Applying DTN to Mobile Internet Access: An Experiment with HTTP

Technical Report TR-TZI-050701

Jörg Ott
Helsinki University of Technology,
Networking Laboratory
<jo@netlab.hut.fi>

Dirk Kutscher
Technologie-Zentrum Informatik,
Universität Bremen
<dku@tzi.org>

2005-07-12

http://www.drive-thru-internet.org/

Applying DTN to Mobile Internet Access: An Experiment with HTTP

Jörg Ott* and Dirk Kutscher**
* Helsinki University of Technology, Networking Laboratory <jo@netlab.hut.fi>
** Technologie-Zentrum Informatik, Universität Bremen <dku@tzi.org>

***Abstract*— The DTN architecture and protocol specifications developed in the IRTF provide a platform for communication in challenged networking environments. The delay and disruption tolerant messaging service is inherently asynchronous in nature and generally used by specifically developed applications, e.g., to exchange data between intermittently connected nodes. Intermittent connectivity is also experienced by mobile users as they usually cannot rely on being always connected in a wireless environment—and thus will also benefit from a disruption-tolerant networking infrastructure. This paper explores applying the DTN architecture to interactive access to Internet services based upon HTTP as non-trivial example and discusses the protocol requirements, analyzes performance implications, and discusses issues with DTN as a communication substrate.**

I. Introduction

Over the past years, starting from the ambition to provide a framework for an *Interplanetary Internet (IPN)* [1], the architecture for *Delay-tolerant Networking* [2] has been developed in the DTN Research Group of the IRTF [3]. While originally targeted at (deep space) long-delay links that may only be unidirectional, may often be only intermittently connected, and may also require explicit scheduling of communications ahead of time, the asynchronous nature of the DTN architecture is also suited for terrestrial communications in challenged networking environments. Examples for such deployments include information exchange within and data retrieval from sensors and sensor networks [4] [5], providing asynchronous network access to remote and otherwise poorly connected regions [6] [7], and other opportunistic wireless communications (e.g., [8]). Most present deployments of the DTN concepts have in common that they largely apply disruption tolerance to new communication environments and make use of newly developed applications. Two notable exceptions are providing email access using existing client and server software [6] and the tetherless application architecture. These examples hint at the principal value of applying DTN to general Internet communications.

In fact, everyday mobile or nomadic communications exhibit conceptually similar connectivity properties as the aforementioned challenged environments: mobile users experience intermittent connectivity when they connect to and disconnect from networks. Connectivity patterns cannot necessarily be influenced nor predicted by users, connection and disconnection periods may be of arbitrary duration, ranging from a few seconds to many hours or days. Particularly extreme and often unpredictable connectivity patterns can be observed when users move at high speeds, e.g., in vehicles or aboard trains [9] [10]. While numerous approaches are pursued towards achieving ubiquitous and seamless connectivity across different underlying networks and service providers to keep users *always best connected* [11] [12] [13] [14], experience shows that permanent network access is far from reality today [15] [16] (and striving for ubiquitous connectivity does not appear to be an (economically) viable option either, at least not at a global scale [17]).

Instead, nomadic and wireless Internet access needs to be able to deal with intermittent connectivity and protocols as well as applications should be designed to consider this as the rule rather than the exception (or: the error condition): *disruption-tolerant networking* is required across all layers of the protocol stack. In the Drive-thru Internet project, we have followed this approach and designed a protocol and system architecture for dealing with extreme intermittent connectivity in which dedicated nodes (which run specific session layer protocol [15]) shield unchanged application peers from the variable connectivity conditions and may additionally provide performance enhancements [16].

In this paper, we apply the system concepts of our Drive-thru approach to the DTN architecture and investigate the implications of generalizing the DTN concepts beyond purely asynchronous communications to also support certain classes of (unmodified) interactive Internet applications. The DTN architecture—designed for dealing with (tentatively larger) messages of application protocols that require only few exchanges—is clearly at odds with often chatty Internet protocols and we expect significant overhead if application message exchanges are mapped one-to-one onto DTN messages. Nevertheless, we deliberately choose a particularly highly interactive and hence particularly unsuitable protocol to stress-test the DTN architecture and determine some performance characteristics: the HyperText Transfer Protocol [18]. HTTP is much more interactive than other protocols widely used (not just by mobile users) such as SMTP, POP, and IMAP and is also more complex to adapt to a DTN environment as we will discuss below. As such, we assume it to be close to a worst-case benchmark when operating Internet protocols in a delay-tolerant environment.

This paper is structured as follows: section II introduces the mobile Internet access scenario in more detail. Section III discusses relevant related work. Section IV presents a possible system architecture mapping mobile Internet access onto a

DTN infrastructure and reviews different classes of (existing) applications. Section V discusses the performance evaluation we have carried out. Sections VI and VII suggest operational optimizations and improvements for the DTN architecture to better deal with interactive mobile communication scenarios. Finally, section VIII reviews our results and hints at future work.

II. Mobile Internet Access: A DTN Scenario

Figure 1 depicts a mobile Internet access scenario as we have investigated in the Drive-thru Internet project [19]: a mobile user in a car (or other vehicle) is passing through wireless LAN hot-spots—connectivity islands—located at the roadside. The hot-spot may be explicitly provided for travelers, e.g., by gas stations or restaurants in rest areas, or may be implicitly available for use in city areas from cafés or other locations. Hot-spots may comprise one or more access points and may be operated by different WISPs as is shown on the left hand side of the figure [9].

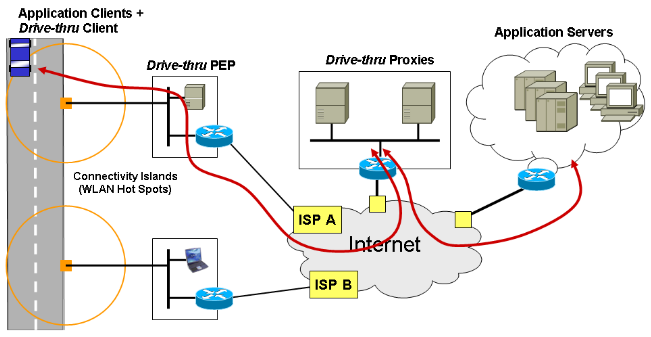

Fig. 1. Example for Mobile Internet Access on the Road

When a mobile user travels along the road, she will experience alternating periods with and without connectivity: The beginning of a connectivity period may often not be predictable, their respective end may be predicted from signal strength readings. Due to traffic conditions, the reach of a connectivity island may vary as will the quality of the radio channel (and hence transmission and error rates). These also heavily depend on the radio equipment installed in the vehicle or the mobile device and used in the hot-spot. Our measurements have shown that, with proper equipment, mobile users may, e.g., obtain more than 60 s of connectivity at 120 km/h (up to 2 km) and may transfer some 20–70 MB of data to/from a peer located in the hot-spot in a single pass [10]. A mobile user shares the bandwidth reasonably fairly with fixed and other mobile users. Automatic authentication with wireless ISPs and autoconfiguration enable users to effectively use most of their connectivity window for data transfer [20] [21].

To support existing Internet applications in such an environment, we apply connection-splitting between the mobile and the fixed endpoints, realized in two dedicated components: the *Drive-thru client* running locally to or on the mobile device and the *Drive-thru proxy* somewhere well-connected in the fixed network. Our *Persistent Connection Management Protocol (PCMP)* follows a session layer approach towards achieving continued application sessions in spite of connectivity interruptions, changing IP addresses, NATs/firewalls, and highly variable performance characteristics. Application-specific modules implemented as plug-ins on the Drive-thru client and proxy perform additional functions to prevent application layer timeouts, provide user feedback, and allow for asynchronous processing of messages (exchanged with the application peers) while the user is disconnected—thereby also offering performance enhancement to maximize the utilization of the short connectivity period [15].

