IBR-DTN: An Efficient Implementation for Embedded Systems

Michael Doering, Sven Lahde, Johannes Morgenroth, and Lars Wolf
Institute of Operating Systems and Computer Networks, Technische Universität Braunschweig
Braunschweig, Niedersachsen, Germany
{doering|lahde|morgenro|wolf}@ibr.cs.tu-bs.de

ABSTRACT

In our demonstration we present an implementation of DTN for embedded systems and demonstrate how a WLAN access point can be turned into a stand-alone DTN-node for mobile applications. The modular software design of "IBR-DTN" is centered on the efficient use of resources and interoperability with the DTN2 reference implementation. Our modules comprise a DTN Core, Bundle Router, Persistent Storage and a Convergence Layer Manager. IBR-DTN is work in progress, but the comparison of the features and performance to DTN2 is already very promising. Finally, we present a practical evaluation in a mobile scenario in which a vehicle mounted node passes a stationary node.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Design, Performance

Keywords

Delay Tolerant Networks, Embedded Systems, IEEE 802.11, DTN2, OpenWRT, IBR-DTN

1. INTRODUCTION

The EMMA [2] project deals with environmental monitoring based on public transportation vehicles and delay tolerant networking. During the development, we realized that a commercially successful system requires an inexpensive embedded platform for the computing and communication of sensor data. Since the design and implementation of a custom embedded system is rather time consuming and a low volume production is always expensive, we decided to enhance the firmware of off-the-shelf WLAN access points with DTN-capability. These consumer devices offer many features, e.g. network address translation, port forwarding and packet filtering. Moreover, various network services like DHCP, DNS, NTP, HTTP and UPnP are implemented. Current devices are equipped with integrated IEEE 802.11 access points and USB 2.0 host interfaces which allow remote access e.g. to storage devices, printers and webcams. The majority of these multifunctional devices (termed e.g. "wireless routers" or "multimedia routers" by product marketing) are running an embedded version of Linux, although it is normally not accessible to the user and well hidden behind a web-based configuration interface. Hardware- and manufacturer-specific modifications of the Linux kernel are usually not documented and due to the lack of common standards it is virtually impossible to implement portable software packages. This problem is addressed by the OpenWrt [5] project, maintaining a well defined, manufacturer and model independent Linux-based firmware, build environment, and a software package management system. It is not limited to consumer devices, but also runs on standard PCs as well as on professional networking equipment.

Currently there are several DTN implementations available. DTN2 by the Delay Tolerant Networking Research Group (DTNRG) is mainly implemented to demonstrate basic functionality and therefore does not operate very efficiently. Another Linux implementation is ION. DTNLite [6] is designed for sensor networks running tinyOS, but does not allow for using common applications and programming languages. DASM is an implementation for Symbian smartphones [3]. In contrast to existing realizations, the idea behind IBR-DTN is to develop a powerful and efficient implementation that runs on embedded devices as well as on standard Linux systems. Except for ION and IBR-DTN, none of the implementations support version 6 of the DTN Bundle protocol. However, ION lacks important features like e.g. discovery.

In this paper, we present first results of our ongoing work of implementing IBR-DTN, a delay tolerant networking package for networked consumer devices. The evaluation shows promising performance results which are compared to the DTN2 reference implementation.

2. SYSTEM ARCHITECTURE

Our aim is to develop a light-weight implementation of delay tolerant networking which is interoperable with the DTN2 reference implementation [1], but is lean enough to run on common WLAN APs. These APs are to be used as compact "plug-and-play" DTN-Modules for vehicular and stationary applications, e.g. in (public) transportation and

CHANTS'08, September 15, 2008, San Francisco, California, USA.
ACM 978-1-60558-186-6/08/09.

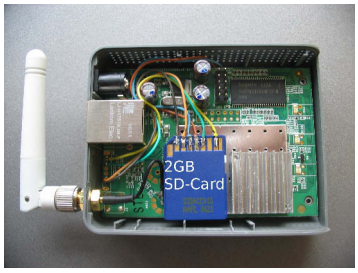

Figure 1: FON2200 with additional Storage

environmental monitoring. This way, it is possible to avoid relatively large and expensive computers in many scenarios.

The main resource constraint on mid-range APs is RAM. Usually only 5-6 MB of a total of 16 MB remain available for additional processes. Since IBR-DTN shall co-exist with the usual AP features (routing, firewalling, 802.11 encryption, etc.), it must not consume more than 4 MB RAM. For this reason, DTN2 - which uses various libraries and occupies roughly 40 MB of RAM - is inappropriate for our purposes. Further the computing power of an AP (typically with a 200 MHz MIPS CPU) is much lower compared to a regular computer. Therefore the efficiency of data manipulation algorithms requires more attention for an embedded implementation. A solution for both constraints is to use self-implemented, efficient code and data structures instead of using libraries whenever reasonable. An exception is uClibc(++) [8], a lean library optimized for embedded systems. uClibc and uClibc++ are the only libraries used by our implementation.

