

A Disconnection-Tolerant Transport for Drive-thru Internet Environments

Jörg Ott
Dirk Kutscher

Technologiezentrum Informatik (TZI), Universität Bremen,
Postfach 330440, 28334 Bremen, Germany
Email: {jo|dku}@tzi.uni-bremen.de

Abstract—Today’s mobile, wireless, and ad-hoc communications often exhibit extreme characteristics challenging assumptions underlying the traditional way of end-to-end communication protocol design in the Internet. One specific scenario is Internet access from moving vehicles on the road as we are researching in the Drive-thru Internet project. Using wireless LAN as a broadly available access technology leads to intermittent—largely unpredictable and usually short-lived—connectivity, yet providing high performance while available. To allow Internet applications to deal reasonably well with such intermittent connectivity patterns, we have introduced a supportive Drive-thru architecture. A key component is a “session” protocol offering persistent end-to-end communications even in the presence of interruptions. In this paper, we present the design of the Persistent Connectivity Management Protocol (PCMP) and report on findings from our implementation.

I. INTRODUCTION

The reach of the Internet in today’s world is ever expanding. New link layer technologies are constantly emerging and allow to connect remote locations as well as nomadic users. However, these new technologies and deployment scenarios may exhibit communication characteristics that (traditional) Internet protocols and applications are not designed for. Wireless networks may suffer from packet losses due to bit errors (rather than congestion), thereby impacting TCP’s performance; networks with a high bandwidth-delay product raise other issues with TCP performance; mobile ad-hoc networks are too dynamic for traditional routing schemes; and most Internet applications are unsuitable for intermittently connected networks, to name just a few examples.

Serving nomadic users needs to deal with various aspects of challenged networks and has been a particular focus of interest in networking for quite some time. Wireless local and wide area networking technologies

have been developed and concepts for their integration have been devised to keep users “always best connected” [1] in a hybrid networking environment. In fact, the majority of the approaches to support mobile users strive for some form of ubiquitous connectivity in 3G and 4G networks. Such *permanets* [2], however, are expensive to build and maintain (and hence costly to use), often at relatively poor performance. The authors’ practical experience based upon GSM and GPRS networks in Europe and in the US suggests that coverage is not ubiquitous today and will not be in the near future.

Fortunately, ubiquitous access is not needed—at least not for many applications as we will discuss below. In contrast, *nearlynets* provide high performance connectivity at little cost in limited geographic areas, the most prominent example today being IEEE 802.11 WLAN (with farther reaching networks such as IEEE 802.16 and 802.20 on the horizon). However, so far WLANs are basically only used to serve rather stationary users (in cafes, hotels, conference centers, and the like) or as means to connect users to a local onboard network, e.g., on trains or buses [3].

In the *Drive-thru Internet* project, we take a different approach to mobility support and trade off ubiquity for pricing and performance. We base network access for mobile users in moving vehicles (cars, trains, etc.) on WLAN technology deployed, e.g., at the road side, in rest areas, or at gas stations. Recent trends in hot-spot provisioning show that we may expect to have network access available at reasonable density [4].

Our approach allows applications to deal with the intermittent connectivity, exploiting even short connectivity periods whenever they are available. The *Persistent Connection Management Protocol* (PCMP) that we present in this paper is a crucial element in the overall *Drive-thru Internet* architecture: it is employed by *Drive-*

thru proxies and *Drive-thru clients* in order to maintain application sessions despite connectivity interruptions.

This paper is organized as follows: first, in section II, we outline the Drive-thru architecture that supports mobile users in vehicles in accessing Internet content, discuss characteristics, and derive requirements of such scenarios. In section III, we review related work addressing specifics of wireless networks and intermittently connected users. In section IV, we describe the design of PCMP, a “session” protocol that provides a persistent transport across connectivity gaps. We briefly review our experience from its implementation and testing in section V. Section VI concludes this paper summarizing our findings and pointing out areas of future work.

II. DRIVE-THRU INTERNET

The left hand side of figure 1 depicts a scenario that is commonplace today: basic Internet access is provided by connectivity islands, each established by one (or more) access points and operated by some Wireless Internet Service Provider (WISP). The connectivity islands are independently managed, so that we expect different (W)ISPs and access networks to be chosen and private address spaces and NATs to prevail. Access to these wireless networks is usually governed through commonly accepted (mostly web-based) authentication techniques [5], but access points for free Internet access may be available, too [6].

As noted above, nomadic users are expected to remain in a hot-spot area as long as necessary to configure their mobile devices and run their respective Internet applications: typically e-mail and web browsing, VPN access to corporate networks and databases, and, increasingly, IP telephony, media streaming and chat [7]. With Drive-thru Internet, we aim at enabling mobile users to automatically detect and access a wireless network while walking, driving, or otherwise passing through a hot-spot even at significant speed when only a short period of connectivity may be available [8].

A. The Drive-thru Architecture

In order to enable useful communications in such environments of “extreme” intermittent connectivity and mobility, as shown in figure 1, we apply an enhanced variant of *connection splitting* and introduce two intermediaries—the *Drive-thru client* on or co-located with the mobile node and the *Drive-thru proxy* somewhere (well-connected) in the fixed network—to shield the respective application entities (usually clients and servers) from the intermittent nature of connectivity.

Furthermore, a *Drive-thru PEP* may be placed in a WLAN hot-spot to separate the characteristics of the wireless and the wired networks [9]. These three Drive-thru components and their communication relationships are depicted in more detail in figure 2.

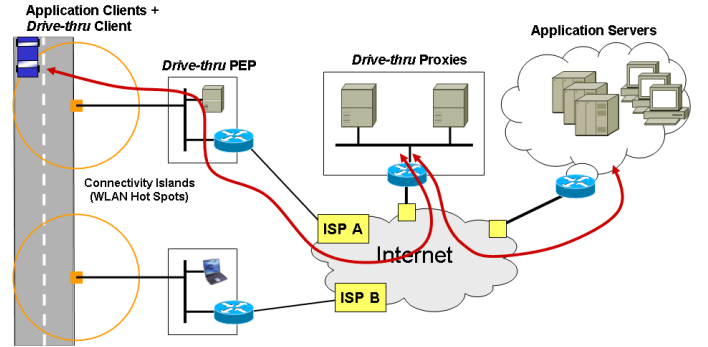


Fig. 1. Drive-thru architecture overview

The *Drive-thru client* is a transport layer proxy (presently for TCP connections). It is responsible for maintaining persistent connections with the Drive-thru proxy in the fixed network across connectivity islands using the “session” protocol described in section IV. In particular, it is the Drive-thru client that detects (loss of) connectivity and takes the necessary action to resume communication sessions whenever possible. At the application layer, the Drive-thru client provides a mechanism for application-specific plug-ins that may implement additional functionality and thus act as an application layer gateway (ALG) for different application protocols such as HTTP, SMTP, POP3, etc. The Drive-thru client uses the aforementioned standard protocols to communicate with its co-located applications entities and thus can either be run on a user’s device (e.g., a laptop) or can be realized as part of a vehicle’s communication platform.

