Technische Universität Braunschweig

Institut für Betriebssysteme und Rechnerverbund

IBR-DTN

A lightweight, modular and highly portable Bundle Protocol implementation

Sebastian Schildt, Johannes Morgenroth, Bastian Pöttner, Lars Wolf

WASA-NGI 2011

Outline

DTN and Bundle Protocol

- Delay- and Disruption tolerant networks are networks which do not provide continuous end-to-end connectivity
  - Store-and-Forward principles are applied under these conditions
- Bundle Protocol (RFC 5050 and friends) is a protocol specification for realising DTN networks
  - It is implemented - among others - by DTN2, which acts as a reference implementation for Bundle Protocol

DTN Applications

DTN@IBR

- Equipping trams with DTN nodes
- Cooperation with BBR Verkehrstechnik
- Update passenger information displays

Other applications

Interplanetary Communication, Distributed Sensing, VANET's and providing connectivity to sparsely populated rural areas

Lightweight and Portable

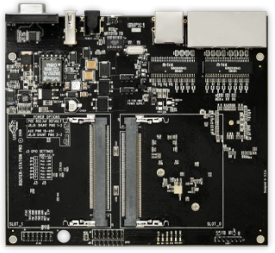
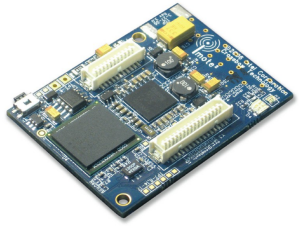
Targeted platforms

- Embedded Linux plattforms (OpenWRT as main target)
  - (We are POSIX compliant, so porting to others OS's should be feasible)
- Scalable to "Big Iron", no compromises compared to DTN2 running on similar hardware

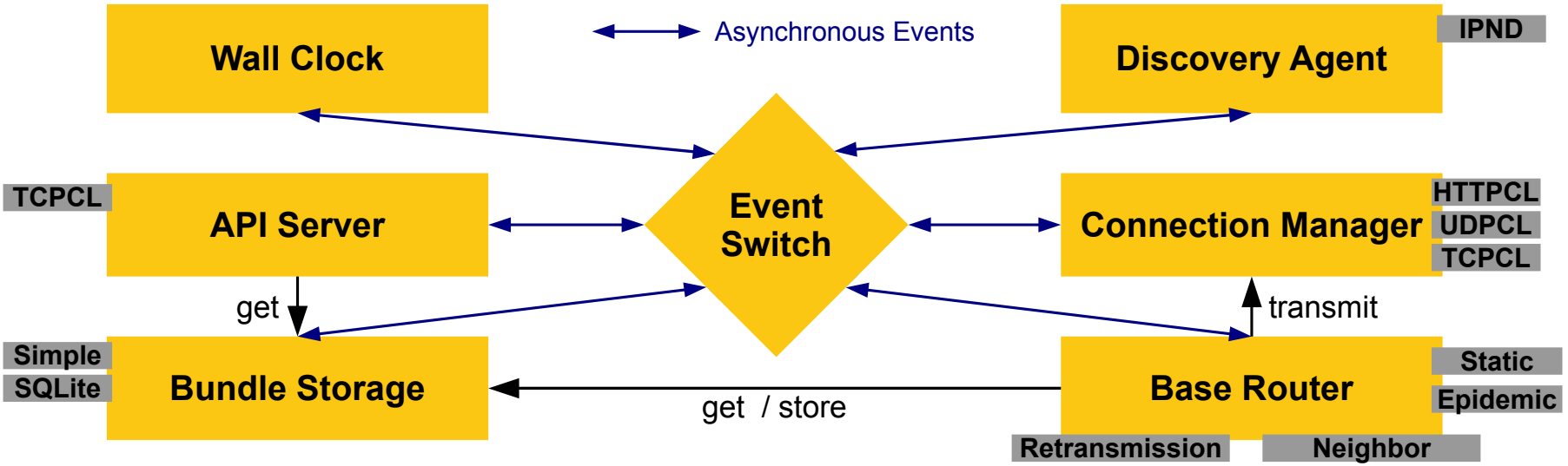
Not Targeted

- No sensor nodes with 8 bit μCs
  - (Although we are doing some work for the convergence of BP based DTNs with classical sensor node platforms)

