection-aware proxy that hides disconnection from legacy servers. The proxy is resident in the Internet and essentially has two halves. One half establishes disconnection-tolerant connection sessions with applications running on a recycled PC or on mobile user's device. The other half communicates with legacy servers. Data forwarding between the two halves is highly application dependent. For example, a proxy that fetches email from a POP server on behalf of a user needs to implement the POP protocol. To support application-specific protocols, we allow applications to instantiate an application-specific plugin at the proxy. Our system can support multiple proxies; for each gateway we use the proxy closest to it in terms of the RTT between them so as to gain maximum TCP throughput. In the rest of the discussion we assume a single proxy to keep the explanations simple.

Finally, the last component of our architecture are the legacy servers that are typically accessed using TCP/IP and an application-layer protocol such as POP, SMTP, or HTTP by a proxy. We do not require any changes to legacy servers.

6.1 Network model

We model the system as shown in Fig. 2. The Internet IP core is a fully connected cloud where all nodes form an overlay clique. The core is connected using low-speed links to the Internet gateways I_1-I_3. Each ferry on a particular route is represented by a ring of nodes, with each node representing a point in the ferry's trajectory where it communicates with a kiosk or user. The weight of an edge on the ring can be used to represent the transit time between contacts. Note that some ferry routes go past multiple Internet gateways, while others go past none.

Kiosks such as K_1 - K_3 hang off the ferry ring. Some kiosks, such as K_1 and K_3 can be used to route bundles from one bus route to another. Finally, users are attached to a single kiosk (e.g. U_3), to multiple kiosks (e.g. U_4), directly to a ferry (e.g.. U_1), or directly to multiple ferries (e.g. U_2). For the purpose of routing, we distinguish between users who always access a ferry at a particular point in its trajectory, such as from a farm house, and users who opportunistically download data from a ferry at some (potentially varying) point in its trajectory. We represent the former as a node along the ferry's trajectory because it is possible to speak meaningfully about paths and edge weights. The latter are not represented in the graph, and, for routing, we treat them as if they are located on the ferry itself. If such a user were to move from one ferry to another, we update their location from one ferry to another.

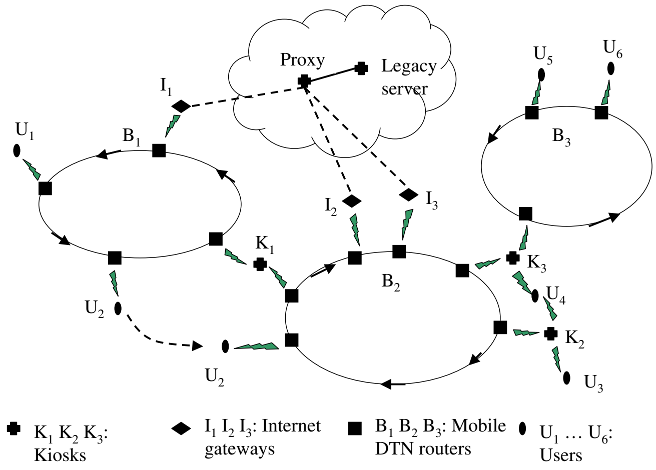

Figure 2: Network model

The performance achievable by this system can be characterized by two metrics: the overall client-to-server throughput achievable, and the mean end-to-end delay. In terms of throughput, the path from a legacy server or proxy to an Internet gateway is highly constrained by the speed of the gateway's access link [15]. This link operates typically at 100 kbps over a DSL connection. Therefore, to maximize performance, we should keep the Internet gateways as busy as possible, balancing load amongst all available gateways.

In terms of end-to-end delay, naturally, ferry transfer latencies can add hours or days to a communication path. Surprisingly, a significant contributor to end-to-end delay is not only the ferry transit time, but also the wait time at an Internet gateway awaiting upload. To see this, note that a bus can pick up 20MB at each kiosk it visits [15], so, if it visits 10 kiosks, which we expect to be the median, it would pick up 200 MB. Of this, perhaps 80% or 160MB would need to be transferred over an Internet link. At 100 kbps, this will take nearly 4 hours. If more than one bus were to share a single gateway, the delay can be substantially larger. So, with a poor choice of routing, a gateway could introduce a few days worth of delay for data arriving on a single ferry contact!

6.2 Protocol architecture and data path

We now trace the flow of data from a client to a legacy server, shown in Fig. 3. The client software application executes either on a mobile device or on a recycled PC, and the subsequent discussion applies equally to both situations.

Applications may either be aware of our architecture, or not. If they are, then the application directly communicates with a Opportunistic Communication Management Protocol (OCMP) [19] daemon. This protocol, implemented

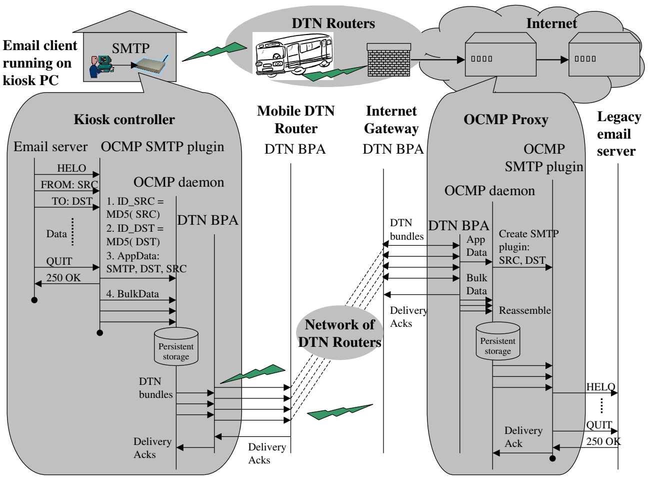

Figure 3: Data path

in Java, maintains a client-to-server disconnection-tolerant session with the help of application-specific plugins running at the client and at the proxy. Besides maintaining a disconnection-tolerant session, OCMP also manages multiple client NICs and provides application-specific ID to end-system ID translation at the proxy (details can be found in Section 7).

In order to support disconnection-tolerant applications on the client that are not aware of OCMP, we run a modified dummy server on the mobile itself, the kiosk-controller, or the ferry. This server pretends to be the legacy server. For instance, this server could be a Jabber server or an email server. On receiving data from the client, the modified server invokes OCMP which encapsulates client data into bundles. This allows us to support legacy client applications with no modifications. Of course, the application needs to be inherently delay tolerant – our approach cannot mask the inherent delays in the communication path.