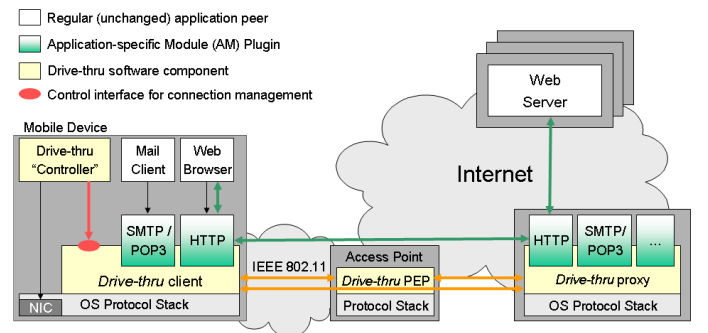


Fig. 2. Interaction between Drive-thru components

The *Drive-thru proxy* is the counter-part of the Drive-thru client in the fixed network and conceals the mobile node's temporary unavailability from the corresponding (fixed) application peers (e.g., web or mail servers). The Drive-thru proxy passively awaits new or resumed transport connections from the Drive-thru client. While the Drive-thru client is connected, the Drive-thru proxy relays information to and from the fixed network; during disconnection periods, it continues communications with the fixed network where necessary and buffers data for the mobile node.

The Drive-thru proxy may also implement application-specific functionality using a plug-in concept similar to the Drive-thru client. The Drive-thru proxy runs standard protocols to communicate with the servers/peers in the fixed network. It may be located anywhere, the sole requirements are a public IP address and an Internet link with sufficient capacity so that it does not become the communication bottleneck.

The *Drive-thru PEP* is an optional component that, if present, must be located in a hot-spot. Its task is to perform connection splitting for the underlying transport so that the network characteristics of the WLAN and the fixed network are isolated. Numerous future enhancements to its functionality are conceivable (including application-specific functions) but, for the time being, we restrict its use to optimizing throughput at the transport layer.

The *mobile application peers* (e.g., web browser or mail client) are the user's unmodified standard applications. Depending on the application, they may need to provide explicit support proxies or application layer gateways and to be configurable accordingly.¹

For *fixed application peers*, not even a re-configuration is needed; they remain entirely unchanged. They communicate with the Drive-thru proxy just as they do with any other peer. It is the Drive-thru entities' responsibility to preserve end-to-end semantics of application layer protocols (e.g., successful submission of an e-mail) as much as possible—possibly requiring additional means for user notification (e.g., to indicate to the user when a previously buffered e-mail was successfully delivered to the outgoing SMTP server in the fixed network).

B. Communication Characteristics

We have performed several laboratory tests and so far twelve measurements series on German highways

¹In addition to the explicit use of intermediaries such as HTTP proxies and SMTP relays, transparent TCP connection capturing could be employed to support intermediary-unaware applications.

and autobahns for IEEE 802.11b and 802.11g. The objective was to validate the feasibility of Drive-thru Internet access at higher speeds, to analyze the link layer characteristics of IEEE 802.11 in these mobile scenarios, and to evaluate the performance of UDP and TCP under the observed conditions. For the initial 802.11b measurements, we did not use elaborate equipment and focused on basic feasibility of our approach. These tests showed that moving vehicles can obtain Internet connectivity via a single access point at velocities ranging from 40 km/h to 180 km/h for a total distance of some 800 meters, allowing data transfers of several megabytes via both TCP and UDP from as well as to the mobile node [10].

Subsequent measurements using IEEE 802.11g equipment with range extending antennae confirmed the basic findings and showed significant improvements of the observed performance [9]: the reach of the WLAN connectivity island created by a single access point extended to a total diameter of some 1800 meters, with 300-400 meters usable at the maximum link layer transmission rate of 54 Mbit/s—yielding a peak net rate of 16 MBit/s for TCP applications, which roughly corresponds to some 80% of our laboratory measurement results. At 120 km/h, a minimum data volume of 30 MB could be transmitted in virtually all test runs in a single pass.

From our measurements, we identified a three phase model based upon the transmission characteristics for both UDP and TCP depicted in figure 3²: after associating with the access point, the throughput increases for several seconds during an *entry phase* with delays and packet losses still being significant [10]. After some time, transmission quality stabilizes in the *production phase*, during which packet losses are rare and delays short (i.e., no link layer retransmissions needed, no queues building up). This phase may last for up to thousand meters and significantly contributes to the TCP goodput. Afterwards, the link quality drops again over a long period of time in the *exit phase* until connectivity is lost.

In test runs carried out from March through September 2004, again with IEEE 802.11g equipment, we have investigated situations with different real-world conditions for Drive-thru Internet access.³ We can summarize some of our key findings as follows:

- Depending on the traffic situation, we were able to

²The figure shows measurements using our IEEE 802.11g equipment at speeds of 120 km/h. The roughly equidistant (in intervals of 10 seconds) drops in transmission rate stem from the WLAN driver and/or firmware used in this measurement.

³Because of the unstable weather conditions, the impact of rain on WLAN access on the road still requires systematic investigation.

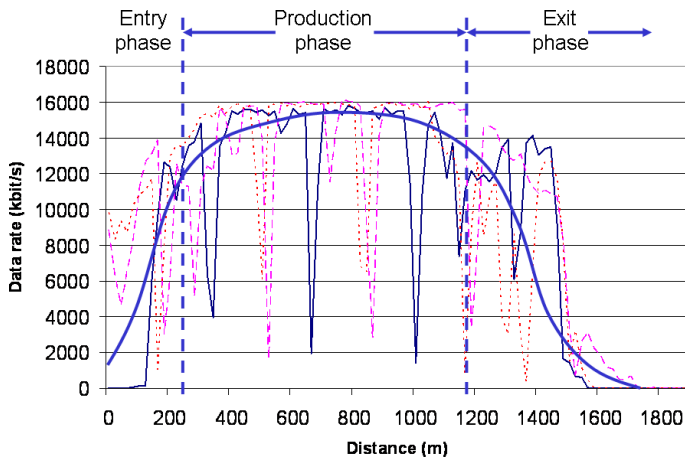


Fig. 3. Three-phase model for WLAN connectivity on the road

transmit a total data volumes per pass ranging from 30 MB (continuous flow of trucks and cars on two lanes in each direction at inter-vehicle distances of 50 m and less) to more than 70 MB (fairly empty road with one or two vehicles every few hundred meters on average).

- The TCP data rate and total data volume was independent of the transmission direction (sending fixed to mobile or mobile to fixed).
- TCP background traffic of a stationary node (one or two TCP streams at maximum possible transmission rate) shared the WLAN resources fairly with a mobile node in a passing car.
- Two passing cars in distances ranging from 30m to some 400m also shared the available access bandwidth in a fair fashion yielding 20MB-55MB per car in this particular measurement set.
- DHCP employed for auto-configuration completed early in the entry phase (even in case of packet losses due to background traffic) thus providing full IP-layer connectivity before the start of the production phase. In [8] we have also presented an automated user authentication approach and have shown that the complete auto-configuration and authentication step can complete in the first few seconds after connectivity has been established.
- Measurements and simulations with different access link types for connecting the hot-spot to the Internet (DSL, ISDN, satellite links) in [8] have shown that today's available terrestrial access technologies are suitable to provide Drive-thru Internet services in regular WLAN hot-spots. As long as queues in the access routers (both at ISPs and in the hot-spot) are

kept short so that the RTT remains low, there is no need for performance enhancing proxies (PEPs) in the Drive-thru connectivity island: with terrestrial access networks, the variations of the wireless link's communication characteristics can be handled "end-to-end", i.e., between Drive-thru client and proxy.