When the application-specific plugins operate asynchronously, this scenario bears striking similarities with DTN scenarios using the DTN bundle protocol, e.g., for data collection or email transmission. The major difference is that the Drive-thru architecture also provides efficient support for interactive applications (as we will also discuss in section IV below) while the DTN architecture targets asynchronous applications. In the remainder of this paper, we will assess the suitability of the DTN architecture to also support “synchronous” applications.

III. Related Work

IP communications on the road has independently been studied by the FleetNet [22] and Networks on Wheels [23] projects, albeit with a different focus and slightly different goals: both primarily target inter-vehicle communications in wireless ad hoc networks for traffic-related control information and data sharing across vehicles. Despite Internet access was only considered a secondary objective, a proxy-based architecture was developed [24]. Network access for vehicles via multiple (complementary) networks to keep users always best connected has also been addressed in various projects, including OverDRIVE [13], IPonAir [14], and the Mobile Access Router [12]. A mobile router dealing with temporary connectivity loss is also studied in the eMotion project [25]. The latter two implement dedicated mobile routers to provide wireless network access, addressing multi-provider support at the IP layer and temporary connectivity interruptions at the transport-layer, respectively. Finally, the DHARMA project targets agent-supported mobility in general [26] as does the Tetherless computing architecture (TCA) [8]. All of these projects have devised their own infrastructure support to deal with intermittent connectivity.

Dealing with intermittent connectivity is an explicit goal of disconnection/delay-tolerant networking (DTN) [2], originally developed for deep space communications [1] but now also applied to terrestrial scenarios, including sensor networks and communication infrastructures for remote areas. Today’s DTN design uses the paradigm of purely asynchronous communications (modeled after postal mail and email): Rather than relying on end-to-end communications on the basis of (small) packets, DTN routers with persistent storage are introduced to convey information units of arbitrary size (*bundles*) hop-by-hop from the source to the destination. DTN routers operate above the transport layer and may interconnect different internets with arbitrary underlying protocol stacks [2]. The DTN routers interconnect so-called *regions*[1] in which all nodes share the same (internetworking) protocols and thus build up the overall DTN (an inter-internetwork). Bundles are passed from the source via one or more routers to the final destination.

DTN nodes are identified by peers of the type *(region-id, node-id)* and routing takes place based in a two-stage process: first towards the target region, then to the ultimate destination node (e.g. identified by an IP address or a domain name).[2] Routing algorithms take the non-permanent connectivity into account [27]. A *custody transfer* mode allows to ensure reliable delivery while delegating the responsibility to the next capable DTN router; forwarding and custody notifications may provide information about the delivery progress of a bundle, return receipts from the receiver provide end-to-end semantics, albeit not necessarily as an immediate response [28].

The DTN architecture serves now as a general framework for communications in *challenged networks* and has found application in numerous projects: e.g., to provide asynchronous connectivity to the Saami people in Lapland [6] or to other remote areas [7], to collect information from sensor networks [4] [5], or to provide messaging services between mobile nodes. The similarity of the tasks make the DTN architecture basically applicable to the mobile Internet access scenario introduced in section II—with the exception that a) mobile users will not restrict themselves to asynchronous communications and b) their local applications (as well as the peers in the Internet) do not support the DTN architecture. While the former part a) has not been investigated in depth yet, there is precedent for the latter b): for Internet access via challenged networks (such as wireless terrestrial and satellite networks), proxies have frequently been used for performance enhancement [29] [30]—as also suggested in the context of FleetNet [24] and Drive-thru Internet [16]. [3]

IV. A DTN Proxy Architecture for Mobility Support

In this paper, we combine ideas from various of the aforementioned approaches: we start by modifying our Drive-thru architecture and substituting the Drive-thru client and proxy by bundle routers.[4] To interface the DTN infrastructure to existing application peers, we implement dedicated DTN applications on top to serve as specific *DTN application proxies*. They translate between stream-based communication using TCP to the application peers and the bundle-based transmission between the bundle routers. The mobile DTN application proxies may either run on the DTN router or on the mobile user's device; in our setup, all three are integrated in the mobile user's laptop. Similarly, the fixed DTN application proxies may or may not be co-located with the fixed DTN router. An overview of a simple case of the resulting architecture is depicted in figure 2.

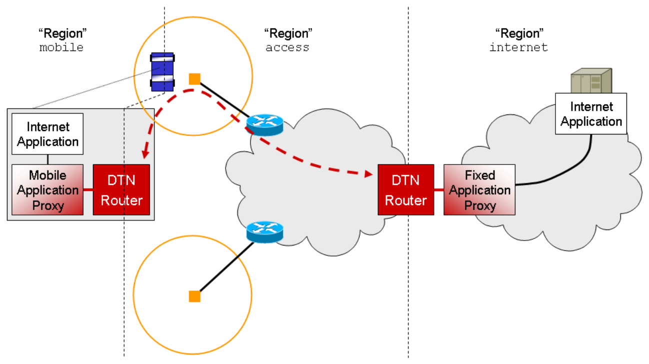

Fig. 2. Example for Mobile Internet Access on the Road

The following subsections address the various aspects of introducing DTN to Drive-thru Internet in more detail.

A. Setup, Naming, Addressing, and Routing

As shown in figure 2, we distinguish three realms: the *fixed* Internet on the right hand side, the *mobile* node-/link-local network, and the DTN region serving as *access* "network" and providing mobility support. Fixed and mobile realm essentially belong together and they are unified in case of a permanent network attachment by the mobile node. While mobile, however, the access realm interconnects the two plain IP networks by performing disconnection-tolerant tunneling of user traffic).

From a DTN perspective, these three realms map onto three regions: the mobile network of a vehicle ("mobile"), the regular (fixed) Internet ("internet") and the intermittently connected access network ("access"). The mobile bundle router connects to the *mobile* region and when the user is on the road, the mobile DTN router opportunistically connects to the *access* region. When the user is connected via a stationary link, the mobile bundle router has also direct access to the *internet* region. The fixed bundle router always connects to the *internet* and the *mobile* regions.

Fixed bundle routers are assigned names by their respective providers with the same names used for both the fixed and the mobile regions: e.g., `(internet, br1.wisp.com)` and *(access, br1.wisp.com)*. Mobile bundle routers can be named using some unique scheme for the access and the mobile network, e.g., *(access, user-3361875.provider.com)* and `(mobile, user-3361875.provider.com)`.[5] For address resolution within a region, we follow the same approach

[1]**UPDATE:** The concept of regions is being heavily debated on the DTNRG mailing list at the time of writing and their notion of having topological relevance will likely go away. As the replacement concept is not yet finalized, we stick with regions throughput this paper at this point. We note that the approach we present will also work with the new DTNRG naming concept.

[2]The DTN application is identified further by some local identifier such as a port number).

[3]Application-specific support for *disconnected operation* as provided, e.g., in Coda [31], is well-noted but is not discussed further as our focus in this paper is on interactive Internet applications.

[4]We will discuss a possible further optimization—optionally placing bundle routers also in the hot-spots—in section VII below.

[5]We consider it useful to use unique names even for the mobile region to avoid name clashes in case several mobile networks temporarily merge.

as for Drive-thru Internet and require mobile node (i.e., their bundle routers) to actively contact the fixed ones. Thereby, IP address bindings need only exist for fixed bundle routers. In this model, all three regions may use DNS for address resolution and TCP/IP for exchanging bundles.

Routing decisions from mobile nodes to the fixed network are governed by static routing tables: Each user may choose her own (set of) fixed bundle routers and also select a target DTN application proxy (per application protocol). Both implicitly allow for load distribution, robustness against failure of individual systems, and proper selection of a trusted peer entity per user. Fixed bundle routers and DTN application proxies may implement customer-specific policies defining how many resources a particular user is allowed to consume. Routing in the opposite direction is statically configured to lead directly to the respective mobile bundle router as there is only a single region to cross from the fixed to the mobile bundle router. Hence, the former may forward bundles to the latter whenever a path becomes available, i.e. a transport connection is set up by the mobile bundle router.