IBR-DTN is currently developed on a Mikrotik Routerboard 532, which features a serial console for debugging. However IBR-DTN was successfully tested on various low-cost APs e.g. Netgear WGT634U, Linksys WRT54G3G and even on an ultra low budget FON FON2200. APs with USB host adapters are most suitable, since external mass storage like USB-flashdrives offer large capacity for bundle storage. But even APs without USB host can be modified to feature additional flash memory, e.g. by attaching an SD-Card to unused GPIO-pins of the CPU. Figure 1 shows a FON2200 with an 2 GB persistent storage modification. This system was successfully tested with IBR-DTN and can be run on battery.

As shown in figure 2, IBR-DTN comprises three main modules derived from the main functions defined in RFC5050 [7], which provide basic services to the DTN core module:

Bundle Router: returns a schedule for a given bundle. Currently static routing is implemented, however the Bundle Router is designed to allow for modular routing plug-ins.

Bundle Storage: management, storage and retrieval of bundles. Additionally, it generates notifications whenever it deletes an expired bundle so that a status report can be sent if required.

Convergence Layer: provides an adapter for transport mechanisms. Currently an UDP convergence layer is implemented. For the EMMA project we also implemented a special convergence layer for environmental monitoring, which features a broadcast discovery mechanism for wireless networks. Different convergence layers are attached to and managed by the MultiplexConvergenceLayer.

The **DTN core module** implements the DTN protocol

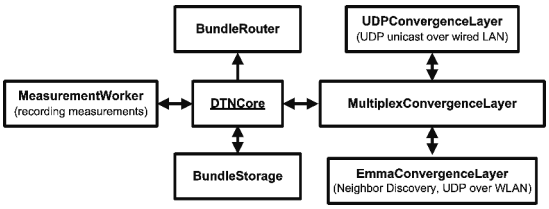

Figure 2: IBR-DTN Software Architecture

	DTN2	IBR-DTN
The Basic Bundle Protocol	X	X
Self Describing Numeric Values	X	X
Endpoint Identifiers (see below)	X	X
Bundle expirations (based on timers)	X	X
Registrations (with expiration timers)	X	-
Persistent storage of bundles	X	-
Bundle fragmentation (reactive)	X	-
Bundle fragment reassembly	X	X
Custody Transfer	X	X
Bundle Status Reports:		
Received	X	X
Forwarded	X	X
Delivered	X	X
Deleted	X	X
Acknowledged by Application	-	-
Custody Acceptance	X	X
Extension blocks	X	X
Bundle Security blocks	X	-
Proactive Fragmentation	-	X

Table 1: RFC5050 Features: DTN2 vs. IBR-DTN

logic and provides an interface to applications or "endpoint modules". An exemplary application collecting environmental data is the **Measurement Worker**.

3. EVALUATION

IBR-DTN is evaluated quantitatively and qualitatively to analyze performance and efficiency. Table 1 compares the features and table 2 compares the consumption of memory and storage resources of DTN2 version 2.5 and IBR-DTN. It is obvious that the implementation of IBR-DTN is more efficient. Not only the slim application itself, but also the lower consumption of memory makes IBR-DTN suitable to run on embedded devices with limited resources. In the next step, we compared the performance of the DTN reference implementation and IBR-DTN in a testbed. The intention was to analyze the throughput of both implementations that may be reached in a best-case scenario. For eliminating possible interferences on the wireless channel, we setup two identical notebooks with a direct Ethernet connection. The

	DTN2	IBR-DTN
Size of the daemon application	21.9 MB	114 kB
Size of the attached libraries	7.1 MB	605 kB
RAM usage (swappable VZS)	41.7 MB	3.4 MB
RAM usage (non-swappable RSS)	4.7 MB	1.7 MB