Once OCMP receives data, it stores it in a local on-disk database. This allows it to gracefully tolerate power disruptions that may bring down the kiosk controller, perhaps as often as several times a day.

When an opportunistic connection presents itself, OCMP hands bundles for transmission to a DTN Bundle Protocol Agent [4], which invokes a routing and flow control protocol (described in Section 11) to decide which bundles to transfer on the opportunistic link. The selected bundles are transported, typically using TCP/IP, to a DTN router on the ferry. We use the standard bundle protocol for this transfer, as described in [4]. When the ferry goes past an Internet gateway, or a kiosk is used to route between ferries, routing and flow control are again invoked to select bundles for transmission to the gateway. These are then transferred to the gateway using the bundle protocol. The destination of the bundles is either one of the proxies (for legacy applications) or the bundle-aware server. The Internet gateway accepts custody of the bundles and schedules them for transmission on the internet link, keeping in mind bundle priority and any other scheduling decisions.

Bundles meant for legacy servers that arrive at a proxy are demultiplexed and handed to the appropriate application-specific plugin. This plugin then invokes a connection to the legacy server over TCP/IP and delivers the data.

Data flowing in the reverse direction is symmetric. The proxy registers itself on behalf of the clients to receive, for example, email destined to them. On receiving data, the application-plugin on the proxy establishes a disconnection-tolerant connection to the client and queues bundles for transmission. These are then delivered to the OCMP stack running either on the client or the kiosk-controller using OCMP layered over DTN, and thence to the client application.

The next few sections expand on this overview. We start with a detailed description of naming, addressing, and location management.

7. NAMING, ADDRESSING, AND LOCATION MANAGEMENT

In our system, a user's human-readable name is his or her telephone number (IMSI) or email address. For uniformity, the system translates both into 20-byte strings with a SHA-1 hash. We call such a string a Globally Unique ID or GUID. Because of its length, we assume that users with distinct names will almost surely map to distinct GUIDs. We also use the same GUIDs for forwarding, thus our system forwards bundles using names rather than addresses.

Creating GUIDs from a hash of a human readable name allows translation from a human-readable name to a fixed-length numeric GUID to be carried out without any additional infrastructure. In contrast, with DNS names, a sender needs to use the DNS service to find its correspondent's IP address, and with HIP, a sender needs to determine the cryptographically signed ID of its correspondent using PKI. This choice of GUIDs is also motivated by our security architecture (Section 9): essentially, the SHA-1 hash of a user's name is also that user's public key [2].

Every user in our system registers itself with at least one DTN router at a kiosk, bus, or Internet gateway, called its *custodian*. A custodian is similar to a mail server in an email system, in that it holds data on behalf of a potentially disconnected user, and that it participates in a routing protocol to forward bundles between users. A custodian can be thought of as a rendezvous point that coordinates a sender and receiver, in the same way as a Foreign Agent in Mobile IP [16], an anchor point in Hierarchical Mobile IP [21], or an I3 server in I3 [22]. Note that from the perspective of the system, as long as bundles are delivered to the custodian, they will be somehow picked up by the user. Custodians play multiple roles in our architecture. They store bundles on behalf of a user to be picked up later. They also act as an anchor point for mobile users and as a gateway to limit the spread of routing updates. A user's custodian must be able to reach that user either directly, i.e. on a one-hop link, or by means of a series of DTN routers that need only the user's GUID to deliver bundles to it. Each custodian keeps track of the user GUIDs that are registered with it, and whether the bundles will be directly picked up by the user or they have to be routed further. If the bundles have to be routed further, then a bit is flipped in the bundle header to indicate that the bundles are to be forwarded to the user rather than to the intermediate custodian.

The full 'address' of a user is the tuple with two names: [custodian GUID, user GUID]. If a sender does not know the custodian GUID, it can use the special form ['unbound', user GUID]. This instructs a DTN router to forward the bundle on its default path to a router that can eventually find the user's custodian and forward bundles to it. To resolve unbound bundles, we assume that a redundant and fault tolerant server on the Internet (which may be implemented as a multi-site cluster or a distributed hash table for scalability) stores a lookup table or registry with a mapping from a user's GUID to its custodian's GUID. Following cellular telephony terminology, we call this lookup table the Home Location Register (HLR). A user can simultaneously register with multiple custodians; in this case, the HLR resolves the user's GUID to the GUIDs of multiple custodians, all of which represent viable routes. A user can also be identified by multiple GUIDs based on different public identifiers. In this case, the HLR maintains records mapping the different GUIDs.

When a user moves, the HLR is updated if and only if the user changes her custodian. Users can potentially send location updates anticipating future movements if they know their eventual destination, so that the data can be forwarded in advance to the closest custodian.

Fig. 1 shows three example cases. In the first case, a static user who visits a single kiosk defines the associated kiosk controller as her custodian and registers with it. The controller then retains all data for the user locally. In the second case, a user who moves between neighboring villages define her custodian as a DTN router higher up in the hierarchy towards the Internet. In this case, the custodian uses a routing protocol (described in Section 8) to direct bundles to the user. In the third case, a mobile DTN router, such as a bus, is defined as the custodian, for example, to deliver data to a rural farmhouse PC that is far from any village kiosk.

7.1 Setting up the HLR

The HLR keeps track of the custodian(s) associated with each user. How should it be kept up to date by a potentially disconnected user? We first describe a solution that assumes that every user is associated with a single custodian.

In case of a user who will always access the system from a single kiosk (the common case), when a user is added to the system, the user's ID is registered with the kiosk controller, which is also its local DTN router. In the case of a mobile user, it opportunistically associates with any available DTN router, which we call its local DTN router.

On association, the local DTN router updates the routing protocol (for instance by creating a Link State Packet) to indicate that it can reach the user. The router also informs the user of its choice of nearby custodians (which may be a nearby kiosk, a custodian in the Internet, or the bus itself). The user chooses one or more custodians and informs the local DTN router of its choice.