Note that, in the present setting, no bundle exchange takes place within the *mobile* and the *internet* regions (plain application-via-IP communication is used instead). Nevertheless, they are defined for routing purposes: if a direct path becomes available from the mobile bundle router to the `internet` region, this is an indication that the `access` region *may* be bypassed and communication can be carried out directly (which can be expressed, e.g., in terms of routing metrics).

Finally, it is up to the application protocols to be used in this environment to provide the necessary contact information to properly route their messages to the ultimate destination across (multiple) intermediaries. This is discussed in the following subsection.

B. Applications

With a proper routing infrastructure in place, we can turn our attention towards the applications. While DTN-aware applications are designed to work with bundle routers in a purely asymmetric communication environment, existing Internet applications are not. However, under certain circumstances, many of them may become workable with DTN-based mobile Internet access, at least to a limited degree.

We have classified Internet applications with respect to their interaction properties—which hint at their suitability to be used in intermittently connected environments—into five categories [16]: 1) asynchronous applications (such as email using SMTP, POP3, IMAP4); 2) distributed object synchronization (as used by, e.g., file systems, data bases, calendars); 3) interactive applications with human users (such as web access, presence and messaging); 4) real-time audio-visual communications (such as IP telephony or media streaming); and 5) DTN-aware applications.

With the exception of (interactive) audiovisual communications that require largely continuous connectivity to preserve the user experience, these applications may become workable in intermittently connected environments given sufficient support as discussed in the following. Of course, most benefit and best user experience can be expected from new applications and application protocols that may be designed to inherently take disruptive connectivity into account.

Asynchronous applications such as e-mail access (POP3, IMAP3) and transmission (SMTP) as well as applications based upon distributed object synchronization (such as calendars, offline file systems, data bases, and repositories for distributed authoring) are basically able to deal reasonably well with intermittent connectivity. However, the respective protocols expect some level of predictability for network access and particularly require that their transactions (sending an e-mail, retrieving an e-mail, synchronizing a file) complete while being connected—otherwise they are likely to repeat at least some of the operations when recovering at the next time. The operations are often not sufficiently fine-grained and may thus lead to a significant rollback of interactions which may ultimately make it impossible to complete a transaction at all if only short-lived connectivity is available. Similar considerations apply to interactive web browsing and related applications—except that (a) particularly HTTP exhibits a much higher degree of interactivity and (b), with web access, a human user is typically waiting for a synchronous response: this usually requires permanent connectivity.

Application protocols are the better suited for mapping onto a DTN infrastructure the fewer (end-to-end) interactions they require. For example, while email exchange is fundamentally asynchronous, the present application protocols for sending (SMTP) and retrieving (POP3, IMAP4) mails are fairly verbose involving numerous message exchanges and often require user credentials to be provided. With an increasing number of interactions, more complexity, application-specific knowledge, and user trust has to be put into a (mobile and/or fixed) DTN application proxy and it is likely to have to maintain application state in order to map the respective Internet application protocol to the DTN infrastructure and vice versa. For example, to interact asynchronously with a mail server a DTN POP3 proxy needs to obtain the user credentials to for authentication (that is supposed to be performed end-to-end) and it needs to maintain the POP3 state, collect emails, bundle them, and forward the bundles to the peer DTN POP3 proxy that then needs to reverse these steps. Otherwise, a DTN proxy would just map (parts of) application messages to bundles and vice versa—which would neither be efficient nor likely to be helpful in dealing with intermittent connectivity.

While we have already implemented proxy functions for SMTP and POP3 in two (client- and proxy-side) plugins for our Drive-thru Internet environment (running on top of PCMP) [32], we have chosen the more demanding access to web pages via HTTP as "worst case" for our evaluation of interactive mobile Internet access using DTN. As noted above, on one hand, HTTP is significantly more interactive and hence more demanding from a performance perspective when mapping to a DTN environment. On the other hand, HTTP operation is largely stateless for intermediaries and therefore requires only minimal application state (and no trust!) in proxies—allowing us to focus on performance aspects.

C. DTN Proxies

As outlined above, using HTTP in a DTN environment requires the use of DTN application proxies that map HTTP/TCP/IP as spoken to the web browser and the server to some bundle-based protocol used between the DTN applications.[6] One obvious goal is to minimize the—at least in terms of computational and header overhead potentially costly—exchange of bundles. Therefore, DTN proxies need to understand the application protocol (i.e., HTTP) to perform reasonable *bundling* of requests and responses.

A natural approach is to convey individual requests for immediate processing or batches of requests collected while the mobile node was disconnected in a single bundle from the mobile to the fixed DTN application proxy.[7] For the opposite direction, the fixed DTN proxy should ideally collect all resources making up a web page and forward everything needed to display the page as a single bundle back to the mobile node. Optionally, e.g., triggered by user preferences or specific request, the fixed DTN proxy may continue and collect further web pages referenced by the requested one and, again, convey these to the mobile node, e.g., in one bundle per web page.

Such an approach requires the fixed DTN application proxy to understand the content type of the received resources and interpret selected contents (particularly HTML) to look for embedded links to other resources belonging to the web page (e.g., images or frame contents). This process is also referred to as *(predictive) prefetching* [29] and also requires the DTN application proxies to act as a some minimal web cache.

It is well understood that HTTP prefetching with today's sophisticated web page design is a hard task: web pages extensively make use of cookies, JavaScript code, and dynamically generated components, to name just a few of the problem areas. Quite some effort has gone into research and product development of performance enhancing proxies (PEPs), e.g., for Internet access via satellites. Also, numerous drawbacks are well-known, including the load incurred at the web server as well as at the fixed proxy, the risk of prefetching and transmitting data that will not be used because the user has aborted displaying a particular page, and the potential of launching DoS attacks by request multiplication.

While still not perfect, reasonable results can be achieved for a broad range of web pages [33]. Furthermore, even only partially obtained contents are likely to be helpful to the user as displaying the page contents can proceed. Missing pieces are automatically retrieved in subsequent requests (albeit often without much bundling) while connected and may be marked as missing while disconnected.[8] Finally, DoS threats can be counteracted by using proper bundle (and content) authentication.

D. Implementation

We have implemented a prototype DTN application proxy for plain TCP connections and HTTP: *dtntcp*. Our implementation has been developed for Linux 2.4, is written in plain C and is based on the DTN reference implementation version 2.1.0 [34], using a snapshot from the public cvs repository from 18 May 2005. We use the *dtnd* included in the distribution as fixed and mobile DTN router.

dtntcp is designed in a symmetric fashion but command line parameters can be used to configure one instance as the (mobile) client and another one as the (fixed) server. On the client side, *dtntcp* accepts incoming connections on a configured transport address and forwards the received data as bundles to a configured peer (identified by its DTN address). On the server side, *dtntcp* accepts incoming bundles, opens a new TCP connection to the target, and then forwards the data via TCP. After connection setup, data forwarding operates largely symmetrically and if one TCP connection is torn down a corresponding event is relayed via DTN to the peer.

dtntcp supports four modes of operation that can be controlled by means of command line flags: 1) With plain TCP forwarding, each piece of data received via a TCP connection is simply forwarded transparently. 2) HTTP with external proxy uses a predefined HTTP proxy (squid) on the server side to forward all HTTP messages. On the client side, the HTTP message is parsed and a complete HTTP request is placed in each bundle forwarded. On the server side, received data is simply forwarded in both directions. 3) For HTTP without external proxy, the application proxy on the server side interprets the HTTP request messages to determine the target address from the request URI and the `Host:` header and connects directly to the server. Responses from the web server are forwarded without further interpretation. 4) The same holds for HTTP prefetching but this mode additionally uses *wget* on the server side to recursively obtain all resources *wget* believes are necessary to display the respective web page.[9] The resources returned by *wget* are retrieved from the hard drive (i.e., the buffer cache) and packed into a DTN bundle for forwarding. For our measurements discussed below, we have used modes 2) and 4).