Figure 4 compares the total data volume transmitted during a single pass: for a plain TCP connection, for different types of background traffic, and for two cars passing. As can be seen, all cases show that WLAN-based connectivity islands are able to provide sufficient throughput to passing cars even in the presence of significant background traffic or multiple mobile nodes.⁴

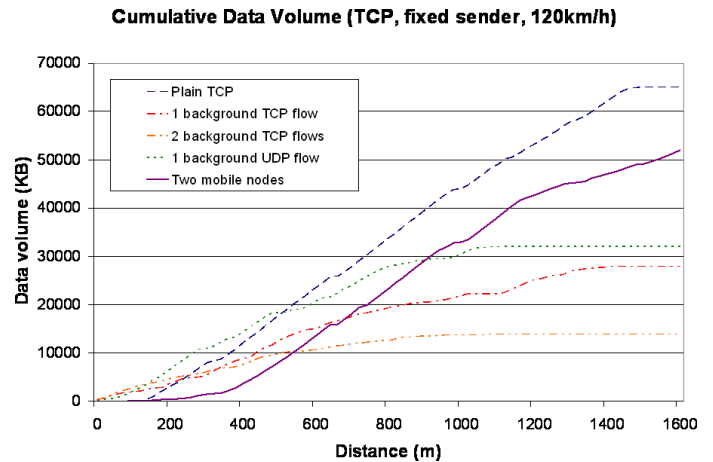


Fig. 4. Transmitted data volume in a single pass at 120km/h

In summary, IEEE 802.11 WLAN is capable of offering powerful network access services to mobile users even when passing through at high speed. Assuming that WLAN link layer detection, autoconfiguration (e.g., via DHCP), and auto-authentication (e.g., via the Universal Access Method conventions defined in [5]) become workable [8], the final prerequisite for exploiting this kind of connectivity, is that the applications are capable of dealing with such potentially short and unpredictable connectivity periods.

C. Drive-thru Applications

Several classes of today's typical Internet applications can be distinguished: asynchronous applications, web applications, audiovisual (and other real-time) communications, and distributed object synchronization. With the exception of audiovisual communications that require largely continuous connectivity to preserve the

⁴Giving precedence to traffic of passing nodes over stationary ones is subject to ongoing research.

user experience⁵, all these applications are workable given sufficient support and a few considerations as discussed in the following. Of course, new applications and application protocols may be designed to inherently take disruptive connectivity into account and/or provide specific services to mobile users [7].

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 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. Also, 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).

Except for the higher degree of interactivity, the same considerations apply to web browsing and related applications. While the aforementioned asynchronous applications are designed for offline use (and users are accustomed to this), users expect virtually immediate responses from web applications. To better conceal the discontinuous nature of connectivity, for example, prediction/prefetching schemes [12] or push techniques combined with caching may be employed—which nevertheless remain probabilistic. The latter types of support imply application-specific infrastructure components somewhere “in the network”.

Independent of the application class, effectively using short connectivity periods requires the applications to be triggered just in time to initiate their respective interactions when the mobile node enters a connectivity island so that the major data exchange can take place during the production phase. If a transaction cannot be completed within a single connectivity period, the transport layer connection may need to be preserved and resumed in the next connectivity island—which, in turn, makes further application protocol-specific considerations necessary: for example, application layer timeouts may interfere

leading to transactions being aborted and errors reported to the user.

Finally, all applications depend on the availability and responsiveness of their remote peer (e.g., a server in the Internet). Observations with mail servers made by the authors have shown that those spend a significant fraction of mail download time with retrieving the respective mailbox during the initialization phase (particularly, if the user keeps copies of e-mails on the server). It is not unusual that 60 seconds or more pass before the server is ready to react to user commands—a time span during which a car may have well passed through a connectivity island. Similar considerations apply to heavily loaded web (and other) servers. And, besides its load, congestion on the server side access link may affect a server’s responsiveness.

D. Session Layer Requirements

In [9], we have discussed general requirements on the Drive-thru architecture: particularly to be independent of hot-spot architectures and (W)ISPs and, as mentioned above, to support existing operating systems and existing applications without modification. With this, we have also motivated the case for an intermediary-based solution to conceal the intermittent nature of connectivity from user applications and servers. To achieve this concealment, we have chosen to provide a dedicated session protocol between Drive-thru client and proxy that allows arbitrary applications to maintain their transport connections persistently across connectivity periods and in spite of sudden failures.

From the above application considerations, we have gathered the following requirements for such a session protocol in a Drive-thru environment:

- The protocol must provide persistent connections across arbitrarily short connectivity periods and offer reliable communications.
- The protocol must not rely on link or network layer mobility mechanisms (such mechanisms are expected to be orthogonal).
- The protocol must be able to deal with changing IP addresses and port numbers and, therefore, must provide network and transport layer independent identification of endpoints.
- The protocol must be able to support any number of applications and application connections in parallel.
- The management (set up, suspend, resume, tear-down) of end-to-end transport/application connections must be efficient and incur as few round-trips

⁵Alternatively, the service model (and thus the user experience) can be adapted to match scenarios with intermittent connectivity, e.g., as presented in [11] for SIP-based voice applications.

as possible.⁶

- The protocol must allow for application-specific hooks (application layer gateways, ALGs) to be introduced at both Drive-thru client and proxy.
- The protocol should support suspending and resuming application connections independently to allow the user or applications to decide about relative priorities of applications.
- The protocol must support strong user authentication, as system resources will be occupied at the Drive-thru Proxy.
- The protocol must support confidentiality for those applications that cannot rely on end-to-end application layer security.
- The protocol should allow for providing hints for cleaning up state at the Drive-thru proxy.
- The protocol must be able to work with performance enhancing proxies (PEPs) that operate at the link, network, or transport layer.

Finally, the session protocol should use existing underlying transport protocols and be kept independent of these so that suitable candidates can be selected. The underlying transport is expected to provide the necessary reliability, flow, and congestion control mechanisms to enable efficient and fair communications.

III. RELATED WORK

The Drive-thru environment requires dealing with numerous aspects of challenged networks at the same time: error-prone wireless communications, connectivity changing in a rapid and unpredictable fashion, user mobility with changing network addresses, and intermittent connectivity. A variety of work has been carried out on these subjects, particularly regarding transport layer (mostly TCP) optimizations for mobile users and wireless links.