At a future time, the DTN router informs the custodian of the user's request. The custodian updates its state to indicate that the user is now registered with it. If the custodian has not seen the user before, then the HLR needs to be updated to reflect the new custodian choice. The custodian updates the HLR by sending a registration message to the HLR. This message is also sent to the user's previous custodian to free up any old bundles: these old bundles are then sent to the new custodian, as described below. As a final step, the old custodian removes any state associated with that user.

If a user has multiple custodians, then the sender, (or, for unbound bundles, the Internet gateway that discovers the set of custodians associated with the user) adds multiple custodian GUIDs in the bundle header of a bundle destined to that user, in addition to the user's GUID. The forwarding engine in the DTN router uses the custodian headers to deliver one copy of the bundle to each custodian named in the header. It is left to an application-layer protocol to delete (or to allow the time out of) bundles that have been delivered to the user, but are still pending delivery at one or more custodians.

7.2 Dealing with race conditions

Mobility intrinsically introduces race conditions. Bundles may be sent to one of the mobile's old custodians or local DTN router before it has heard of the mobile or after it has moved to a new custodian. Because these race conditions are not easily avoidable, we have designed our protocols to work correctly, though with reduced efficiency, in case of races. The basic design principles are to update location information *before* old information is deleted i.e. *make-then-break* and to be generous in accepting bundles. This way, bundles may be sent to an out-of-date location, but are very likely to be eventually delivered to the right destination.

Consider a mobile registered with custodian Old that moves to custodian New. On receiving a registration, custodian New first updates the HLR to point to itself, then tells Old that it is no longer the custodian. Bundles sent to the user after the HLR change will arrive at New correctly. However, bundles in flight may incorrectly arrive at Old. Worse, they may have been forwarded by Old on the path towards the user.

Bundles at Old that have not been picked up before the arrival of the custodian update message are forwarded to New when the update message arrives at Old. To sweep up bundles that have been forwarded to the user from Old, it forwards the custodian update message on the path to the user and every DTN router along the path resends these bundles to New by changing the custodian portion of the address to 'unbound' and resending the bundle. Once this is done, the mobile user's state is removed from Old, and by every DTN router along the path from Old to the user. Subsequently, any bundles arriving to Old for that user will be bundles arriving to an unknown user, which we describe next.

Bundles that arrive to a custodian with a destination GUID that is unknown to that custodian (i.e do not have an entry in that custodian's local state) have arrived before the user has registered with the custodian, or after the custodian has cleaned up that user's state. In this case, the custodian first looks up the HLR to find the new custodian for that user. If the HLR has the new custodian for the user, then the custodian forwards the bundle to the new custodian. If the HLR has no information, then the custodian saves the bundle and awaits a registration until the bundle's time to live expires, with periodic HLR lookups to see if it can be handled by some other custodian.

8. ROUTING

The goal of a routing protocol is to map, at each DTN router, from a destination GUID (of a custodian or user) to the next hop link.

Getting bundles from a user or any DTN router to an Internet-based proxy is straightforward if we use default routing. Similar to the way routers are configured with a default route in the Internet, we also manually configure every DTN router with a default link which is the link on which to send a bundle whose destination custodian is 'unbound'. For example, an end-point's default link would point to one of the ferries, which would have a default route to one of the Internet gateways.

Having a single default route is not fault-tolerant. To deal with failures, unbound bundles can be flooded in the disconnected region. This will lead to the same bundle being received by multiple gateways.Therefore, on receiving an unbound bundle, the gateway looks up the destination's GUID in the HLR to find its current proxy or gateway node, and then conducts a simple handshake protocol among other gateways in the disconnected region to avoid sending duplicate bundles to the same proxy or gateway. For small disconnected regions, we believe that flooding provides sufficient redundancy to achieve reasonably good performance in most cases.

The reverse path (i.e from proxy to kiosk) is much harder to determine. The proxy needs to know the best Internet gateway to use, and the Internet gateway needs to know how to reach the user's custodian. Finally, the custodian needs to know how to reach the user. Unfortunately, general routing in DTN is an unsolved problem. We therefore present three alternative routing protocols.

8.1 Flooding

This is the simplest routing scheme. Here, the proxy floods bundles into the entire DTN network. Reachability is guaranteed, but scalability is a problem, particularly given that the bottleneck link capacity, which is the access link from an Internet gateway, has a capacity of typically around 100kbps, or around 1 GB/day. Opportunistic links may also become a bottleneck because they are likely to have a maximum connection duration of only a few minutes, and flooding may result in excessive duplication of the same data on multiple links. So, flooding is unlikely to be useful in any but the smallest deployments.

8.2 Reverse path forwarding

This scheme is more efficient than flooding, but it gains efficiency at the cost of increased system fragility. In a nutshell, reverse path forwarding uses a locationing update to simultaneously set up forwarding tables along a sink tree, therefore combining locationing and routing. As we shall see, it also requires that custodians be either present at, or be associated with, a single Internet gateway, and, moreover, to be present on the path from the user to the Internet gateway.

In this scheme, the system uses only a single path to every entity in the system – all alternate paths are ignored. Therefore, the problem of reaching a user reduces to finding a unique path from a sender to every custodian for that user, and from every custodian to every user registered with that custodian.

We create paths to custodians by associating a unique Internet gateway with each custodian, and storing the Internet gateway's IP address in the HLR, in the same way as for Internet-accessible custodians. Unbound bundles associated with a particular custodian, therefore, automatically reach the Internet gateway as described earlier. The choice of Internet gateway for each custodian, and setup of a reverse path from the Internet gateway to a custodian and from a custodian to a user, are done using reverse path forwarding as described next.

Specifically, we define a special control message called the REGISTER message. When a mobile registers with a new local DTN router, or a new user is created at a kiosk, the user sends a REGISTER message containing the user's GUID and the GUID of its custodian, that is forwarded along the default path to an Internet gateway. Along the way, we require it to touch the custodian, therefore the custodian must lie on this path. As the REGISTER message propagates to an Internet gateway, all the DTN routers along the way record the previous hop of the message (i.e. the TCP/IP address of the previous DTN router) in their forwarding tables to be the next hop for any data addressed *to* the user. Thus, data to be transferred to the user follows the reverse of the default path taken by the REGISTER message, while data to be transferred to the Internet from the user follows the default path.

When an Internet gateway gets a REGISTER message, it updates the HLR to map the user's GUID to the GUID of the user's custodian as well as to its own IP address. All future communication with the user happens through that Internet gateway.