Finally, before turning our attention towards the measurements we have carried out we need to point out numerous inherent limitations of the experimental system we have set up:

[6]As HTTP natively supports proxies, inserting one for DTN translation into the path is straightforward.

[7]During the disconnection periods, the mobile DTN proxy can provide temporary feedback to the web browser on an automatically reloading page, e.g., as proposed in [30], and thereby avoid application layer timeouts.

[8]The biggest issue are web pages requiring secure access, typically implemented by running HTTP on top of TLS—which usually leads to proxies being bypassed. Obviously, end-to-end security cannot be achieved in a DTN environment as long as application protocols rely on transport layer mechanisms to do so. At this point, we consider the non-availability of secure access to web pages via DTN as a feature as this requires fixing the problem at the application layer (e.g., by an appropriate use of S/MIME).

[9]We use *wget -p -H -s -nv -o wget.status* $< Request-URI >$ to retrieve the resources and obtain an overview of the resources retrieved.

- Our *dtntcp* as well as the *dtnd* and the libraries of the DTN reference implementation are prototypes for demonstration and experimentation rather than optimized production versions. *dtnd* and the library do not support asynchronous notifications about incoming bundles, requiring *dtntcp* to perform polling. Bundles exceeding 50 KB are fragmented by *dtntcp* and re-assembled at the application layer due to limitations from the DTN implementation. Application layer fragmentation and multiplexing as well as design simplifications also increased the per-bundle header overhead. *dtntcp* copies data in memory for simplicity.
- The implementation of HTTP prefetching is only rudimentary in many ways: *wget* is not integrated into *dtntcp* but rather invoked via *fork()* and *execve()* and the retrieved resources are obtained via the file system. *wget* itself is not optimized for parallel retrieval of resources and uses only a single TCP connection at a time (unlike web browsers that usually open up to four TCP connections in parallel). *wget*'s identification of required resources is limited to those easily identifiable from elements in a simple HTML page (e.g., the link given in the `image` and `frame` elements) while resources referenced by scripts are not considered; advanced features (such as the use of cookies) is not considered either.
- Finally, a systematic performance issue is that bundles are only be created and forwarded efficiently (in terms of overhead) once the HTTP request or response is fully received. This contributes to the overall delay as data cannot be efficiently passed on while parts are still being received.

While by design limited, when used with a reasonable set of web sites, this allows us to obtain a rough feeling how HTTP over TCP compares to HTTP via DTN bundle transfer as we will discuss in the following section.

V. Validation and Evaluation

The objective of our test measurements is to assess the impact of using protocols supporting intermittent connectivity in Drive-thru Internet scenarios on the performance interactive application protocols. Our focus in this paper is on the behavior of the DTN bundle router infrastructure and we provide measurements with other infrastructures for comparison. Note, again, that due to the aforementioned experimental nature of the DTN implementation, the performance figures need further interpretation.

A. Setup

We have set up a measurement environment following the DTN proxy architecture for mobile Internet access depicted in figure 2. In our application scenario, a mobile user uses a web browser configured with a local proxy. The local proxy encapsulates HTTP requests into bundles that are routed to a next hop bundle router in the Internet region from where they are decapsulated and forwarded (via another proxy) to the corresponding origin server. Responses travel the opposite direction.

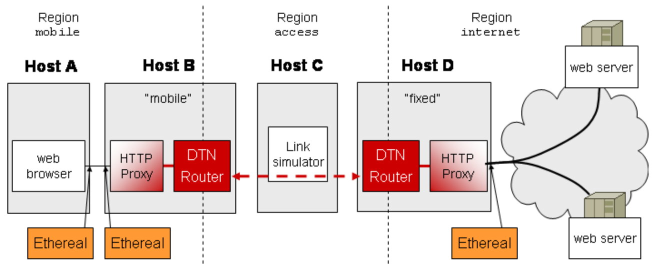

Fig. 3. Measurement setup

The measurement setup is shown in figure 3 and consists of four hosts connected with full duplex 100 MBit/s Ethernet links, all of them running Linux 2.4. The *mobile system* is represented by hosts `A` and `B`, with host `A` running the user agents and host `B` the local HTTP proxy and the mobile bundle router.[10] The fixed bundle router in the Internet region in running on host `D`. Between the two hosts, we have placed a bridging host (`C`) with two Ethernet interfaces and a configurable link simulator for simulating different network link characteristics (delay and packet loss probability). In the default configuration, the bridge transparently forwards packets from host `A` to `B` and vice versa with no extra latency. Our setup is connected to the Internet via a 155 Mbit/s access link and we use regular web servers in the Internet as peers.

On host `A`, we use *Firefox* as an HTTP user agent and measure the time for obtaining different selected web pages (an HTML document and all embedded objects such as inline images, CSS style sheets, script objects etc.). For our measurements, we use six different configurations as depicted in figure 4 below: 1) – 5) are used for detailed comparison of results while 6) is included as a reference measurement to assess how a production system could perform.

1) Direct access: The user agents are configured to not use any proxy server but to direct all requests to the corresponding origin server. We use an additional Ethernet interface connected to the Internet region.

2) HTTP-proxy: In order to be able to quantify the effects of bundle communication more precisely, we have also setup a scenario with two non-caching *Squid* proxy servers on host `B` and `D`. The objective was to measure the overhead incurred by using two additional hops without the delay caused by the processing of bundles itself. In this scenario, the user agents are configured to direct all requests to a local *Squid* proxy on host `A`, and the *Squid* proxy is configured to forward all requests to its parent proxy on host `D`, without applying any caching.

3) Drive-thru: We use the PCMP implementation developed in the Drive-thru Internet project as transport and session protocol between mobile (host `B`) and fixed (host `D`) nodes.

[10]The division into two systems allows for easily snooping the packets exchanged between `A` and `B`.

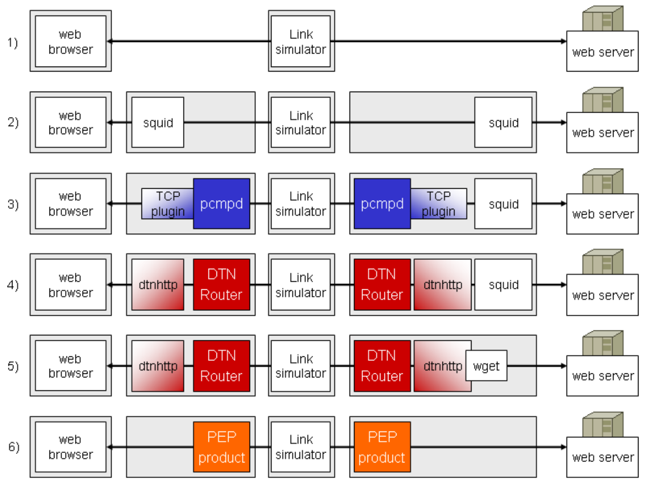

Fig. 4. Measurement overview

The PCMP implementation on host `D` performs transparent forwarding of HTTP connections to the squid proxy also running on host `D` which then interprets the HTTP requests and forwards the messages to the respective web servers.

4) DTN: The user agents route all requests to an HTTP proxy that is an application specific module of the local DTN bundle router and encapsulates all received HTTP requests to bundles to be sent to the bundle router on host `C` (via the bridge on host `B`).

5) DTN Prefetching: In this scenario, we have changed the application-specific modules in the bundle routers on host `A` and host `C`. When receiving an HTTP request, the HTTP module on host `C` fetches the requested resource and all easily identifiable embedded objects which are then aggregated into a single response bundle. This response bundle is sent to the bundle router on host `A` where it is delivered to the proxy module. The proxy module stores the received resources and then answers the original request. All subsequent requests from the user agent can then be answered locally by the proxy on host `A`.