Supported Platforms

CPU Architectures	Software Stack	Hardware Platforms
x86 x86-64 MIPS TECHNOLOGIES Power™ ARM®	OpenWrt Wireless Freedom uClibc Linux 2.6 Linux 2.4	 Ubiquiti RouterStation Pro  Memsic (CrossBow) iMote2  QNAP TS219-P NAS  Buffalo TeraStation Pro NAS

Architecture Overview



- Modular architecture
- Written in C++

Interacting with the DTN Daemon

C++ API

- Link IBR-DTN C++ library to your program
- Communicates with the DTN daemon using a TCP or Unix socket
- Full flexibility, but can be complex

Commandline Tools

- Diagnostics: `dtnping`, `dtntracepath`
- Simple data transfer: `dtnsend`, `dtnrecv`
- DTN enabled scripting: `dtntrigger` ("Poor man's DTN API")

Storage Backends

Memory

- Non persistent, RAM based

File

- Persistent, file based
- Bundles survive scheduled daemon restarts as well as power failures

SQLite

- Stores more meta information for bundles
- Useful for more complex routing modules

Routing modules

Static

- Routes and available neighbours are configured statically

Neighbor

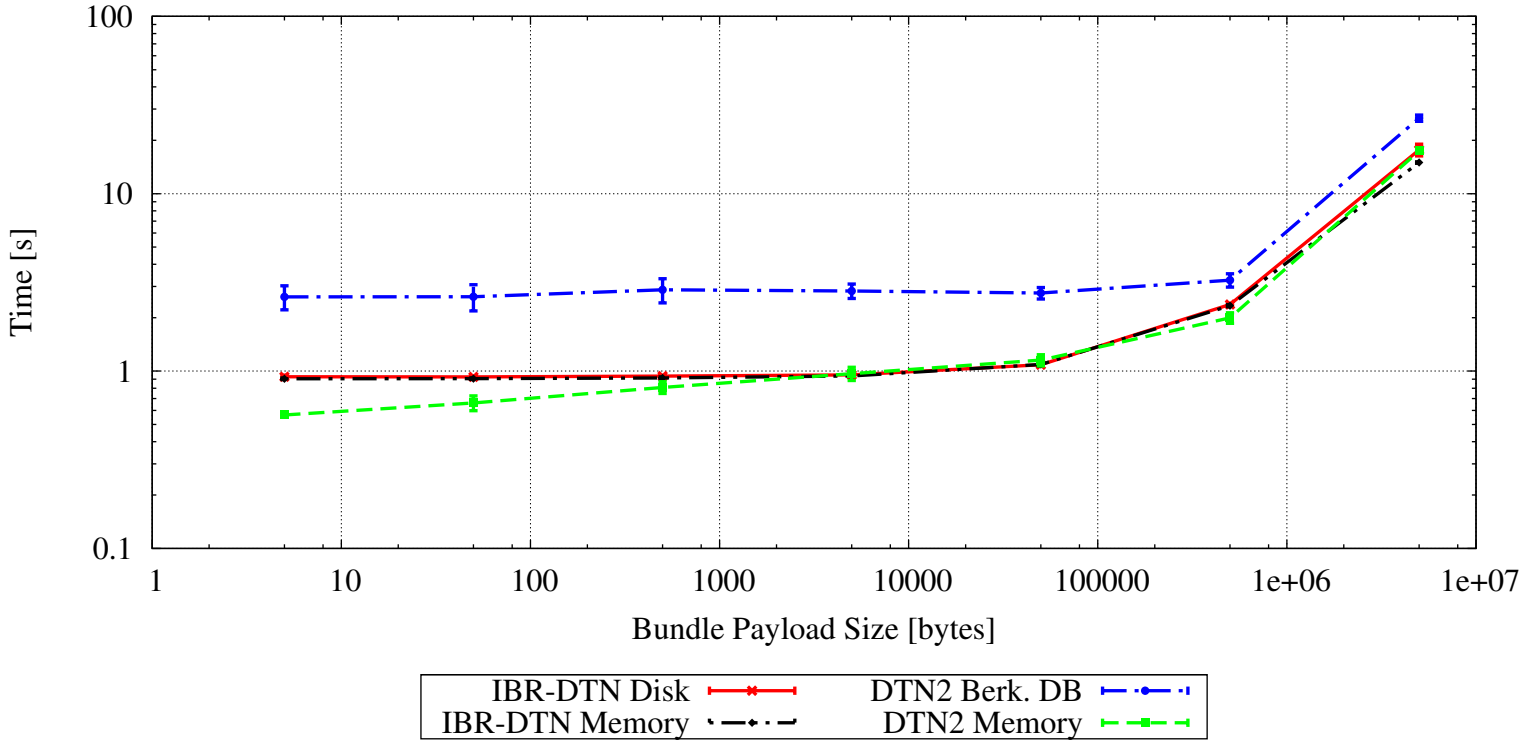
- Route packets to nodes seen by a Discovery Agent
- When using IPND or DTN2 announcements, nodes within the same subnet are reachable