The same procedure is followed by custodians to select the Internet gateway they are associated with. Ferries also send REGISTER messages to discover the custodians reachable by them.

Reverse path forwarding is useful for routing bundles on shortcuts. That is, if the destination's GUID is known to a DTN router along the default path, then a reverse path exists to that destination, and bundles can be sent there directly. This allows, for example, a bus going from one village to another to carry bundles between the villages without having to go through the Internet. This allows rapid and secure user-to-user communication without the mediation of a server.

The use of reverse path forwarding has both its pros and cons. On the one hand, in the absence of a definitive solution to the DTN routing problem, it offers a simple way to set up the routing tables in a network. On the other hand, these tables are fragile: if a link were to break, or a DTN router were to fail, the protocol does not recover gracefully from this failure. The lack of fault tolerance can be handled in several ways. For instance, a user or DTN router can periodically send a REGISTER message to refresh paths to it. These protocols would limit outages to approximately one update period, which may be sufficient in practice. Another disadvantage of this protocol is that it relies on manually configured default routes, and can result in sub-optimal performance. We are currently looking into how link state routing, described next, can be coupled with reverse path forwarding for better performance.

8.3 Link state routing

Link state routing has been proposed in recent work on practical DTN routing [10, 12]. In this work, standard link state packet flooding is used to construct network topology graphs, where link weights are set to the expected link delay. A shortest-path algorithm is then used to create forwarding tables. We believe that the solution described here can be used in our system (though as of now, our system only implements simple reverse path forwarding).

The problem of choosing link weights, in general, is a difficult one. The weights should represent not only bus schedules, but the proxy-gateway links should also take into account the queue-lengths at a proxy for data destined to different kiosks. This is because the proxy can reduce end-to-end delay by intelligently scheduling bundles on the proxy-gateway link ahead or behind of each other, depending upon the bus schedules from the gateways to the different kiosks. Essentially, the problem is complex because the time scale of data forwarding on the proxy-gateway link is the same as the time scale of routing. We do not yet have a complete solution to this problem, although a first attempt has recently been made in [9].

Note that even the bus schedules may not be very accurate because buses can get delayed or rerouted on a more or less random basis. Thus, the topology graph at each DTN router should be represented such that incremental changes can be made to it, based on local observations and periodic control plane updates. We are currently looking into adapting the work from [10, 12] into our network model, and extending it with routing and scheduling based on approximate information and incremental updates.

9. SECURITY

We would like our solution to provide secure, private communication, but not at the cost of complexity to a degree that makes the system unusable. Unfortunately, traditional security mechanisms like Public Key Infrastructure (PKI) and Certificate Revocations Lists (CRLs) are inefficient, complicated, and sometimes unusable in our environment for the following reasons.

- PKI decouples user-identity from the public/private key pair. This requires a round trip to a central or replicated lookup database of user-id against public-key whenever a secure end-to-end communication channel has to be established. This can substantially delay actual data transmission. Of course, if a cellular or dialup control plane is available, large communication delays can be avoided. However, this requires both end-points to be enabled for cellular communication, and is an unreasonable assumption to make for all situations.

- In order to evict malicious or compromised entities from the system, CRLs have been used to allow any entity to become aware of other entities whose security keys have been compromised. However, CRLs are unsuitable when updates can be excessively delayed.

For these reasons, we use the new security architecture based on Hierarchical Identity Based Cryptography (HIBC) described in detail in [18], and a briefly outlined next.

9.1 Hierarchical Identity Based Cryptography (HIBC)

Boneh and Franklin [2] proposed the first practical Identity Based Cryptography (IBC) scheme and many variations have subsequently been described in the literature. Unlike traditional PKI, where a user obtains the public/private key pair from a certifying authority, public keys in IBC can be any string, but private keys are obtained from a trusted authority called the Private Key Generator (PKG). Hierarchical IBC extends IBC by establishing a cooperative hierarchy of PKGs. The top-level PKG is called the root PKG, and the other PKGs are called domain PKGs, each of which inherits the first part of its public ID from its parent. A detailed description of Hierarchical Identity Based Encryption (HIDE) and Hierarchical Identity Based Signature (HIDS) is given in [8].

Identity Based Cryptography (IBC) is ideally suited for creating a secure channel in a disconnected environment because the public key of an entity can simply be its public ID, and hence a lookup step is not required. For example, the public ID for a user can be the email address or IMSI of the user itself, and users can even communicate this to each other over a telephonic conversation. Another advantage is that the possession of a valid private key implies that the certification authority has certified the identity. Therefore, a valid signature serves as an assurance of authentication, thus eliminating an extra certificate verification step. The need for CRLs is also avoided by having time based keys, as explained below.

9.2 Incorporating HIBC

Key establishment: We assign a unique public ID to each entity in the system, where entities can be users or DTN routers. Users can choose to have their existing email address or IMSI as their ID or they can apply at a kiosk for a system-id of the form *user@provider.* A single user can even have multiple IDs, in which case a mapping will be maintained by the HLR. The HIBC public keys are derived as SHA-1 hashes of the public IDs, and an initial setup procedure is used to register public IDs with the system and acquire the corresponding private keys. This procedure is described in [18], and uses symmetric key encryption to fetch private keys from the PKG over the network of DTN routers. The kiosk controller grants a one-time symmetric key to each new user. The PKG is responsible to securely supply the kiosk controller with new sets of symmetric keys for this purpose.

The genuineness of public IDs requested by the users is ensured through manual inspection of identification documents by the kiosk or franchise operators. This is modeled on how cellular subscriptions are created when customers buy SIM cards from outlets of cellular companies.

Users who access the system through mobile devices like PDAs or laptops, can store the keys on their personal access devices. Those users who use public access terminals at kiosks can have their keys stored securely at the kiosk controllers, or carry them in USB flash drives. In case USB drives are not available, users can be given registered photo-id cards embossed with their public IDs. Thus, if a kiosk user moves to a new kiosk and does not have the facilities to carry the keys with herself, then she can present her photo-id card to the kiosk operator to gain access to the facilities. Her keys can later be fetched securely from either the PKG or from the old kiosk. Another cheap alternative is for the user to carry a printout of a human-readable base-64 encoded cipher-text of a secret password (about 8-10 characters) known only to her and encrypted on her public key. The kiosk operator can manually verify the correctness of the cipher-text and grant access to the user.