Improving TCP performance for wireless links (where packet losses may be due to bit errors rather than being caused by congestion) is addressed in various ways. Optimizations at the (TCP-aware) link layer including the *Snoop* protocol [13] and *WTCP* [14] are invisible

⁶For efficiency, a protocol implementation should support indications about changes in link layer connectivity and be able to interact with system autoconfiguration (and, ultimately, the mobile node's authentication mechanisms). We have chosen to provide a control interface (depicted in figure 2) that allows a "controller" application to trigger protocol actions (such as suspending and resuming communications with the Drive-thru proxy). Such a Drive-thru controller may monitor network interfaces, react to explicit user input, etc. Further considerations, however, are beyond the scope of this paper.

to the end points: a dedicated agent on the path between a mobile and a fixed station (usually in the base station or access point) monitors (*snoops* on) the TCP communication, buffers TCP segments and transparently provides retransmissions. The TCP peers are thus shielded from segment loss that can be repaired by the intermediary agent without loss of end-to-end semantics.

Split connection approaches operate at the transport layer and, in the simplest case, link two independent TCP connections, one for the wired and one for the wireless side [15]. This allows each TCP connection to tune its operating parameters to the respective environment (fixed vs. wireless), to perform this adaptation efficiently (particularly for the WLAN side due to the comparably low RTT from the access point to the mobile node), and even to negotiate different (optimized) TCP options for the respective links (see, e.g., *SCPS-TP* for satellite links [16]).

To address efficiency across the wireless link even better, numerous split connection approaches substitute TCP on the wireless link by a different protocol: *MSOCKS* [17] provide a remote socket interface via the wireless link, with the option of substituting the entire TCP/IP protocol suite with a specifically optimized one. *I-TCP* [18] additionally addresses handover of a Wireless Host between two Mobile Support Routers: it is capable of concealing losses during this handover period but requires explicit connection state transfer from one Mobile Support Router to the next. *Mobile-TCP* [19] uses a dedicated wireless transport to reduce the processing load and power consumption on the mobile device, make most efficient use of the scarce mobile bandwidth, and also supports the handover process.

Another class of protocols offers mobility support at the transport layer. For example, the approach to reliable network connections [20] allows for changing IP addresses at either endpoint and maintain TCP connections state across those addresses. *Migratory TCP* [21] supports transport and application layer state migration between servers (rather than clients) when a client moves to a different network and can potentially be better served by a different peer. A transport-protocol independent approach for maintaining connections in spite of changing IP addresses is being defined in the IETF as the *Host Identification Protocol (HIP)* [22].

Intermittent connectivity may also be supported at the transport layer: *M-TCP* [23] and *Freeze TCP* [24] allow TCP connections to survive blackout periods. The *M-TCP* approach uses an intermediary communicating via a dedicated protocol (*M-TCP*) with the mobile node

and is capable of freezing the TCP connection to the fixed network in case the mobile node loses connectivity (*persistent mode*). Freeze TCP achieves the same functionality end-to-end but obviously requires sufficient time to communicate the respective information. Those approaches, however, do not consider implications of outages at the application layer.

Performance enhancing proxies (PEPs) [15] may be used with split connection approaches not just to adapt to the characteristics of the wireless link but also to help concealing intermittent connectivity from the mobile user. Predictive prefetching of web pages (e.g., [12]) may be employed for advance retrieval of content as may be push techniques for cache prefilling to move contents closer to the mobile nodes. Such approaches may help (not just for web traffic) to make best use of the short connectivity periods and provide contents ahead of time so that it can be accessed when no longer connected.

Maintaining session contexts in the presence of intermittent transport layer connectivity is typically considered a session layer service, e.g., as historically addressed by the OSI X.225 session layer protocol. Conceptually, a session is characterized by a session identifier, and a session layer protocol provides the notion of synchronization points in order to enable resuming sessions after loosing an underlying transport connection. The Wireless Session Protocol (WSP, [25]) of the WAP suite is a specific implementation of a session protocol—targeted at the WAP domain and thus not directly applicable to the general Internet environment. In the WAP architecture, WSP and WTP (Wireless Transaction Protocol [26]) are used as communication substrate between a mobile device and a WAP gateway, which translates WSP and WTP to HTTP.

Specific support for certain (legacy) applications such as web browsing in mobile vehicular environments is provided by the MOCCA architecture developed in the FleetNet project [27], albeit with a focus on ad-hoc communications, mobility-aware applications, and a dedicated fixed infrastructure.

Finally, a different communication paradigm has been proposed for networking in challenged networks with extreme communication delays and intermittent connectivity [28]. Asynchronous (e-mail-style) communication is assumed based upon which interworking between internetworks is achieved: potentially large *bundles* of data are moved between communicating peers with intermediate nodes taking up responsibility for their ultimate delivery from the source to the destination; end-to-end communication optionally only takes place at the

application layer via explicit return receipts [29].

Our design of a persistent session protocol for Drive-thru Internet presented in the next section draws on many of these ideas but deliberately keeps other aspects orthogonal. The Drive-thru architecture follows a split connection approach, currently employing standard TCP implementations on both sides (to minimize the necessary modifications and thus facilitate immediate deployment). This allows for future link layer optimizations in the access point or elsewhere in the hot-spot to improve performance. Our architecture may also substitute TCP by a different underlying transport protocol for the wireless link as long as the service of TCP is preserved. Using our Drive-thru intermediaries, we follow the transport layer approach towards supporting mobility by introducing a dedicated session protocol on top that is responsible for maintaining persistent connections in spite of changing IP addresses and intermittent connectivity. Our architecture provides the necessary hooks for application-layer enhancement (such as HTTP prefetching and caching) that may, to some degree, be modeled following the custody transfer mode of the bundle protocol specification.

IV. PCMP: PERSISTENT CONNECTION MANAGEMENT PROTOCOL

The Persistent Connection Management Protocol (PCMP) provides the service of maintaining reliable transport layer sessions (TCP sessions) in the presence of intermittent connectivity and address changes. It is used by PCMP entities, i.e., by a Drive-thru client and a Drive-thru proxy, that manage persistent connections on behalf of application components such as an SMTP client on a vehicle laptop and an SMTP server run by an e-mail service provider.

While the fundamental service of PCMP is to make transport layer sessions persistent by providing a careful transmission of bytes in TCP sessions, PCMP also provides hooks for application layer protocols that can be employed by application-specific modules (AMs) in PCMP entities. This enables a Drive-thru proxy to provide application specific functions such as careful transaction completion, proxy services, caching, and prefetching.

A. Protocol Architecture

The basic PCMP model is depicted in figure 5. A Drive-thru client accepts TCP connections from any number of clients, multiplexes them via a single Drive-thru connection (Connectivity-loss resilient connection,

CLRC), and the server demultiplexes the incoming information again into individual transport connections. PCMP currently uses standard TCP (and, optionally, TLS) as underlying transport for CLRCs but other reliable, congestion-controlled transport protocols (such as SCTP) are conceivable as well.