Epidemic

- Epidemic routing (a flooding scheme) implementation
- Uses Bloom filters for summary vectors and is extended with purge vectors

API Send Performance

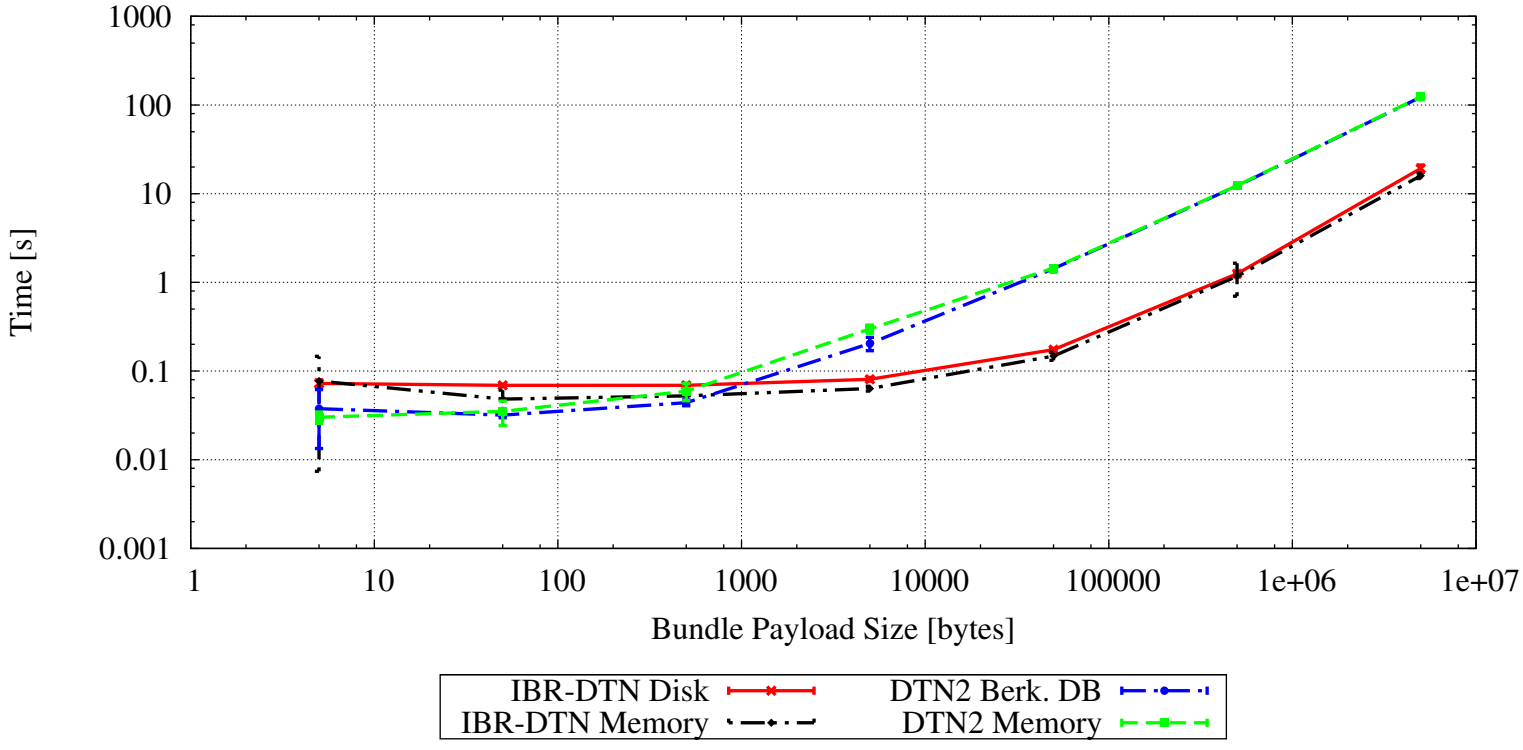
Send 100 bundles using `dtnsend` and store in respective storage