6) Commercial HTTP Prefetching PEP: We use a pair production performance enhancing proxies that support HTTP prefetching and use an improved transport protocol for efficient communications across long-delay links. This setup is intended to assess what is *achievable* with a commercial prefetching system as opposed to our demonstrator implementation.

We have used multiple instances of Ethereal as depicted in figure 3 in order to obtain packet traces at different positions in the path.

We have used a set of selected websites with different complexity (with respect to the number of embedded objects) and different geographic and network topological locations (Germany, US, Japan, Australia).

We have carried out six rounds of measurements for all the web pages above and all of the six setups above:

We have measured the performance for accessing each web

Id	URL	Objects	RTT (ms)
DT	http://www.drive-thru-internet.org/	11	1–10
TZI	http://www.dmn.tzi.org/	11	1–10
TKK	http://www.netlab.hut.fi/	6	10–100
Werder	http://www.werder.de/index.php	65	100-500
KDDI	http://www.kddi.com/english/	37	100-1000
WIDE	http://www.wide.ad.jp/	31	100-1000
Ebay	http://www.ebay.com/	45	10–100
Apache	http://cocoon.apache.org/	21	10–100
IETF	http://www.ietf.org/	6	100–200
W3C	http://www.w3.org/	10	100–200
News	http://www.theaustralian.news.com.au/	44	10–100
Adelaide	http://www.adelaide.edu.au/	23	10–100

TABLE I

OVERVIEW OF WEB PAGES, THEIR ASSOCIATED RESOURCES, AND THE OBSERVED RTT

site in each of the configurations depicted in figure 4. In addition, we have varied each measurement by adding artificial delay to the link between host `B` and host `D`: we have created artificial delays of 0 ms and 200 ms by means of Host `C` in addition to the RTT observed via the Internet. Table I shows the web pages we have used, the respective RTTs we have measured (from Ethereal traces for the direct connection scenario), and the number of resources belonging to a web page.

B. Results

We have used the Ethereal output from our measurements taken between hosts `A` and `B` to determine the period from the initial SYN packet sent by *Firefox* to the last TCP data packet received for a web page. We did not account for the icon displayed in the browser next to the URI, nor for final TCP ACKs and for closing the TCP connection as all these do not affect the rendering process of the web page. The results are shown in table II.

Id	(1)	(2)	(3)	(4)	(5)	(4) avg
DT	0.6 s	0.9 s	0.8 s	13.1 s	2.4 s	1.2 s
TZI	0.3 s	0.5 s	0.5 s	17.0 s	3.1 s	1.6 s
TKK	0.7 s	0.9 s	0.9 s	10.6 s	2.2 s	1.8 s
Werder	16.2 s	17.0 s	16.4 s	97.0 s	81.6 s	1.5 s
KDDI	12.8 s	10.0 s	10.4 s	72.6 s	26.2 s	2.0 s
WIDE	12.4 s	6.5 s	6.3 s	45.6 s	25.8 s	1.5 s
Ebay	4.5 s	4.5 s	4.0 s	64.2 s	65.5 s	1.4 s
Apache	0.9 s	1.6 s	1.4 s	59.4 s	25.0 s	2.8 s
IETF	1.3 s	1.2 s	1.4 s	10.1 s	5.0 s	1.7 s
W3C	1.9 s	1.5 s	2.2 s	15.0 s	7.6 s	1.5 s
News	5.9 s	7.7 s	5.9 s	121.0 s	60.6 s	2.7 s
Adelaide	23.8 s	8.3 s	8.0 s	56.7 s	31.6 s	2.5 s

TABLE II

RETRIEVAL TIME FOR WEB PAGES WITH NO ARTIFICIAL LATENCY

The detailed results show that introducing proxies comes at the expense of a few 10s of milliseconds, otherwise the performance of a directly connecting to the server (1), using a pair of HTTP proxies (2), and using PCMP proxies (3) achieves roughly a similar performance. This also applies to the PEP production system (6) not shown in the table. We attribute the variations we have observed to different network (and server) load—an effect that we cannot exclude when communicating with real-world systems. This measurement set shows that introducing intermediaries, in principle, does not impact the user experience when operating in a well-connected network environment.

The performance figures for using web access via the DTN infrastructure without HTTP prefetching (4) are rather poor. The last column of the table ((4) avg.) shows the average retrieval time per web resource. This reveals that it takes 1.5 to almost 3 seconds on average to retrieve a single web object and gives a clear hint at the non-optimized DTN implementation as a source of this inefficiency.

As we can see further, using HTTP prefetching based upon *wget* increases the performance significantly in some cases but seems to provide little benefit in others. As our approach to use *wget* provides only rudimentary prefetching capabilities, looking at the cache misses after the first request, reveals that the measurements for “Werder” (20 misses out of 64 resources), “Ebay” (all resources were cache misses!), and “News” (all but one request led to a cache miss) showed particularly little improvement[11]. In contrast, web pages with only one or two misses (“DT”, “TZI”, “TKK”, “WIDE”, “IETF”) showed significant improvement.

We expect that performance would further improve if *wget* was retrieving web resources as aggressively as a web browser does. However, as *wget* uses just a single TCP connection to the web server(s) and fetches the resources one by one while web browsers use up to, e.g., four TCP connections in parallel, roughly a factor of four is expected in performance degradation. The commercial production system also used several TCP connections in parallel and, consequently, no similar degradations were observed here.

Finally, *wget* performed poorly in terms of identifying resources to prefetch thus harming performance in the well-connected case because repeated “expensive” DTN requests needed to be issued. For the disconnected case, missing (too many) objects in prefetching may lead to poor user experiences ranging from parts of a page missing (“Werder”, “KDDI”) to web pages not displaying at all (as would be the case with “Ebay” and “News”). Experience with the production systems shows that HTTP prefetching performance in terms of the fraction of necessary resources actually being prefetched properly may get close to 100% for many web pages [33]—from our observations, this appears to hold for the pages we looked at. Therefore, the approach of combining HTTP prefetching in a powerful variant with bundled information exchange appears seems a workable approach for otherwise virtually non-connected users.

The measurements with a 200 ms delay between host `B` and host `D` have largely shown that delay affects both DTN configurations to a similar extent. The DTN setup with prefetching roughly incurred a 100% increase in overall duration for accessing “DT” and “TZI” (the local web sites). We expect prefetching to perform relatively better in scenario with even larger delays (or disconnections), as can typically by experienced for satellite communications.

[11]Halfing of the retrieval duration for “News” appears to be due to changes in network congestion.

DTN Performance:

The major performance bottleneck for DTN-based communications appears to be the DTN reference implementation—which, as stated before, has not been optimized for performance or tuned in any way. In the following, examine the operation of the DTN components involved—two bundle routers and two HTTP application proxies—closer to determine their on the information exchange (and the overall latency incurred).