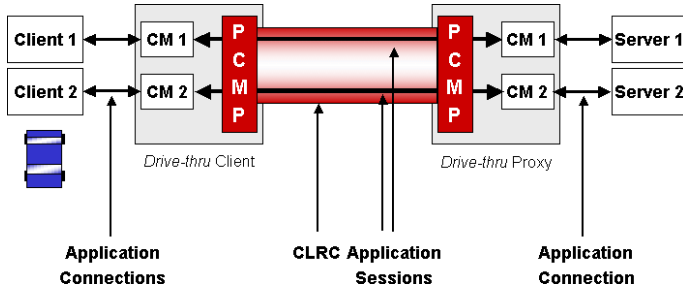


Fig. 5. The PCMP operation model

The CLRC provides an authenticated, reliable, congestion-controlled channel for bidirectional communication and can host multiple application transport sessions. The transport sessions inside the CLRC are explicitly created, suspended, resumed and finally removed; they are independent of the CLRC they are created in: once the TCP connection between a Drive-thru client and the Drive-thru proxy fails, the CLRC implicitly terminates, but the transport sessions continue to exist. When re-connecting, the PCMP peers establish a new CLRC, in which the persistent transport sessions can be explicitly resumed. In addition to explicit termination, a timeout mechanism can be employed to automatically remove transport sessions, e.g., when the client has not re-connected for several hours.

Decoupling of CLRCs and the persistent transport sessions bears several advantages: The CLRC provides the fundamental transport services for the PCMP peers, including authentication, reliability, congestion control and, potentially, encryption and data compression. Transport sessions can be instantiated efficiently, and all transport sessions are running under a common congestion control regime. The decoupling of CLRCs and transport sessions also allows a Drive-thru client to deliberately instantiate multiple CLRCs, e.g., one per network interface, and assign transport sessions to certain CLRCs and network interfaces. Explicit resuming of transport sessions after a re-connection removes the need for actively probing each transport session upon CLRC establishment: instead, transport sessions are resumed when they are needed, i.e., when one peers wants to send data, and are otherwise terminated when the correspond-

ing TCP connections are closed or when the timeout has occurred.

The setup of “end-to-end” connections occurs in two steps. First, the CLRC is established (either triggered by an incoming client connection or by link layer detection of an available network). The establishment of the CLRC involves full (client and server) authentication potentially incurring multiple round-trips. This stage is also used to create a shared context for reliability, confidentiality, compression, and other functions at both peers.

In the next step, individual transport sessions are set up. Each application connection intended to run from a client to a server is mapped to a single session within the CLRC. The session setup procedure is simplified and does not require additional round-trips. For each individual transport session, both parties maintain a session state. For example, it is monitored how many bytes have been transmitted in each direction in order to allow for a seamless resuming should the session be paused or the CLRC be closed or interrupted.

B. Names and Addresses

In a PCMP session, there are several entities that have to be identified:

- the Drive-thru client has to identify and authenticate the Drive-thru server and vice versa;
- within CLRCs, persistent transport sessions have to be identified, e.g., in order to resume a transport session after CLRC re-establishment; and
- the application client has to identify the remote application server to ultimately connect to.

PCMP is intended to be used in an environment where client IP addresses may change frequently, e.g., resulting from a new network attachment in a WLAN hot-spot. Hence, IP addresses cannot be used as unique client identifiers. Instead, we use the concept of “peer names”—globally unique identifiers that are independent of a host’s IP address. For the current version of the protocol specification, peer names are represented as host names, e.g., `on-the-road.example.com`⁷ but other types of host identifiers (such as public keys as used by HIP [22]) are conceivable, too.

When initiating a CLRC, both peers exchange their identifiers, and, based on these identifiers, a 32 bit CLRC ID is calculated that is later used to identify messages pertaining to the CLRC context.

⁷It should be noted that these do not have to be resolvable but are merely used to identify endpoints.

The creation of transport sessions within a CLRC is parameterized with a *session name* (provided by the session initiator) that uniquely identifies the session. Because transport sessions may be suspended in a specific CLRC and resumed in another CLRC, e.g., after a disconnection and re-connection cycle, the session names have to be globally unique. Within the current CLRC, the session is identified with a 32-bit session identifier (also chosen by the initiator upon setup or resumption).

The actual communication endpoint addresses for transport sessions (*target addresses*) are also provided at transport session setup. A target address may be an arbitrary URL or a tuple providing an IP address or DNS name and a port number. In addition, the desired transport protocol and (if applicable) the application layer protocol can be specified.

C. Transport Session Management

The actual persistence management relies on the two PCMP peers tracking the state of the CLRC and the different application sessions. PCMP provides a reliable transport between the PCMP peers (currently TCP/TLS). A connection loss, e.g., determined by a TCP timeout or derived from a link layer indication, results in a termination of the CLRC and in an implicit suspension of the transport sessions that have been used in the CLRC.

The data transmission in a transport session employs a *record marking* principle. Each message provides a sequence number (of the first byte in the message). When a CLRC is lost due to disconnection, all embedded transport sessions are suspended. When communication is resumed in a new CLRC, e.g., because the Drive-thru client has entered the next connectivity island, the PCMP peers exchange the sequence numbers they have successfully received so far. Thereby, a peer can notice if the other peer is lacking data, and will re-send the missing bytes before moving on with the data transmission. This is accomplished by three mechanisms:

- 1) Each PCMP peer tracks the number of bytes sent and received so far (per transport session);
- 2) each PCMP peer keeps a copy of yet-to-be-acknowledged bytes in case a retransmission may be required; and
- 3) PCMP provides an explicit acknowledgment message that is intended as a reliability mechanism indicating to a sender how many bytes have been received by the peer.

Buffer management is an important function for implementing persistent transport sessions: for each transport session, a peer must maintain an independent buffer

for storing data that has been received locally, e.g., during a connectivity blackout period. There is also a buffer for buffering data that has been received over PCMP and is to be delivered to an endpoint, possibly facing congestion. In order to manage the buffer sizes, PCMP provides a flow control mechanism, implemented by a receive window.

D. Protocol Operation

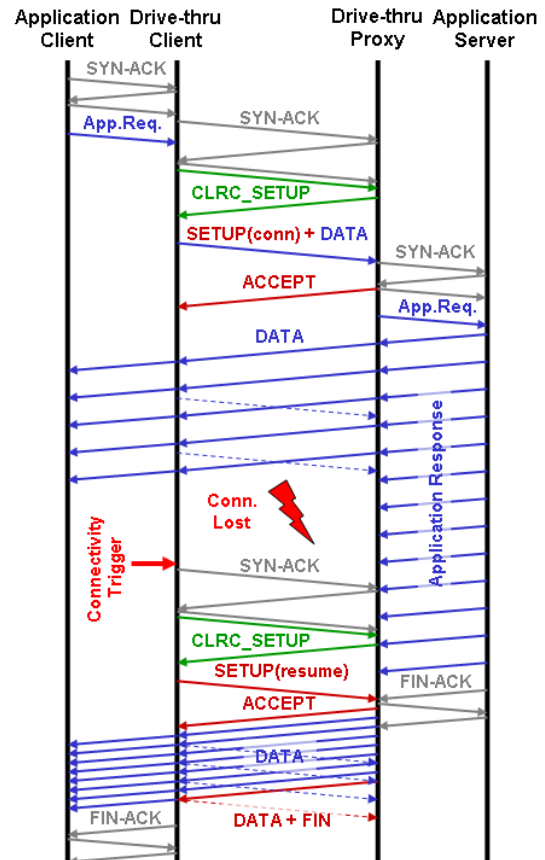


Fig. 6. The PCMP operation example

Figure 6 depicts a sample PCMP message exchange⁸: A client application is initiating a TCP connection, e.g., for an HTTP transaction, to a server in the fixed network. The application is configured to use the Drive-thru client (PCMP client) as a proxy. The Drive-thru client accepts the incoming TCP connection, and the client application transmits the first application protocol request, e.g., an HTTP GET request. As soon as connectivity is available, the Drive-thru client initiates a TCP connection to its

⁸The complete PCMP operation is described in the PCMP specification that is available at <http://www.drive-thru-internet.org/spec.html>.

configured Drive-thru proxy in the fixed network and establishes a PCMP session by creating a CLRC, and both Drive-thru client and Drive-thru proxy authenticate each other.