- $\sim$ 265 MBit peak throughput

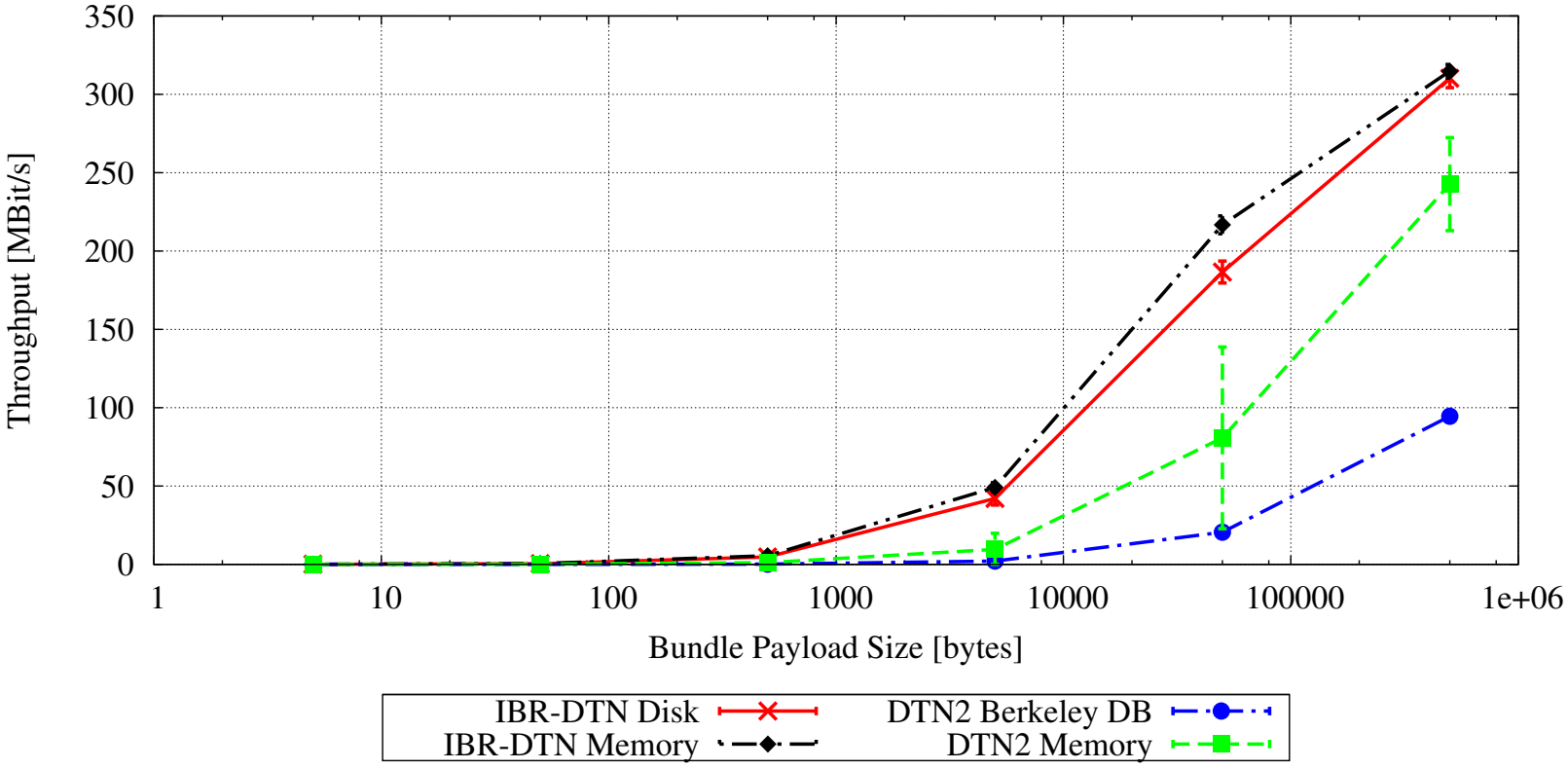
API Receive Performance

Receive 100 bundles using `dtnrecv` from respective storage



- $\sim$ 250 MBit peak throughput

Transfer Test



- 10 runs of 1000 bundles for each size
- $\sim$ 310 MBit peak throughput

Conclusions

- IBR-DTN provides a full featured Bundle Protocol stack, which is interoperable with DTN2
- Specifically targeted for uClibc based embedded Linux platforms but scalable to non-embedded environments
- Comparable or better raw performance compared to DTN2
- Modularized, lightweight C++ codebase
- Available from our website to be used in your projects now!