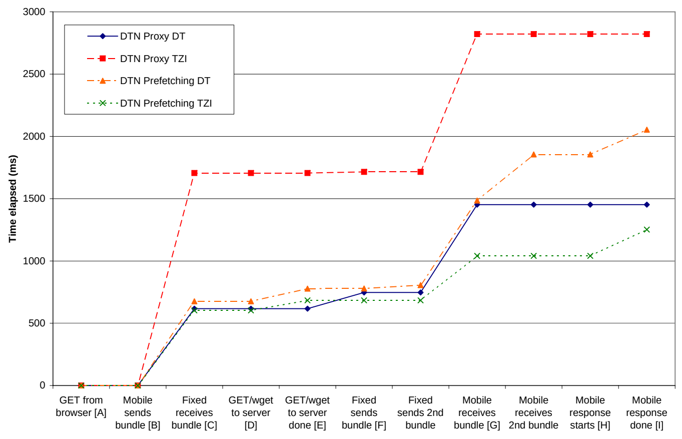

Fig. 5. Latency incurred during DTN communications

Figure 5 depicts the distribution of latency across the different components measured at the two HTTP application proxies: they record (A) at the mobile HTTP application proxy, each incoming HTTP request from a web browser, (B) its forwarding to the mobile bundle router, (C) the reception of the bundle at the fixed HTTP application proxy, (D) transmitting the request to the web server (either via *squid* or by means of *wget*, (E) completion of receiving of all the data from the web server (a single resource or the resources necessary to display a web page) at the fixed HTTP application proxy, (F) forwarding this bundle to the fixed bundle router, (G) reception of the bundle at the mobile HTTP application proxy, and (H,I) completing forwarding the first response to the web browser. Using NTP-synchronized clocks in the involved hosts, we combine the log files of the two proxies and generate a cumulative delay diagram.

We have chosen two sites for which we can largely exclude external influences (e.g., due to network congestion), “DT” and “TZI” as both web servers are hosted on the university campus. The figure clearly shows that a large fraction of the delay (some 80%) comes from DTN bundle transfer. Some 30 ms alone come from the multiple interactions between bundle router and HTTP application proxy when performing fragmentation of contents across multiple bundles.

An optimized DTN implementation is expected to minimize the incurred delay for bundle transmissions (e.g., marked by an appropriate service class). For trusted peer routers, future optimizations may even allow forwarding a bundle before it is fully received, when necessary making use of reactive fragmentation.

Figure 5 also shows the impact of using *wget* on the overall performance: using several parallel connections instead of one should be able to reduce the delay. And better prefetching functionality (as seen from the production PEP) will significantly reduce the number of DTN messages exchanged (ideally down to two per web page).

The latter aspect is crucial to the data volume exchanged. DTN-based bundle transfer incurs significant overhead: the DTN convergence layer we are using is based upon TCP and each bundle carries either a full HTTP requests including all headers or a (set of) HTTP responses, again including all HTTP headers as well as the retrieved resource, i.e. all overhead for HTTP communications is also included in a bundle and DTN-related information constitutes overhead. The bundle protocol specification [28] introduces a header overhead of 28 bytes per bundle (assuming use of the primary bundle header plus security features), the TCP convergence layer adds another 13 bytes per bundle (plus 12 bytes once for initial handshake which we neglect here and 9 bytes per bundle acknowledgement). This results in a per-bundle overhead of $41 + 9 = 50$ bytes in each direction for a single bundle-based request-response pair).

Our basic HTTP adaptation protocol adds another 56 bytes (48 of which are due to the present limitations of the underlying DTN implementation) so that we get an extra 100 bytes of overhead—20% for each HTTP GET request (which is usually 400–600 bytes in size). For the prefetching implementation, we need to identify all resources in the bundle for which we need to transmit a resource name of variable length plus the file size which may easily add up to another 30 bytes (for short resource names) to several hundred bytes.

With HTTP prefetching operating properly, we can reduce the overhead from the mobile to the fixed network ideally down to a single resource request which outweighs the extra DTN-incurred overhead. In the opposite direction, no such optimizations are possible: as resources arrive in a bundled fashion, they need to be identified individually—but more efficient encodings than our experimental one are surely conceivable. Finally, productions PEP often employ compression to the resources they convey, a function that can easily be integrated into an HTTP application proxy that gathers all the resources anyway: such an approach should allow to at least neutralize the additional DTN overhead.

VI. Switching Modes of Operation

The previous section has shown that using DTN as a communication substrate is basically feasible for HTTP but is also that this approach is clearly suboptimal. Even if no intermediaries are used, the noticeable delay before completely displaying a web page is unlikely to improve because the wide-area Internet RTT and the server load are the determining factors. They also affect even a perfect HTTP prefetching because the necessary web resources are aggregated interactively over the Internet and only afterwards forwarded as a bundle (unless the web server itself performed the bundling). While reducing the total delay because the mobile-to-fixed latency is ideally eliminated for all but one interaction, prefetching causes a noticeable initial delay before displaying a web page starts.

This all-at-once style of operation does not match typical user experience where web pages are displayed incrementally, thereby allowing the user to cancel web page retrieval prematurely, e.g. to follow an already visible link to the next resource found or to discard the page if it did not meet the user's expectations—and thus save time and network capacity. What is inevitable for disconnected operation, may hence be disturbing well-connected browsing. Furthermore, DTN-based operation will often be insufficient from a functional perspective: not all application features may be available (e.g., security), end-to-end semantics are not preserved (e.g., when sending email [16]), and disconnected operation may be limited due to the details of the application operation (e.g., with HTTP prefetching). *At this point*, this clearly calls for DTN as a fallback solution in potentially disruptive environments rather than as the standard substrate for Internet applications: interactive communications shall be optimized whenever connectivity is stable while the DTN approach should be used otherwise. This requires the mobile node (a) to determine the preferable mode of operation—*IP* or *DTN*—and (b) to switch back and forth as appropriate.

For the first part (a), this requires the mobile node to determine when it is connected to the `fixed` region in a sufficiently stable fashion. In the limited case, that the application is either running on the same machine as the access/DTN router also providing the network interfaces (and thus has access to the link state) or the latter can inform the application via local advertisements, this may be done through some heuristics, e.g., by observing the signal strength and its variation from a wireless interface, and concluding the when it is safe to operate in IP mode. Also, fixed networks attachments (e.g., Ethernet) may be considered as a hint towards stable connectivity. In case of mobile networks in vehicles, local announcements from the actual access router about its link state may provide similar information [35]. While this can be applied to the Drive-thru Internet scenario, this is obviously not a solution to the general case where intermittently connected links may be far away. It should also be noted that the notion "stability" may be highly application-specific, e.g. depending on the duration and complexity of application protocol interactions.

For the second part (b), when the stability of the connectivity changes, the application—actually: something on its behalf—needs to change the mode of operation without requiring any user intervention. For intermediary-aware application protocols such as HTTP, one option is to configure the application program to always use the mobile application proxy and then leave the decision up to the mobile application proxy. Alternatively, as is frequently done for PEPs for satellite communications, the application's IP packets may be captured transparently (i.e., without requiring any application support or configuration) and the respective TCP connections terminated at the local proxy only if the connectivity is not considered stable.[12] In both cases, the actual switching process is not carried out by the application but by the mobile application proxy instead. Whenever a change in network stability is detected (which is expected to occur rather infrequently, e.g. in the order of hours or even longer time-scales), the mobile application proxy needs to cease using one way of communication and start using the other. Information requested in one mode of operation may additionally be requested via the other. With protocols maintaining minimal state (such as HTTP), switching back and forth appears fairly straightforward as individual requests can simply be re-issued and even partial retrieval of resources (pieces of which may already have arrived) is possible—except that, when switching from IP to DTN, the requested data may not be available for some time. Application protocols that keep extensive session state and offer operations modifying that state are harder to deal with.

In any case, switching the mode of operation to DTN and sudden unavailability of network access will likely need to be dealt with by the application itself. In such a case, again, an HTTP-specific application proxy has very limited means to inform the web browser: it may delay presenting a web page by providing a temporary substitute for (parts of) the page but little further help is available to the user. Hence, ultimately, the switching mechanism should be merged into the application along with appropriate feedback mechanisms and controls for the user—an approach that could also help dealing with stateful application protocols. This essentially means that applications need to become DTN-aware in the long term even if they start out with their existing application protocols [36].

VII. DTN Deployment: Issues and Optimizations

While operating in a Drive-thru environment, the short connectivity periods require extensive use of reactive fragmentation to maximize exploitation of each connectivity island. If the mobile DTN router always communicates with the same fixed DTN router (as in the basic Drive-thru scenario described above), the TCP convergence layer may provide additional support for re-synchronization to achieve a finer granularity (as discussed for PCMP [15]), thus complementing the (frequent) interim acknowledgments for partially received bundles currently under investigation [37].