After the CLRC has been established, the Drive-thru client establishes a transport session for the TCP connection by specifying the desired endpoint address and by assigning a transport session ID. The setup message for the transport session can also include a payload, e.g., the first HTTP request. At the Drive-thru proxy, the transport session is associated to the *user* (not to the current CLRC). A TCP connection to the application server is set up and the result is sent back to the Drive-thru client. The data that has been received so far (the HTTP request) is sent to the application server, where it is processed and answered. The Drive-thru proxy receives data on its TCP connection to the application server and forwards it over PCMP to the Drive-thru client.

During the transmission of the response the TCP connection between the Drive-thru client and the Drive-thru proxy fails, e.g., because the mobile user leaves a WLAN connectivity island. The CLRC is terminated at both peers, but the transport session state is maintained. When connectivity has been recovered, e.g., sensed and triggered by a Drive-thru “Controller” as depicted in figure 2, the Drive-thru clients sets up a new TCP connection to the Drive-thru proxy and establishes a new CLRC. The transport session for the HTTP transaction is explicitly resumed (in this example initiated by the Drive-thru client), and the transmission of the outstanding bytes can continue.

After the response from the application server has been received by the Drive-thru proxy, the application server closes the TCP connection, which leads to a PCMP transport session tear-down, and, finally to termination of the TCP connection between Drive-thru client and application client.

E. Application-specific Support

PCMP’s fundamental persistent connection management service can be augmented by application-specific support. In general, PCMP provides persistent connections that transparently outlive disconnection phases. While this basic service is useful because users, i.e., the users’ applications, do not have to care about disconnections, lost communication contexts due to address changes, etc., additional functions may be required in order to make the persistent connection service useful for applications.

For example, many application protocol implementations employ application layer timeouts for transactions and will consider a transaction failed if no answer has been received for a certain duration despite the underlying transport association still being alive. Typically, an implementation will retry the given transaction at a later time. An application specific support module in the Drive-thru client could provide the following extensions in order to enhance the overall application behavior in these scenarios:

- 1) The Drive-thru client could notice the transaction interruption by the application and then later, upon entering the next connectivity island, proactively complete the transaction and cache the result. When the application later retries the transaction (which may or may not be within a connectivity island), the Drive-thru client can provide the result without having to complete the transaction end-to-end.
- 2) Some application layer protocols such as HTTP allow for finalizing partially completed transactions, e.g., by using the HTTP Range header in HTTP GET requests. An HTTP-enabled Drive-thru client could thus optimize the use of scarce network resources by remembering partial results from earlier requests and thus enhance the HTTP protocol operation accordingly. Also, HTTP prefetching may be employed in the Drive-thru proxy (even while the Drive-thru client is disconnected) and, at the next opportunity, the results pushed to and cached at the Drive-thru client to bundle traffic for the short connection periods.

It should be noted that application specific functions can (or need to) be implemented in both PCMP peers, i.e., the Drive-thru client and the Drive-thru proxy. Many application layer protocols already provide the notion of application layer intermediaries such as HTTP proxies, SMTP relays etc., and the PCMP peers are natural places for providing these services. For example, the Internet e-mail architecture is based on SMTP relays that provide a careful message forwarding, that has been designed for robustness in case servers become unavailable or networks become unreachable.

The usage of these intermediaries can be explicitly configured at applications. However, there are also application layer enhancements that can be applied transparently such as POP3/IMAP4 prefetching at the Drive-thru proxy and content transcoding and compression. [7] provides a taxonomy of application classes and discusses

ways of how these applications can be made to work in Drive-thru environments.

PCMP supports these application-specific functions by application protocol specifiers that can be provided by a PCMP peer upon establishing a new transport session, thus enabling peers to identify the used application layer protocol and to invoke application specific support functions.

V. IMPLEMENTATION AND TESTS

We have implemented PCMP and performed some measurements in our Drive-thru environment as well as in a simple simulation setting in the lab in order to evaluate the protocol design and to compare the performance against our first measurements.

A. Implementation Outline

So far, we have developed two PCMP implementations: a Microsoft .NET based implementation for MS Windows systems and a C++ implementation for UNIX-based systems. While we have tested both implementations in various simple nomadic scenarios (using Ethernet and WLAN access links), we have carried out simulation tests and real-world Drive-thru measurements only for the latter.

The PCMP implementations consist of two PCMP peers that are largely symmetric at the PCMP level: the PCMP client (Drive-thru client) and the PCMP server (Drive-thru proxy). They communicate via plain TCP connections (TLS is yet to be added). The client acts as a TCP relay and can be configured to establish PCMP connections to a PCMP server on a specific host. Currently, both implementations support the fundamental persistent connection service and offer a plug-in framework for application-specific support. Plugins for SMTP and POP3 are already available for the .NET implementation. Further application-specific support is under development.

Furthermore, hooks for detecting network access (as well as disconnection) are provided by means of support for external trigger events from a “controller” (see figure 2). For disconnections, the PCMP client also reacts to operating system error indications for the underlying TCP connection (as provided by MS Windows) and TCP timeouts. Without a controller, the Drive-thru periodically attempts to establish a new connection (initially and after an interruption) and determines disconnection solely by operating system means.

Application clients may use the Drive-thru client as a proxy (e.g. an HTTP proxy) and we currently rely

on the application peer to explicitly contact the Drive-thru client. While not yet implemented, connections originating from the mobile application peer may also be intercepted *transparently* so that no proxy support by the application is needed—as it is commonly done for performance enhancing proxies, e.g., for satellite networks.

We have performed a set of fundamental functional tests and have developed some test applications in order to validate the protocol design and the implementation in general. First tests involved a simulation of intermittent connectivity, e.g., by disconnecting the peers programmatically for random amounts of time. We have also implemented a simple simulation of a Drive-thru environment using a remotely controlled WLAN access point and a program script reproducing a car’s circular track on a Northern German autobahn with different connection and disconnection times.

For our initial tests and our real-world measurements, we have used our TCP traffic generator (*tcp_x*) and manually configured the PCMP entities to relay TCP connections to a specific test server. Initial test applications to validate the PCMP functionality included unidirectional TCP data transmission, web browsing (by using the PCMP client as a proxy, as described above), and a remote file system synchronization application where two application processes synchronize a distributed file system despite intermittent connectivity. We have also used regular e-mail clients running SMTP and POP3 with and without support of the corresponding plug-in.