Secure communication: Once the keys and system parameters have been established, HIBC is used for the creation of an end-to-end secure channel: the sender encrypts all data with the public key of the recipient, and only the recipient can decrypt the data. This provides confidentiality, integrity, and authenticated access. Note that the recipient can either be another user, or an application end-point like an email server. The sender can either be a mobile user, or a kiosk controller that has secure access to the keys of the user.

Mutual authentication: When a mobile host or DTN router establishes an opportunistic connection with another mobile DTN router, it tries to connect to the DTN router and both entities participate in a mutual authentication procedure. This is used to ensure that malicious users and rogue DTN routers are not involved in the communication session.

Control plane security: Besides allowing end-to-end secure channels, HIBC also protects the infrastructure from a class of attacks on the location management subsystem. Recall that a mobile host sends control messages whenever it changes its location. We use HIDS between the mobile host and the location registers being updated, with the system parameters of the mobile host's HIBC system piggybacked on the message. This ensures safety from fabrication of control messages, redirection attacks, and the creation of dead-ends by unauthorized updating of location registers. Similar principles are used for authentication of routing metric information based on which best routes can be selected by the endpoints.

Auditing: Custodians in DTN send messages to the end-systems when custody has been transferred. We require custodian DTN nodes to sign these messages for the bundles that they take custody of, in order to ensure safety from spoofing. End users can store these acknowledgements for auditing. Since the HIDS scheme itself ensures non-repudiation, the audit logs can be used as proof of custody transfer.

Roaming access: Much like roaming access provided across different cellular networks, this can be enabled across DTNs hosted by different providers. Mobile hosts and DTN routers exchange system parameters to be able to communicate with each other, along with collection of verifiable billing statistics.

Private key compromise: A drawback of using a public ID such as an email address or IMSI as the public key is that if the corresponding private key gets compromised, then the public ID has to be changed as well. We provide safeguards for this by allowing the users to store their *basic private keys* on a secure kiosk controller or home desktop, and periodically refresh the keys they carry on their personal mobile devices. In case the mobile device gets stolen or lost, the temporary keys will expire soon after and render the device unusable for decryption purposes. These refreshable

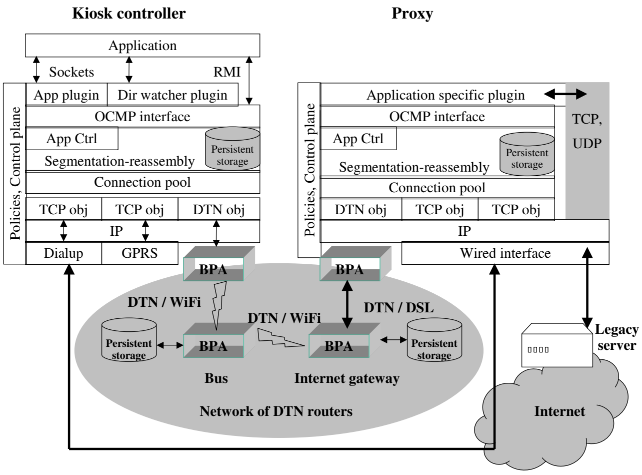

Figure 4: OCMP/DTN integration

keys are meant only for end-to-end data transfer, and provide a practical way to ensure forward secrecy by extending the HIBC trees per user by an extra level.

Key revocation: Unlike the method used for handling private key compromise described above, key revocation is done by having the PKGs periodically refresh a variant of the basic private keys of all the entities. These keys are only used for mutual authentication and not for secure end-to-end data transfer. This method provides a practical alternative to CRLs for user revocation.

10. APPLICATION SUPPORT

We would like to provide a simple API for application development that supports session persistence, intelligent use of multiple networks, and use of unmodified legacy servers. The Opportunistic Connection Management Protocol (OCMP) [19] provides all these features. An OCMP client running on a mobile device can communicate opportunistically over multiple network interfaces to an Internet proxy. Legacy application protocols are hidden from the client through application-specific plugins that talk to legacy servers on behalf of the client.

OCMP differentiates between shared-control, application-control, and application-data packets. Shared-control packets are used primarily for connection state updates between the mobile device or kiosk-controller and the OCMP proxy (common to all applications); application-control packets are used for conveying application-specific parameters between the plugins on the client and proxy; and application-data packets are used for bulk data transfer. On the client side, an OCMP plugin is instantiated for every transaction (send or receive) by a legacy application. The plugin can create multiple data streams to the proxy, where each disconnection-tolerant stream is recognized by a unique session identifier. The proxy then reassembles the data of each stream, and passes it to a corresponding application plugin on the proxy side. This plugin can also receive application-control packets to reconstruct the application state required to instantiate a legacy transaction on behalf of the client. A similar procedure is followed for legacy communication from a server to the mobile device.

We extend OCMP to operate on an end-to-end basis even across multiple levels of disconnections. We do this by modeling DTN as a network interface for OCMP that implements its own transport layer protocol.

Whenever OCMP detects a WiFi network, it examines the SSID to determine whether the network belongs to a DTN router that is a part of our infrastructure, or it is a third party WiFi network connecting into the Internet. A different type of connection is instantiated accordingly. Each connection instance in OCMP is encapsulated in a *Connection* object, and a collection of such objects is maintained in a *ConnectionPool* object. Therefore, we have implemented a new *Connection* object for DTN that talks to a DTN bundle protocol agent (BPA) running either locally or on a DTN router. This is shown in Fig. 4, and discussed as follows.

OCMP on kiosk controller: A BPA runs locally on the kiosk controller. Thus, whenever OCMP detects a new network from a mobile DTN router, it opens a connection to the local BPA, and the BPA connects to the corresponding BPA on the mobile DTN router. OCMP then sends data to the local BPA which encapsulates it in DTN bundles and forwards the bundles to the correspondent BPA. Custody transfer acknowledgments are relayed back from the BPA to the DTN *Connection* object in OCMP. Thus, OCMP is made to believe that it is actually talking to a proxy in the Internet, but the BPA successfully masks the absence of an end-to-end route to the Internet. Similar steps are followed for data to be downloaded to the kiosk controller from the mobile DTN router. Note that the BPA passes data to OCMP only for kiosk users. For mobile users who use the kiosk controller only as a DTN router, this data is retained at the BPA itself.