As a further optimization, the mobile node and its mobile DTN router *R-M* may benefit from additional DTN routers located in hot-spots *R-1* and *R-2* on the path towards its corresponding fixed DTN router *R-F* as depicted in figure 6 (similar to an optional Drive-thru PEP [10]). This allows a *R-M* to communicate at high data rates with, e.g., *R-1*, independent of the access link data rate available to the hot-spot: *R-F* can pass bundles quickly and have *R-1* forward these when *R-M* is no longer connected—as shown by the arrows (1) and (2) in figure 6. While this is meaningful mostly for larger bundles to be sent (e.g., emails, HTTP POST commands), small (interactive) request bundles need to be treated preferentially (e.g., using the class of service field) in order not to be stuck behind large bundles. This may require the DTN routers (not just) in hot-spots to suspend and resume the transmission of large bundles in order to interleave short ones.

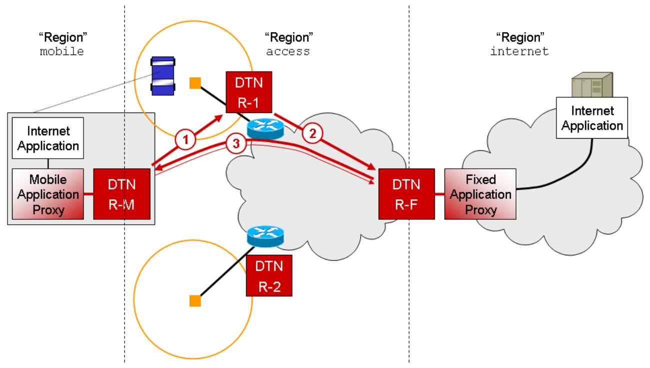

Fig. 6. Deploying additional DTN routers in hot-spots

This approach has several issues, however: Unlike *R-F* that has a trust relationship to the user, *R-1* does not and thus is more susceptible to DoS attacks. Even if no DoS attack is intended, passing mobile nodes may easily deposit significant data volumes in very short time periods that will take a long time to transmit across the access link (that usually constitutes the communication bottleneck in such a setting). *R-1* will need to apply some local policy for accepting bundles (e.g., based on the number of bundles, individual bundle or total maximum size per user). Even then, depending on the forwarding regime, bundle queues may build up and incur extra delay until a bundle gets to the fixed application proxy for processing (and responding) which is not desirable for HTTP. *R-M* has no insight into the buffer status of *R-1* and the load on the access path of the respective hot-spot, nor can it necessarily predict how far away the next opportunistic contact *R-2* is away. Hence, *R-M* may not be able to decide whether it shall forward the bundle to *R-1* or wait for the next opportunity (e.g., at *R-2*), not knowing which will be faster. Given the potentially constrained access link bandwidth, bundle duplication with short TTLs (in the order of minutes) until a response bundle is received may be an option (but rather for small bundles).

Bundle transfer from *R-F* to *R-M* is more difficult as *R-F* needs to predict the movement of the mobile node to determine the next hop for a bundle to be delivered to *R-M*. This knowledge is most likely available on the mobile node (e.g., in the application) it may be an option to not just allow a DTN application (or a bundle router) to include routing hints in the forward direction[13] but also to support specifying return routability information (e.g., routes and their expected

[12]Nevertheless, the proxy may still track the application's interactions to maintain an overview of its activities.

[13]As has recently been discussed on the DTNRG mailing list [38].

contact times) as part of the bundle protocol specification.[14] If the bundle transmission from *R-1* to *R-M* does not complete while the mobile node passes through the connectivity island, a fragmentation status report would need to be sent back to *R-F* so that it knows which parts of the bundle remain to be delivered. The question of resource utilization (particularly storage space) is similar as above. In addition, the issue of garbage collection arises: how quickly can a bundle that will no longer be delivered to a mobile node be purged? In the presence of unpredictable motion patterns and traffic situations, choosing proper TTLs is difficult for *R-F*.

A simple asymmetric routing approach may be an interesting alternative for the time being: *R-M* also establishes a transport connection to *R-F* which eliminates the prediction requirements and allows for instantaneous bundle reception (arrow (3) in figure 6). This direct transport connection may also be used to convey small (HTTP) request bundles directly, thus bypassing *R-1* and avoiding the uncertainties about its load situation.[15] The mobile application proxy needs to mark transmitted bundles according to their priority, e.g., by specifying a strict source route to *R-F* or by using the class of service field, that then needs to be reflected accordingly into *R-M's* routing tables. Given the short and unpredictable communication patterns of a mobile Drive-thru user, this seems to be the most appropriate short-term solution to mobility with resource constrained access links and hot-spot bundle routers. In the mid-term, efficient mobility support for opportunistic short-term contacts is a desirable function from the DTN infrastructure which we expect to be useful for a variety of applications other than Drive-thru-stye Internet access.

VIII. Conclusion

In this paper, we have investigated the applicability of DTN concepts to Internet access from mobile users using existing applications using HTTP as a particularly interactive and thus rather delay-intolerant example. While the DTN concepts enable effective link utilization in disconnected environments even for small messages [39], overhead can be significantly reduced and communications enabled at all, if application-protocol-specific knowledge is exploited in the proxies translating between DTN- and IP-based communications. Particularly for HTTP, prefetching techniques may be used to create large resource bundles in response to a single request: this reduces the DTN message overhead significantly but also minimizes the number of HTTP requests that have to be issued by a web browser across the Internet which also helps to reduce the impact of latency between the mobile node and the fixed network. An important next step is the integration of a more production-quality HTTP prefetching mechanism to arrive at a deployable solution and perform real-world tests on the road.

As long as connectivity is available, resource missed when prefetching simply cause additional HTTP request bundles implicitly providing a fall-back for prefetching. While disconnected, request bundling and prefetching enable asynchronous resource retrieval for delivery to the mobile node at the next opportunity. However, while disconnected, prefetching misses cannot be repaired so that some additional feedback to the user is required at the mobile node—a function that can only be fulfilled to a limited degree by the mobile application proxy. Hence, ultimately, also existing application programs, their user interfaces, and application protocols need to become DTN-aware and deal with different modes of operation, concealing disconnections from the user where possible and making them explicit where helpful. In the meantime, operating with proxies is a reasonable approximation and may provide improved service quality to mobile users. It should be noted, however, that the protocol characteristics of HTTP—allowing largely stateless operation and, except for security, avoiding dependencies on the underlying transport connections—make it rather well-suited for disconnected operation despite the highly interactive nature of web access. Similar considerations are needed one-by-one for other application protocols if they shall become usable in such a mobile setting.

For the DTN infrastructure, we find that desirable optimizations (not just) for the Drive-thru Internet case require further thoughts on the underlying concepts: While infrastructure protection is a very reasonable approach, this becomes extremely difficult with only short opportunistic contacts are available and the mobile node is not necessarily known to the DTN "access" router. Mobile DTN routers should be able to be authorized or rejected *before* transmitting a bundle so that they can fall back to their fixed DTN router if necessary. Policy decisions of a DTN access router when to accept how much bundle data also deserves further exploration: particularly with many opportunistic contacts, a mobile DTN router can benefit from estimates how quickly its bundles may percolate to their ultimate destination. In the opposite direction, getting bundles to highly mobile nodes requires to forward them to the right (set of) "candidate" DTN routers. The currently reworked naming and addressing architecture should provide means to efficiently enable such extreme mobility—e.g. by providing return routing hints as mentioned above. Finally, we have just taken a rather trivial DTN topology with partially well defined trust relationships but we believe our observations are of broader applicability and further issues may arise as soon as DTN overlays gets larger. We continue investigating the use DTN for Drive-thru Internet and beyond where we presently focus on mobility support of short-lived and unpredictable contacts and on improved support for existing applications.