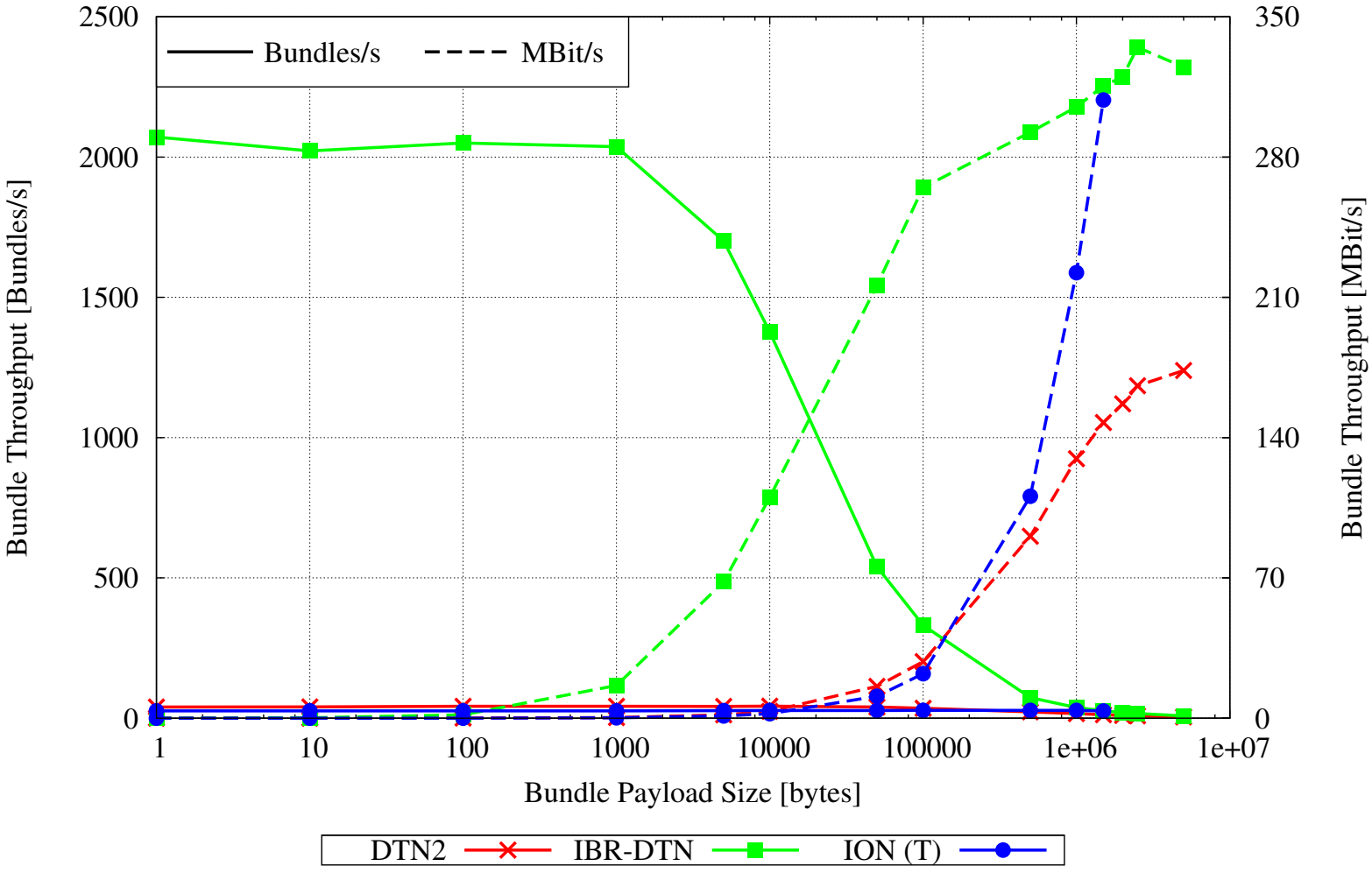
Thank you! Questions?