OCMP on mobile host: A BPA does not run locally on the mobile host, but OCMP connects wirelessly to the BPA on the kiosk controller or a DTN router, and transfers data to it over the BPA's RPC interface. The subsequent working is identical to the previous case.

OCMP on proxy: For data to be sent to a kiosk controller or a mobile host, the proxy first checks whether the endpoint is registered with a custodian in the Internet region or not. This step is crucial for the proxy to decide whether to retain the data in OCMP or to push it into the DTN overlay by routing it to an appropriate Internet gateway. In the former case OCMP layers itself on TCP/IP and for the latter case, the proxy instantiates a connection to the BPA running locally and dispatches all the data to this BPA for eventual delivery to the user's custodian.

11. OPPORTUNISTIC LINK USE

We now present the sequence of actions that happen when an opportunistic connection is detected by a kiosk-controller or mobile device due to the arrival of a ferry. It serves to illustrate how the naming, addressing, forwarding, and security schemes come together at different layers of the protocol stack.

1. **Link association**: The WiFi NIC on the ferry uses active beaconing to broadcast its presence. The WiFi card on the kiosk controller detects the peer WiFi node when it comes in range, and attempts to authenticate and associate with it.

2. **OCMP and BPA initiation**: OCMP running on the kiosk-controller detects a successful association, recognizes the SSID of the WiFi network, and notifies the BPA to connect to its correspondent BPA on the ferry.

3. **Mutual authentication**: The BPA on the kiosk controller tries to open a TCP connection to the correspondent BPA on the mobile DTN router on a well known port. Both the entities then run a challenge-response protocol to mutually authenticate themselves as described in [18].

4. **Data download**: Upon successful authentication, the kiosk BPA requests the ferry BPA for any data addressed to the users registered at the kiosk. If data is available, it is downloaded, decrypted, and passed on to OCMP. OCMP reassembles the data and redirects it to appropriate application specific plugins.

5. **Data upload**: If there is upload data pending, OCMP dispatches its data to the BPA. The subsequent steps involved for data upload are similar to the download process, but in addition, we also allow the kiosk controller to decide which bundles will take which route, based on the routing protocol and current load conditions. This is done to accommodate load sensitive flow control in our architecture.

6. **Routing**: To enable routing based on load statistics, we allow the kiosk BPA to query the mobile DTN router about queue sizes and routing metric information, and source route the DTN bundles for upload accordingly. The information can also be authenticated and verified if needed. The ferry can continue to query other routers on the way, to decide optimal and load balanced routes for the bundles it is carrying.

7. **Session persistence**: OCMP application plugins can specify a session identifier for each data transaction like a file upload or download. This is used to identify the application endpoint at both the proxy as well as the kiosk controller to which data is finally redirected after reassembly. End-to-end session level ACKs are used to ensure that the data finally reaches its destination by incorporating a retransmission timeout in case some data gets lost in the DTN overlay due to system failures.

12. SCALING

In our initial deployment, we intend to support about 80 kiosks and 5000 users with about 10 ferries and a handful of Internet gateways. At this small scale, reverse path forwarding routing on a flat topology appear to be adequate. The Indian government, however, has announced that it intends to deploy 100,000 kiosks by the end of 2008. Issues of scale, therefore, cannot be long ignored. Here, we sketch out an approach to allow scaling, deferring details to future work.

At first glance, the major non-scalable element in our architecture appears to be the use of non-aggregable GUIDs for routing. In fact, we do not believe this poses a significant problem. The HLR may need to track millions, or even billions of GUIDs. However, today's systems can handle gigabyte-sized tables in main memory, and clustered solutions that leverage Distributed Hash Tables can offer several orders more capacity and nearly perfect fault tolerance [17]. Similarly, routing tables in DTN routers can easily scale to tens of millions of entries because lookup delays of the order of milliseconds, or even seconds, are acceptable.

The real issue, we believe, is that a flat topology does not scale well with respect to the number of update messages. For instance, in reverse path forwarding, when a user moves from one kiosk to another, the new REGISTER message has to be sent all the way to the HLR. However, users are typically expected to move within villages close to each other. The standard technique to allow scaling is to introduce hierarchy. We now outline how we can add hierarchy in routing and location management to allow our system to expand to millions of users.

Essentially, we break up the topology into autonomous *regions*. A region is a collection of mutually reachable DTN routers, determined by administrative policies, communication protocols, naming conventions, or connection types. Regions would usually be contained within a physical boundary, though a single region may span multiple geographical areas. We do not place any assumptions on the organization of regions. From the perspective of the outside world, a region is reachable by a set of border gateways (much like BGP routers in the Internet). To reduce the overhead in locationing, the HLR maintains a mapping from a user's GUID, not to a custodian, but to the *border gateways* of the region in which the user is present. Additionally, a region's border gateways maintain a replicated copy of a VLR (Visitor Location Register), that carries a mapping from the user's GUID to its current custodian. Therefore, if a user changes custodians, but stays within a region, it does not need to update the HLR, increasing scalability.

Region partitioning also improves the scalability of routing. Suppose link state routing is being used. In this case, routing updates within a specific region need not include information of the entire network, but only the intra-region routing state of that region. This significantly reduces the number of routing updates. Similarly, if flooding is used, instead of flooding the entire network, only the destination region needs to be flooded.

We defer details of region construction and incremental optimization to future work.

13. USER-TO-USER VS. USER-TO-SERVER COMMUNICATION

We now discuss one aspect of our architecture that is non-obvious, but relevant to our overall goals, that is, its support for user-to-user communication. In other words, our system allows a user to directly send data to another user without having to go through an intermediate server or proxy. The significance of this decision has to do with efficient communication paths between users.

Consider a user in a village who wants to send a private message to another user in one of the villages served by the same ferry. This is likely to be common because most communication is local. In a system that supports only user-to-proxy or user-to-server communication, the sender would send a message addressed to the receiver, but tunneled to a well-known proxy or server. This forces data to traverse the path to the proxy or server and back. This is just another version of the 'triangle routing' problem in Mobile IP [16]. To avoid this, it is necessary to permit direct user-to-user communication. This requires public key information and locationing information to be disseminated in the network. Our use of hashes of user names as forwarding identifiers, and using the same identifiers for public keys, allows us to integrate naming, addressing, routing, and security. Another way of looking at this is that we are co-opting the routing protocol to distribute public keys in the same way as it distributes routes.