B. Performance Measurements

We have carried out our PCMP performance measurements in the original Drive-thru scenario as described in section II. Instead of measuring the transmission characteristics for passing a single connectivity island, we have passed multiple connectivity islands with about 3 minutes of no connectivity between islands.⁹ In the lab, we have also tested PCMP with larger disconnection periods.

We have used the same tools for generating TCP traffic and used identical traffic patterns as for our single-pass tests. During the disconnection phases, the PCMP entities have maintained the TCP connections to their local peers, i.e., the PCMP client has maintained the TCP connection to the client application and the PCMP

⁹In fact, we have installed a single connectivity island and then turned the car after each passing. The disconnection period stems from the travel time from the connectivity island to the next exit and back.

server has maintained the TCP connection to the server application. As both PCMP entities have ceased to read from their TCP socket (of the local TCP connection), TCP flow control has paused sending of the local peer. After a disconnection, the PCMP client has periodically tried to establish a new connection to the PCMP server. After successful re-establishment of the connection, a new CLRC context has been created and the application session has been resumed. The PCMP peers have continued to read from the socket bound to their local TCP connection, which has led to a re-activation of the application client and server.

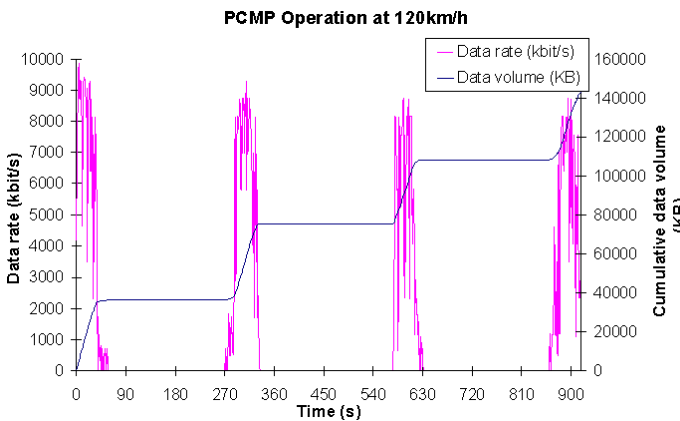


Fig. 7. PCMP throughput measurement results

Figure 7 depicts the TCP throughput for a series of Drive-thru passings in a single application session and the cumulative throughput for the complete measurement. Each Drive-thru pass is shown as a peak in the throughput graph and represents the observed throughput when passing the connectivity island at 120 km/h. As an long-term average, we have observed a throughput of about 1 MBit/s (including all disconnection phases). The individual peaks for each pass are principally similar to those observed for a single pass (as depicted in figure 3). However, the maximum transmission rate is about 10 MBit/s, while for our non-PCMP measurements, we have been able to obtain about 16 MBit/s.

This can only partly ascribed to the PCMP protocol overhead that is needed to transmit the data between the two PCMP peers: The additional PCMP header of 22 bytes makes up about 1.5% of the data traffic and the increased amount of data leads to additional IP packets being sent (thus incurring additional IP and TCP header overhead). Also, slight measurement inaccuracies are expected as our TCP test applications have been sending and receiving data segments of 1460 bytes (which leads

to a less efficient segmentation when the PCMP headers are added). An analysis of the packet traces using Ethernet shows that the major impact comes from the limited TCP window size (of just 18 KB) on the receiving side which causes regular blocking of the PCMP sender. The receiver’s TCP kernel buffer not being emptied sufficiently fast by the Drive-thru client may be due to its experimental (non-optimized) nature peered with the overhead of additional context switches when locally forwarding the data to the TCP test application.¹⁰

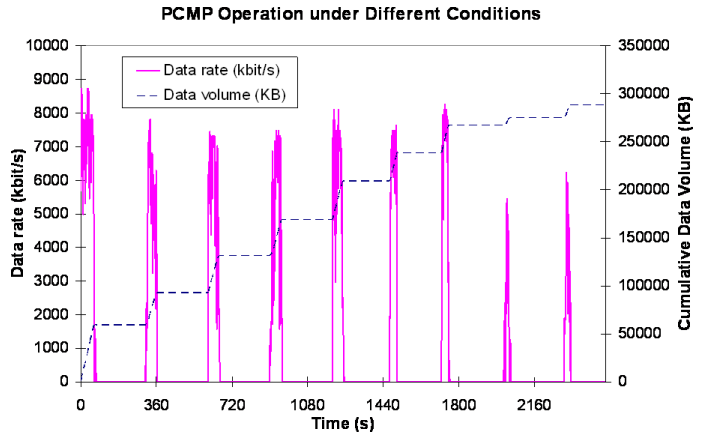


Fig. 8. PCMP throughput measurement results at different conditions

Figure 8 depicts PCMP behavior for different network and driving conditions. The last two passings were performed at 180 km/h (instead of 120 km/h)—we can identify a significant decrease in both the maximum throughput and the cumulative throughput for these passings. However, we still have achieved TCP transmission rates of about 6 MBit/s.

While these experiments are not meant to provide a statistical analysis of PCMP, they clearly show that regular applications can exploit connection chains involving two intermediaries and yet provide decent performance end-to-end in spite of intermittent and rapidly changing connectivity patterns. In addition, our qualitative tests using the same component constellation in a lab setup with simulated connectivity interruptions have proven the Drive-thru architecture workable for e-mail transmission (using SMTP) and retrieval (using POP3), for directory synchronization, and (so far without plug-in support) for

¹⁰For the total data volume, only a minimal additional penalty is expected from the PCMP operation itself: in the absence of explicit triggers, the CLRC setup retry interval incurs some extra delay and the CLRC setup handshake adds another round-trip.

HTTP retrieval.¹¹

In summary, our first PCMP tests have been quite convincing in the lab as well as on the road: We were able to maintain an application layer session throughout a cycle of network attachments and disconnections and have been able to transmit significant amounts of data and work with different existing applications. On the road, we have achieved an average throughput of 1 MBit/s—an encouraging value given the rather long phases of no connectivity (some 80 % of the operation time as per figures 7 and 8). The observed peak performance difference calls for further analysis and particularly for future optimizations in our PCMP implementation.

VI. CONCLUSION

We have presented a transport and application layer solution for disconnection tolerant networking in a particularly challenging environment. The Drive-thru Internet environment exhibits frequent disconnections, comparatively high data rates and, a diversity of network operators, network attachment procedures and network topologies.

Although we have shown that a single Drive-thru connectivity islands may allow for transmitting significant amounts of data (up to 70 MB per connectivity island), applications, i.e., users, cannot take much advantage of it without additional support. We have shown that, in many cases, the limiting factor is not the applications themselves—many transactional applications could be made to work with only little adaptations. The fundamental issue is that transport protocols and computing environments are designed with the notion of permanent (or: seamless) connectivity in mind—a concept that is in fact only realistic for controlled environments but begins to fail for any serious degree of mobility in 2G networks, let alone WLAN-based Drive-thru environments.

A set of approaches have been developed for dealing with aspects of challenged networks, ranging from improving TCP performance in heterogeneous networks by applying split connection approaches, to new host mobility solutions, to performance enhancing proxies, to protocols for session management, and to network architectures for interplanetary (and other delay-challenged) networks. We have shown that, although our approach draws on many of these ideas, none of the presented approaches is directly applicable.