Sebastian Schildt
schildt@ibr.cs.tu-bs.de

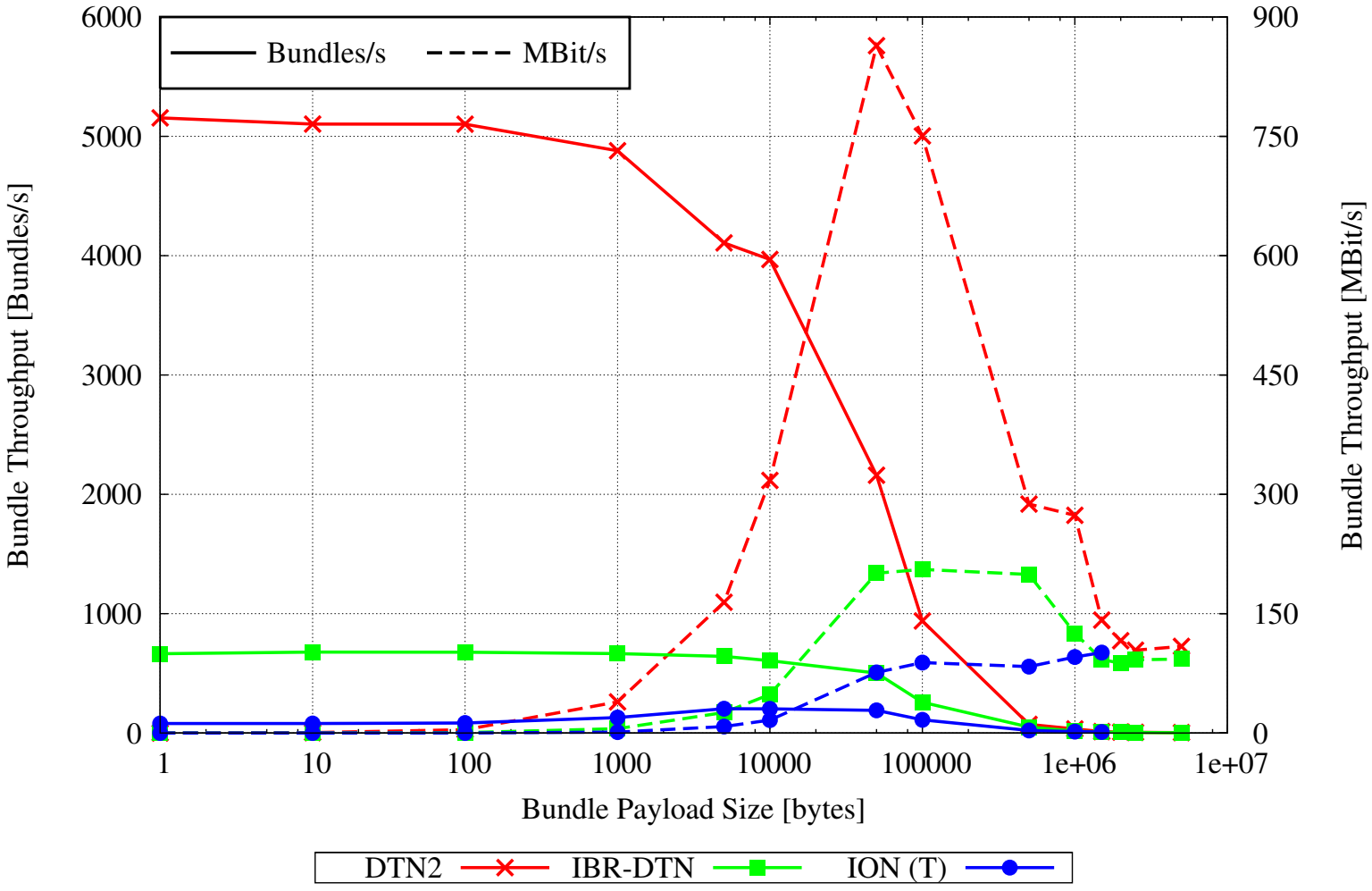
http://www.ibr.cs.tu-bs.de/projects/ibr-dtn/

New version 0.6 released yesterday evening!