[14]Given sufficient regional topological knowledge, the bundle—or fragments of it—could be replicated across multiple "adjacent" hot-spot DTN routers to increase the delivery probability. However, besides the burden of obtaining the regional knowledge, this comes at the cost of increased access link utilization.

[15]Obviously, the access link is still shared so that this helps only dealing with the internal load of *R-1*.

References

[1] S. Burleigh, V. Cerf, R. Durst, K. Fall, A. Hooke, K. Scott, and H. Weiss, "The Interplanetary Internet: a Communications Infrastructure for Mars Exploration," 53rd International Astronautical Congress, The World Space Congress 2002, October 2002.

[2] Kevin Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," Proceedings of ACM SIGCOMM 2003, Computer Communications Review, Vol 33, No 4, August 2003.

[3] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H.Weiss, "Delay-Tolerant Network Architecture," Internet Draft draft-irtf-dtnrg-arch-02.txt, Work in progress, July 2004.

[4] Website of the SeNDT project, "http://down.dsg.cs.tcd.ie/sendt/," .

[5] Andrew Parker and Arun Somasundara and Aman Kansal and Deborah Estrin and Mani Srivastava, "UCLA DTN Sensor Networks Update," Available at http://www.dtnrg.org/docs/presentations/IETF60/UCLA_DTN_Update.pdf.

[6] Avri Doria, Maria Uden, and Durga Prasad Pandey, "Providing connectivity to the saami nomadic community," in *Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Development, Bangalore, India*, December 2002.

[7] Alex Pentland, Richard Fletcher, and Amir Hasson, "DakNet: Rethinking Connectivity in Developing Nations," vol. 37, no. 1, pp. 78–83, January 2004.

[8] Website of the Mindstream project, "http://mindstream.watsmore.net/," .

[9] Jörg Ott and Dirk Kutscher, "Drive-thru Internet: IEEE 802.11b for „Automobile" Users," in *Proceedings of the IEEE Infocom 2004 Conference, Hong Kong*, March 2004.

[10] Jörg Ott and Dirk Kutscher, "The "Drive-thru" Architecture: WLAN-based Internet Access on the Road," in *Proceedings of the IEEE Semiannual Vehicular Technology Conference May 2004, Milan*, May 2004.

[11] Eva Gustafsson and Annika Jonsson, "Always Best Connected," *IEEE Wireless Communications*, vol. 10, no. 1, pp. 49–55, February 2003.

[12] Pablo Rodriguez, Rajiv Chakravorty, Julian Chesterfield, Ian Pratty, and Suman Banerjee, "MAR: A Commuter Router Infrastructure for the Mobile Internet," in *Proceedings of the ACM Mobile Systems, Applications and Services Conference (ACM Mobisys 2004)*, June 2004.

[13] Website of the OverDRIVE project, "http://www.ist-overdrive.org/," .

[14] M. Zitterbart, K. Weniger, O. Stanze, S. Aust, M. Frank, M. Gerharz, R. Gloger, C. Görg, I. Gruber, S. Hischke, P. James, H. Li, C. Pampu, C. de Waal, W. Weiß, D. Westhoff, J. Wu, D. Yu, and X. Xu, "IPonAir – Drahtloses Internet des nächsten Generation," PIK, Vol 26, No 4, October 2003.

[15] Jörg Ott and Dirk Kutscher, "A Disconnection-Tolerant Transport for Drive-thru Internet Environments," in *Proceedings of the IEEE Infocom 2005 Conference, Miami*, March 2005.

[16] Jörg Ott and Dirk Kutscher, "Why Seamless? Towards Exploiting WLAN-based Intermittent Connectivity on the Road," in *Proceedings of the TERENA Networking Conference, TNC 2004, Rhodes*, June 2004.

[17] Clay Shirky, "Permanet, Nearlynet, and Wireless Data," http://shirky.com/writings/permanet.html, March 2003.

[18] Roy T. Fielding, Jim Gettys, Jeffrey C. Mogul, Henry Frystyk Nielsen, Larry Masinter, Paul Leach, and Tim Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, June 1999.

[19] Drive-thru Internet Project, "http://www.drive-thru-internet.org/," .

[20] Jörg Ott and Dirk Kutscher, "Exploiting Regular Hot-Spots for Drive-thru Internet," in *Proceedings of KiVS 2005, Kaiserslautern, Germany*, March 2005.

[21] Jörg Ott, Dirk Kutscher, and Mark Koch, "Towards Automated Authentication for Mobile Users in WLAN Hot-Spots," Accepted for Publication at VTC Fall 2005.

[22] "Homepage of FleetNet," http://www.fleetnet.de/, 2003.

[23] Website of the Network on Wheels project, "http://www.network-on-wheels.de/," .

[24] Marc Bechler, Walter J. Franz, and Lars Wolf, "Mobile Internet Access in FleetNet," in *13. Fachtagung Kommunikation in verteilten Systemen, Leipzig, Germany*, April 2003.

[25] Adeel Baig, Mahbub Hassan, and Lavy Libman, "Prediction-based Recovery from Link Outages in On-Board Mobile Communication Networks," in *Proceeding of IEEE Globecom 2004*, December 2004.

[26] Yun Mao, Björn Knutsson, Honghui Lu, and Jonathan Smith, "DHARMA: Distributed Home Agent for Robust Mobile Access," in *Proceedings of the IEEE Infocom 2005 Conference, Miami*, March 2005.

[27] Sushant Jain, Kevin Fall, and Rabin Patra, "Routing in Delay Tolerant Networks," Proceedings of the ACM SIGCOMM 2004 Conference, Portland, OR, USA, 2004.

[28] Keith L. Scott and Scott C. Burleigh, "Bundle Protocol Specification," Internet Draft draft-irtf-dtnrg-bundle-spec-02.txt, Work in progress, September 2005.

[29] Venkata N. Padmanabhan and Jeffrey C. Mogul, "Using Predictive Prefetching to Improve World-Wide Web Latency," Proceedings of the ACM SIGCOMM '96 Conference, 1996.

[30] Henry Chang, Carl Tait, Norman Cohen, Moshe Shapiro, Steve Mastrianni, Rick Floyd, Barron Housel, and David Lindquist, "Web browsing in a wireless environment: Disconnected and asynchronous operation in artour web express," Proceedings of ACM MOBICOM 97, Budapest, Hungary, 1997.

[31] James J. Kistler and M. Satyanarayanan, "Disconncted operation in the coda file system," in *ACM Transactions on Computer Systems*, February 1992, vol. 10.

[32] Sven Häfker, "Realisierung einer Proxy-basierten Anwendungsinfrastruktur für Netze mit intermittierender Konnektivität," Diploma Thesis, Univeristät Bremen, April 2005.

[33] Nils Seifert, "A Pragmatic Approach for "d-burdened" Internet Services," in *Presentation at the Dagstuhl Seminar on Disruption Tolerant Networking*, April 2005.

[34] DTN Reference Implementation, "http://www.dtnrg.org/wiki/Code," .

[35] Jörg Ott and Dirk Kutscher, "A Mobile Access Gateway for Managing Intermittent Connectivity," June 2005, Accepted for publication in the Proceedings of the IST Mobile and Wireless Communication Summit 2005.

[36] Carsten Bormann, Dirk Kutscher, and Jörg Ott, "Disruption Tolerance: The Near End," in *Presentation at the Dagstuhl Seminar on Disruption Tolerant Networking*, April 2005.

[37] Michael Demmer, "Personal communication," March 2005.

[38] DTN Website, "http://www.dtnrg.org/," .

[39] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, and R. Patra, "Implementing Delay Tolerant Networking," Tech. Rep., IRB-TR-04-020, Intel Corporation, December 2004.