¹¹In this minimalistic HTTP case, web browser timeouts are still relevant—which we seek to address by dedicated HTTP plugins in the future, e.g. as in [30].

What is needed is an automated connectivity management that liberates the user from manually attaching to WLAN hot-spots and from manually triggering application operation. The automated connectivity management must also shield transport protocols from loss of connectivity and provide support for applications in order to use connectivity phases efficiently.

The Persistent Connection Management Protocol (PCMP) is a key component in our Drive-thru Internet architecture and acts as a “session” protocol that offers persistent end-to-end communications even in the presence of interruptions. PCMP provides the service of managing intermittent connectivity for transport layer endpoints. It is used by PCMP entities, i.e., by a Drive-thru client and a Drive-thru proxy—transport and application layer intermediaries that manage persistent connections on behalf of communicating nodes and their applications.

Our measurement results from first lab and real-world tests with PCMP implementations have validated the overall approach: We have been able to maintain a persistent application session throughout a period of several disconnections and new network attachments. The observed data rate when passing through a single connectivity island under different conditions was reasonable; nevertheless, we believe that some protocol optimizations are still possible. In particular, we are looking into basing PCMP on SCTP and adding TLS support for strong (certificate-based) mutual authentication of Drive-thru client and proxy.

In addition to optimizing the PCMP specification and our implementations for performance, our next steps include the advancement of our local endpoint architecture. We are already able to automate the WLAN hot-spot association procedure for certain environments and are currently working on support for additional authentication and authorization procedures as well as on approaches for (wireless Internet) service announcements in hot-spots [8]. Another activity involves the development of a local architecture for disseminating link layer events, such as `link on` and `link off` to higher layers and applications. Moreover, we are implementing further application-specific PCMP support functions, such as HTTP prefetching and caching schemes and push-to-talk voice applications, and are developing a corresponding PCMP server architecture that allows to include new application modules on-demand. The resulting Drive-thru infrastructure should then address the most pressing applications for nomadic users and improve their working conditions on the road in spite of intermittent connectivity.

VII. ACKNOWLEDGMENTS

The authors would like to thank Dirk Meyer, Holger Bastisch and Sven Häfker for implementing PCMP and a large crowd of co-workers and master students for supporting us in our test drives.

REFERENCES

- [1] E. Gustafsson and A. Jonsson, "Always best connected," in *IEEE Wireless Communications Magazine*, February 2003.
- [2] Clay Shirky, "Permanet, Nearlynets, and Wireless Data," <http://shirky.com/writings/permanet.html>, March 2003.
- [3] TMCnet.com, "Clic TGV Brings Wifi Onboard France's High Speed Trains," available online at <http://www.tmcnet.com/submit/2004/Jan/1022655.htm>, January 2004.
- [4] Anand Balachandran, Geoffrey M. Voelker, and Paramvir Bahl, "Wireless Hotspots: Current Challenges and Future Directions," in *Proceeding of WMASH'03*, September 2003.
- [5] B. Anton, B. Bullock, and J. Short, "Best Current Practices for Wireless Internet Service Provider (WISP) Roaming, Version 1.0," Wi-Fi Alliance, February 2003.
- [6] Terry Schmidt and Anthony Townsend, "Why Wi-Fi Wants To Be Free," *Communications of the ACM*, Vol 46, No 5, 2003.
- [7] 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.
- [8] Jörg Ott and Dirk Kutscher, "Exploiting Regular Hot-Spots for Drive-thru Internet," in *Proceedings of KiVS 2005*, 2005.
- [9] 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.
- [10] 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.
- [11] Jörg Ott, Dirk Kutscher, and Olaf Bergmann, "SIP Voice Services for Intermittently Connected Users in Wireless Networks," in *Upperside Conference Wi-Fi Voice 2004*, May 2004, available at <http://www.informatik.uni-bremen.de/~jo/presentations/2004-upperside-wifi-voice-sip-intermittent.pdf>.
- [12] Venkata N. Padmanabhan and Jeffrey C. Mogul, "Using Predictive Prefetching to Improve World-Wide Web Latency," *Proceedings of the ACM SIGCOMM '96 Conference*, 1996.
- [13] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz, "Improving TCP/IP Performance over Wireless Networks," in *Proceedings of the 1st ACM International Conference on Mobile Computing and Networking (Mobicom)*, November 1995.
- [14] Prasun Sinha, Narayanan Venkitaraman, Raghupathy Sivakumar, and Vaduvur Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," in *Proceedings of ACM MOBICOM'99*, Seattle, WA, USA, 1999, pp. 231–241.
- [15] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135, June 2001.
- [16] Consultative Committee for Space Data Systems, "Space Communications Protocol Specification (SCPS) – Transport Protocol," CCSDS 714.0-B-1, available from <http://www.scps.org/>, May 1999.
- [17] David A. Maltz and Pravin Bhagwat, "MSOCKS: An Architecture for Transport Layer Mobility," *Proceedings of IEEE Infocom 1998*, 1998.
- [18] Ajay Bakre and B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," Tech. Rep., Department of Computer Science, Rutgers University, October 1994.
- [19] Zygmunt J. Haas, "Mobile-TCP: An Asymmetric Transport Protocol Design for Mobile Systems," 3rd International Workshop on Mobile Multimedia Communications, September 1995.
- [20] Victor C. Zandy and Barton P. Miller, "Reliable Network Connections," in *Proceedings of ACM MOBICOM'02*, Atlanta, GA, USA, September 2002, pp. 95–105.
- [21] Florin Sultan, Kiran Srinivasan, Deepa Iyer, and Liviu Iftode, "Migratory TCP: Highly Available Internet Services Using Connection Migration," Technical Report DCS-TR-264, Rutgers University, December 2001.
- [22] Robert Moskowitz, Pekka Nikander, Petri Jokela, and Thomas R. Henderson, "Host Identity Protocol," Internet Draft draft-ietf-hip-base-01, Work in progress, October 2004.
- [23] Kevin Brown and Suresh Singh, "M-TCP: TCP for Mobile Cellular Networks," *ACM Computer Communication Review*, vol. 27, no. 5, 1997.
- [24] Tom Goff, James Moronski, and D.S. Phatak, "Freeze-TCP: A True End-to-end TCP Enhancement Mechanism for Mobile Environments," in *Proceedings of the IEEE Infocom Conference*, 2000.
- [25] WAP Forum, "WAP WSP," July 2001, WAP-230-WSP.
- [26] WAP Forum, "WAP WTP," July 2001, WAP-224-WTP.
- [27] Marc Bechler, Walter J. Franz, and Lars Wolf, "Mobile Internet Access in FleetNet," in *13. Fachtagung Kommunikation in verteilten Systemen, Leipzig, Germany*, April 2003.
- [28] Kevin Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," *Proceedings of ACM SIGCOMM 2003*, *Computer Communications Review*, Vol 33, No 4, August 2003.
- [29] Keith L. Scott and Scott C. Burleigh, "Bundle Protocol Specification," Internet Draft draft-irtf-dtnrg-bundle-spec-02.txt, Work in progress, September 2005.
- [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.