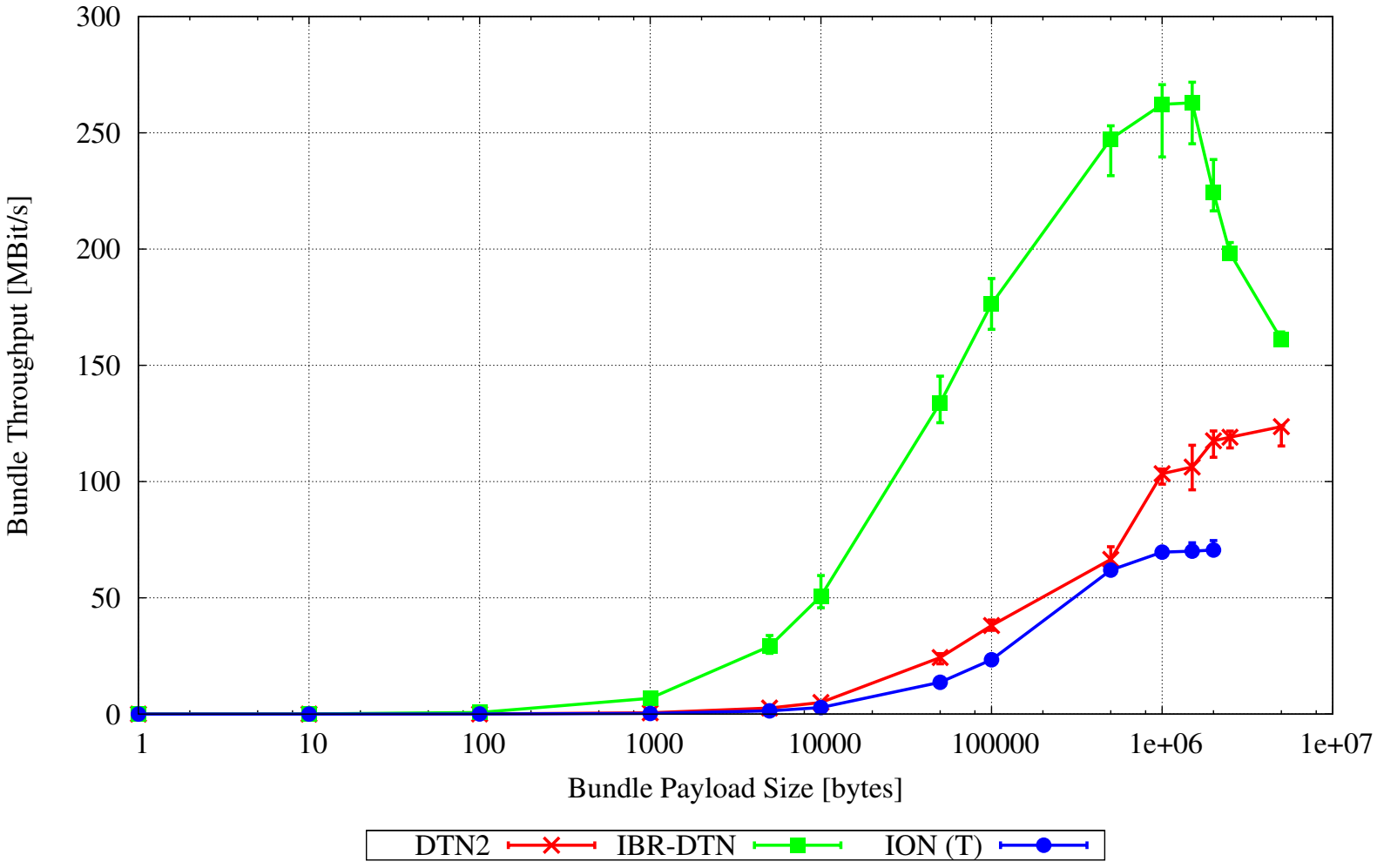
API send (store)