14. IMPLEMENTATION OVERVIEW AND STATUS

We have implemented the architecture built upon the OCMP [19] and the DTN-2 [6] reference implementations. In this section, we outline the implementation and the current status of our work.

14.1 Simplifications

Our prototype makes some simplifications to the general architecture. (a) We treat every Internet gateway as the custodian for all users accessible through it. (b) We have only implemented reverse path forwarding, so all communication to a user is on a single path. (c) We have not yet implemented a cell-phone based control plane, (d) multiple routing regions, and (e) data replication.

14.2 Implementation environment

The two main software components in our design are the OCMP stack and the DTN router, implemented in the form of a Bundle Protocol Agent (BPA). The OCMP stack provides application support and is implemented in Java. The DTN reference implementation BPA, extended as described in Section 10, is written in C++ and runs on the kiosk-controller, ferries, and at the proxy.

OCMP implementation has been described in [19] and we do not describe it further for reasons of space. We next describe the extensions made to the DTN2 reference implementation.

- **Control plane**: The current DTN2 implementation does not clearly separate the data and control planes. Instead, a bundle router class is responsible not only for bundle forwarding decisions, but also for inter-router signaling. We have rewritten the bundle router class to allow a separate application to add, delete, and modify routes. This application, addressable by a special GUID, is responsible for implementing reverse path forwarding, link detection, mutual authentication, and user registration. The control plane application can also make use of other network interfaces like a dialup or GPRS connection for exchange of routing updates.
- **Reverse path forwarding**: The DTN2 reference implementation only supports flooding and manually configured routes. We have implemented reverse path forwarding using handshakes between neighboring control applications.
- **HLR**: We have implemented the HLR using the publicly available OpenDHT distributed hash table [17]. Internet gateways use the OpenDHT API to install and update GUID translation tables that are then stored in the Bamboo DHT.
- **Addressing namespace**: The DTN2 reference implementation allows the definition of customized namespaces. We have used this feature to define our own namespace, called 'tca'. In this namespace, users and DTN routers are addressed as *tca://custodian/GUID*, which conveniently represents both the custodian and user GUIDs in standard URI format.

14.3 Identity mapping

Users at kiosks store files and access applications hosted by the kiosk-controller. Therefore they need user IDs on the kiosk operating system. We are currently using Linux-based kiosk controllers, so we assign each user a Unix UID. To send and receive bundles, we need to map these UIDs to GUIDs. We do so in a file called /etc/idmap. When sending data, either the OCMP-aware application, or an OCMP-aware server, use this mapping to translate from the application's UID to a GUID, which is placed in the bundle's 'sender' field. Symmetrically, on receiving a bundle with a particular GUID, the OCMP demultiplexing engine uses this map to translate to the UID of the user, and hands the bundle to the appropriate application. The /etc/idmap file is managed by the kiosk owner, and entries are inserted when new users join the system using a web-based interface.

14.4 Applications

Our system supports several delay-tolerant applications, including email, Jabber (Instant Messaging), FTP, HTTP-GET, and blog updates. These are described in related work [19] and we elide details in the interests of space. Moreover, over 200 e-governance applications have been implemented by our partners, eGovServices [26]. These include mutual fund registrations, birth and marriage certificates, railway reservations, electricity bill payments, and other applications involving financial and non-financial transactions. These applications have already been designed in a disconnection tolerant fashion. All transaction data is written into a single file that is transferred across to the Internet using our framework. Operators then manually execute these transactions and deliver the receipts back to the users in the same fashion.

14.5 Status

We have implemented OCMP, all of the DTN extensions, and (separately), the security protocols. We are currently integrating the security protocols into the control-plane. In May 2006 we deployed our systems in Anandpuram, a village in South India about 20 km from the city of Vishakapatnam, in collaboration with eGovServices, and we plan to do a bigger deployment in the next few months.

The Vishakapatnam District Rural Development Agency (DRDA) has set up over 40 kiosks around Vishakapatnam, and the Anandpuram kiosk is part of this initiative. At the kiosk, we set up a recycled PIII PC as a public access terminal, and connected the kiosk PC and the recycled PC with a Net4801 Soekris Single Board Computer (SBC) as the kiosk controller. We used a similar SBC as the Internet gateway with DSL broadband (128 kbps) at the DRDA head office in Vishakapatnam. At both places, we used external 9 dBi omni-directional antennas for wireless connectivity, and 40 GB hard-disks for local storage. The kiosk SBC runs from a 42 AH battery that is charged by two 1.2A @ 12V solar panels, and the gateway SBC runs on UPS. This ensures 24 hour uptime for both the nodes. For the ferry, we chose a government vehicle that regularly goes between the head-office and the kiosk for a microfinance initiative also led by the DRDA. We installed a similar Net4801 SBC in the vehicle, powered it from the vehicle battery, padded it with foam to protect it from vibrations, and attached an external 7 dBi omni-directional antenna with a magnetic mount base that sits outside on the trunk. Finally, we integrated the eGovServices e-governance portal with the OCMP API to allow for delay tolerant connectivity.

We plan to observe this deployment over the next few months and learn more about reliability and usability concerns. Further details on our deployment are available at [27].

15. RELATED WORK

We draw upon a wealth of ideas in the literature that deal with disconnection tolerance, mobility management, semantic-free naming, data privacy, and routing. We have outlined the principles derived in past work in Section 3. Here, we consider systems that are most closely related to ours.

The use of data ferries in the context of MANETs and sensor networks is well known; for example, see [23]. Current work on DieselNet [3] is also relevant. However, these systems essentially present point solutions that only address a few of the goals outlined in Section 2. Moreover, this work is not directly applicable to low-cost and reliable kiosk networking in rural areas.