Table 2: RAM usage: DTN2 vs. IBR-DTN

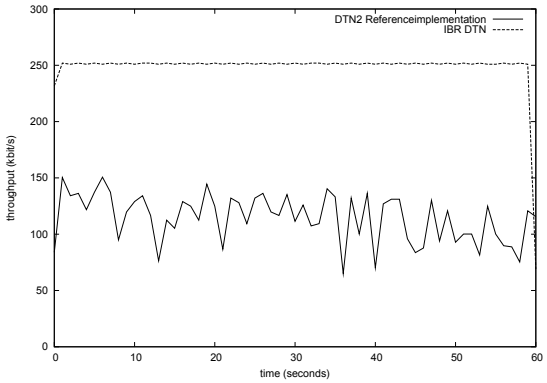

Figure 3: DTN2 and IBR-DTN Throughput

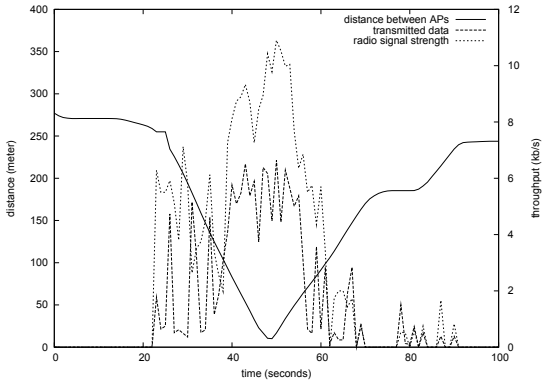

Figure 4: Real World Results

results of this measurement run are depicted in figure 3. It is obvious that the throughput of IBR-DTN is nearly twice of DTN2's throughput. During a run of 60 s, IBR-DTN reached an average throughput of 252 kbit/s, while DTN2 was only able to deliver bundles at a speed of 117 kbit/s.

Besides, we also did real-world measurements to find out how our implementation performs in the wild. For this, we run IBR-DTN on a Mikrotik Routerboard RB 532 with 802.11a/b/g Wireless Mini-PCI Cards (R52H, Atheros AR-5414 chipset). The WLAN card was configured to run in 802.11g mode. The urban traffic scenario we analyzed was a vehicle driving by an building with an indoor WLAN access point at an average speed of approximately 20 km/h. The vehicular node as well as the access point were running IBR-DTN on top of the OpenWrt operating system. The results of these measurement runs are shown in figure 4. It can be seen that the implementation works as intended and is well suited for vehicular DTNs.

4. DEMONSTRATION SETUP

We demonstrate IBR-DTN on embedded devices and show that it is compatible to DTN2. The demonstration setup comprises four mobile DTN nodes: one laptop running DTN2 and one laptop running IBR-DTN. The other devices are embedded WLAN routers running IBR-DTN on OpenWrt. The communication between the mobile devices is based on 802.11b/g WLAN. Another laptop is used to emulate the mobility of the mobile devices via a wired LAN.

In the demonstration scenario mobile DTN nodes collect sensor data while virtually moving in a terrain. Measured data sets are color values on the ground of the terrain annotated with the respective positions. This data is spread to all other participants in the DTN via an UDP convergence layer. For each node, the color values being learned from own measurements or incoming DTN bundles are displayed. The nodes' movement is emulated by switching the wireless interfaces on and off remotely. A graphical user interface gives the opportunity to choose between different movement scenarios, the size of the data sets and the interval of measurements. In addition, the user interface visualizes the bundle exchange process and the content of each bundle cache. Conference participants are able to interact with the demo in order to change the application characteristics and the time nodes are connected in different scenarios.

5. SUMMARY

In this paper we present IBR-DTN, our implementation of the DTN bundle protocol. One of its main advantages is the support of various hardware platforms from smartphones up to PCs or Notebooks. This was reached by using OpenWrt as a basis for the implementation. Moreover, IBR-DTN operates very efficiently and consumes few memory resources. This is an important fact when running the implementation on access points or mobile phones with strong resource constraints. Further advantages of IBR-DTN are the support of the latest bundle protocol version and interoperability with DTN2.

The work on IBR-DTN is still ongoing. At the moment we are porting IBR-DTN to the OpenMoko platform, to support open source smartphones like the NEO FreeRunner featuring GSM, Bluetooth and WLAN [4]. Moreover, we plan to realize a large real-world DTN testbed.

6. REFERENCES

[1] Delay Tolerant Networking Research Group. DTN Reference Implementation, August 2006. http://www.dtnrg.org/docs/code/.

[2] S. Lahde, M. Doering, W.-B. Pöttner, G. Lammert, and L. Wolf. A practical analysis of communication characteristics for mobile and distributed pollution measurements on the road: Research articles. *Wireless Communications and Mobile Computing*, 7(10):1209–1218, 2007.

[3] O. Mukhtar and J. Ott. Backup and bypass: introducing dtn-based ad-hoc networking to mobile phones. In *REALMAN '06: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 107–109, New York, NY, USA, 2006. ACM.

[4] Openmoko Community. The Openmoko Project Homepage, 2008. http://www.openmoko.org/.

[5] OpenWRT Community. The OpenWRT Project Homepage, 2008. http://www.openwrt.org/.

[6] R. Patra and S. Nedevschi. Dtnlite: A reliable data transfer architecture for sensor networks. Technical report cs294–1 course project report, Berkeley, 2003.

[7] K. Scott. Bundle Protocol Specification. RFC 5050, Nov. 2007.

[8] uClibc Community. The uClibc Project Homepage, 2008. http://www.uclibc.org/.