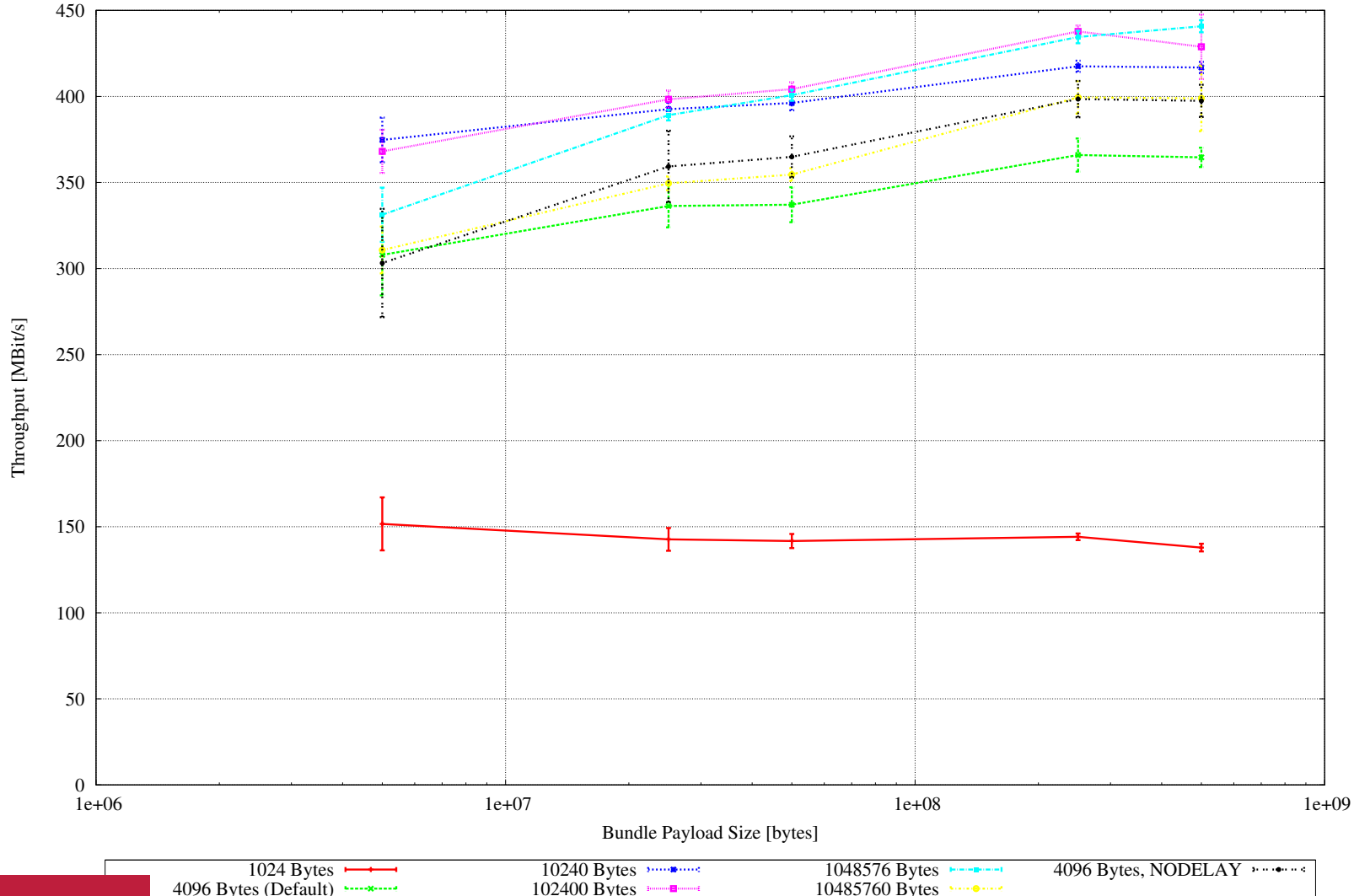
API receive (retrieve)



Throughput


Bundle Throughput [MBit/s]
300
250
200
150
100
50
0
1
10
100
1000
10000
100000
1e+06
1e+07
Bundle Payload Size [bytes]
DTN2
IBR-DTN
ION (T)

Chunksize


Throughput [MBit/s]
Bundle Payload Size [bytes]
0
50
100
150
200
250
300
350
400
450
1e+06
1e+07
1e+08
1e+09
1024 Bytes
4096 Bytes (Default)
10240 Bytes
102400 Bytes
1048576 Bytes
10485760 Bytes
4096 Bytes, NODELAY

Caching / Sync