The work closest to ours in spirit is that of Daknet [15]. They use MAPs (Mobile Access Points) mounted on buses or vans, which regularly traverse villages and come in wireless contact with rural kiosks to opportunistically upload and download data. Buses were fitted with omnidirectional antennas, and kiosks with omnidirectional or directional antennas depending upon the orientation of the kiosk with the road. Data sessions of an average duration of 2:34 minutes were measured, during which up to 20 MB of data could be transferred. We have experienced similar performance because our operating environment is practically the same as that of Daknet. However, Daknet does not implement a generic architecture that can be used to build new applications, support integrated location management system, operate across multiple levels of disconnections, or allow (to our knowledge) data privacy. Our system provides all these features, along with many rich applications ready for immediate use.

Our work relates broadly to rendezvous-based mobility support in HMIP [21] or I3 [22], location-independent identifiers in HIP [13], and semantic-free names in DoA [1], but these protocols are designed to work in connected Internet like environments. We have suitably adapted the insights from these schemes into our architecture.

Disconnection-tolerant networking [24] also forms a fundamental part of our architecture. However, DTN is a general platform, and we have specialized and extended it to handle our specific interest. We presented details of our extensions to the DTNRG architecture in Section 14.

16. CONCLUSIONS AND FUTURE WORK

We have presented a detailed architecture that addresses the problem of low-cost and reliable connectivity for rural kiosks. Unlike past work, which has focused on point solutions, we propose a pragmatic and comprehensive solution that meets our stated goals, as detailed next.

We meet our goal of low-cost through sharing every component of the system; in the base solution, there are no unshared components. We also reduce cost by leveraging existing transportation networks for data transfer. We make our system reliable by judicious use of replication, both of data, as well as of hardware, by carrying spares in the ferries. Our system is inherently disconnection tolerant, because of its use of the DTN architecture. Our extensions to the DTN architecture allow user mobility and interoperability with legacy servers. We use HIBC to guarantee data privacy, making our system suitable for rural banking. Finally, our use of OCMP allows an application to use all available networks, while still giving applications the ability to choose the network that best fits its needs. OCMP, implemented in Java, and communicating over TCP/IP with a bundle protocol agent, allows us to support both kiosk users and laptop/PDA users seamlessly.

We outline future work next.

16.1 Scaling

If our solution is widely adopted, we will need to address scaling issues head on. This remains an open and exciting area of research. In particular, the problem of automatic region construction and its incremental optimization seem both academically challenging and necessary in the field.

16.2 Reliable transfer

The DTN architecture allows a sender to request notifications when custody is transferred or when a bundle is received at the destination. These correspond to hop-by-hop and end-to-end acknowledgments. The open problem, however, is what to do when a source does not get an end-to-end acknowledgment in time.

16.3 Intelligent routing

All three routing solutions that we have studied are unsatisfactory. An ideal routing scheme should take topology, update delays, as well as load balancing into account. We believe that this is also a fruitful area for research.

16.4 Erasure coding

We are presently using TCP for bundle transfer between DTN routers. It appears likely that using erasure coding for this transfer would allow us to choose fractional packet replication. We are working on models that couple this replication with load-sensitive routing and flow control.

17. REFERENCES

[1] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, M. Walfish, "A Layered Naming Architecture for the Internet," Proc. ACM SIGCOMM 2004.

[2] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," Proc. Crypto 2001.

[3] J. Burgess, B. Gallagher, D. Jensen, B. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking," Proc. IEEE Infocom 2006.

[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, "Delay Tolerant Network Architecture," Internet Draft http://www.dtnrg.org/specs/draft-irtf-dtnrg-arch-02.txt, July 2004.

[5] S. Cheshire and M. Baker, "A Wireless Network in MosquitoNet," In IEEE Micro, Feb 1996.

[6] 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, Dec 2004.

[7] K. Fall, "A Delay-Tolerant Network for Challenged Internets," Proc. ACM SIGCOMM 2003.

[8] C. Gentry and A. Silverberg, "Hierarchical ID-Based Cryptography," Proc. International Conference on the Theory and Application of Cryptography and Information Security, 2002.

[9] S. Guo, M. Ghaderi, and S. Keshav, "Opportunistic Scheduling in Ferry Based Networks," Proc. WNEPT, 2006.

[10] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," Proc. ACM SIGCOMM 2004.

[11] A. Jhunjhunwalla, "Wireless Mesh Access in Rural India," Presentation at COMSWARE 2006, New Delhi, January 2006.

[12] E. Jones, L. Li, P. Ward, "Practical Routing in Delay-Tolerant Networks," Proc. Workshop on DTN, 2005.

[13] R. Moskowitz, P. Nikander. P. Jokela, and T. Henderson, "Host Identity Protocol," http://www.potaroo.net/ietf/ids/draft-ietf-hip-base-00.txt, 2004.

[14] J. Ott and D. Kutscher, "A Disconnection-Tolerant Transport for Drive-thru Internet Environments," Proc. IEEE INFOCOM 2005.

[15] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: Rethinking Connectivity in Developing Nations," IEEE Computer, 37(1):78-83, 2004.

[16] C. Perkins, "IP Mobility Suport for Ipv4," http://www.ietf.org/rfc/rfc3344.txt, Aug 2002.

[17] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A Public DHT Service and Its Uses," Proc. ACM SIGCOMM 2005.

[18] A. Seth and S. Keshav, "Practical Security for Disconnected Nodes," Proc. NPSEC 2005.

[19] A. Seth, S. Keshav, and S. Bhattachharaya, "Opportunistic Data Transfer Over Heterogeneous Wireless Access Networks," http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/05/ocmp.pdf, Work in progress.

[20] A. Snoeren and H. Balakrishnan, "An End-to-End Approach to Host Mobility," Proc. ACM MOBICOM 2000.

[21] H. Soliman, C. Catelluccia, K. Malki, L. Bellier, "Hierarchical Mobile IPv6 mobility management (HMIPv6)," http://www.ietf.org/internet-drafts/draft-ietf-mipshop-hmipv6-04.txt, 2004.

[22] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet Indirection Infrastructure," Proc. ACM SIGCOMM 2002.

[23] W. Zhao, M. Ammar, and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," Proc. ACM MOBIHOC 2004.

[24] DTN Research Group (DTNRG), http://www.dtnrg.org/, 2006.

[25] One Laptop Per Child (OLPC), http://laptop.media.mit.edu/, 2006.

[26] eGovServices, http://www.egovservices/, 2006.

[27] Anandpuram Deployment, http://blizzard.cs.uwaterloo.ca/tetherless, May 2006.