Performance Evaluation and Analysis of Delay Tolerant Networking

Earl Oliver, Hossein Falaki

David R. Cheriton School of Computer Science
University of Waterloo

11th June 2007

Motivation

Delay Tolerant Networking

Many projects (KioskNet, DieselNet and SenNDT) use **DTN** over *opportunistic* wireless connections.

Problem Definition

1. What are the hardware requirements for DTN applications?
2. What is the optimal configuration in terms of performance?
3. What methodology should be used to study DTN performance?

Motivation

Delay Tolerant Networking

Many projects (KioskNet, DieselNet and SenNDT) use **DTN** over *opportunistic* wireless connections.

Problem Definition

1. What are the hardware requirements for DTN applications?
2. What is the optimal configuration in terms of performance?
3. What methodology should be used to study DTN performance?

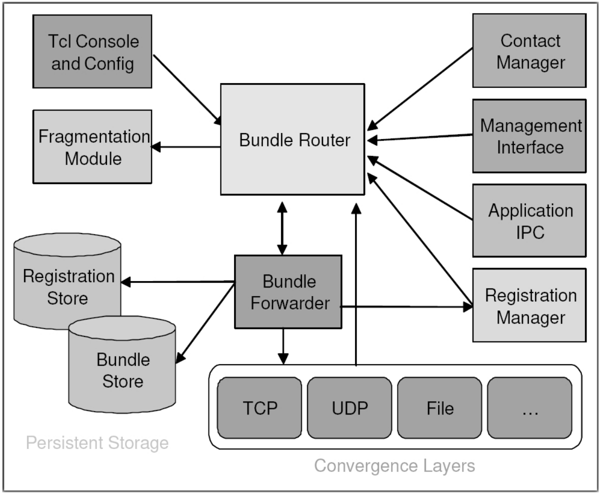
Outline

Outline

Outline

Outline

DTN Reference Implementation (DRI)



Methodology

Steps

1. We chose common DTN hardware
2. Determined its capacity with *microbenchmarks*
3. Hypothesized about **DRI performance**
4. Evaluated the hypotheses

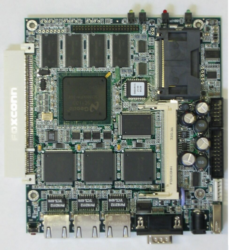
DRI Performance

Throughput between DRI nodes during opportunistic connections.

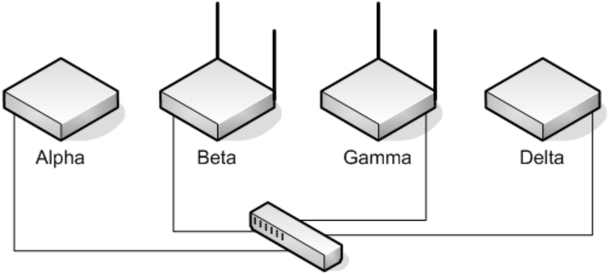
Test Bed

H/S Configuration

- **Hardware**: Soekris net4801: 266 MHz processor and 256 MB SDRAM
- **WiFi card**: Atheros 802.11abg wireless cards
- **OS**: Stable Debian with Linux Kernel 2.6.8-3
- **DRI**: DTNRG CVS head as of February 22, 2007.



Test Bed Topology



- Wired network as control plane
- Wireless connection between *Beta* and *Gamma*

Storage I/O Microbenchmarks

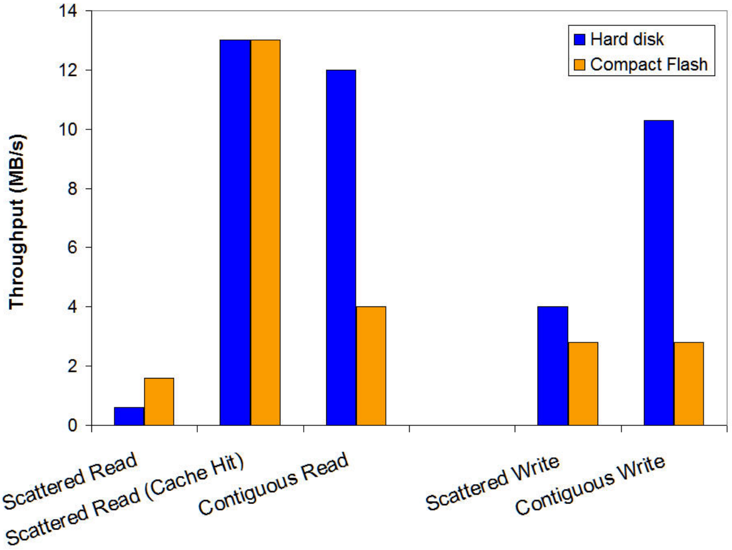
Motivation

Measuring the actual storage throughput limitations

Storage I/O Throughput

- Scattered I/O: reading/writing from/to 1000 to 4000 files
- Contiguous I/O: reading/writing a single large file

I/O Microbenchmarks



Network Microbenchmarks

Motivation

Measuring the actual wireless network throughput limitations

Network I/O Throughput

- Single TCP connection without any disk involvement (3.1 MB/s)
- Reading and writing from/to disk on TCP ends (1.9 MB/s)

Hypotheses: Resource Limitations

- **CPU**: We do not expect to be the primary bottleneck, although it is a scarce resource.

- **Disk**: We expect to be the primary bottleneck of the DRI

- **Memory**: Page faults do not limit the DRI performance

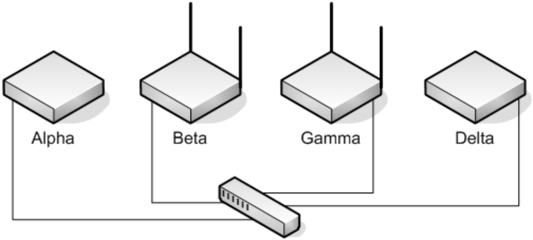
Hypotheses: System and Application Parameters

- **Wireless data rate**: The DRI performance will peak at a rate of at most 24 Mb/s
- **Bundle size**: Increasing bundle size will improve performance
- **Parallelism**: Performance could be improved by parallelism

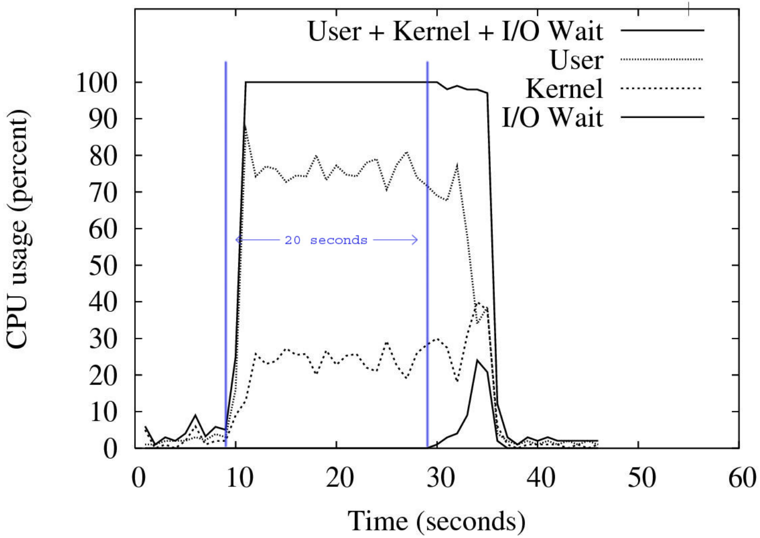
Experiments

Method

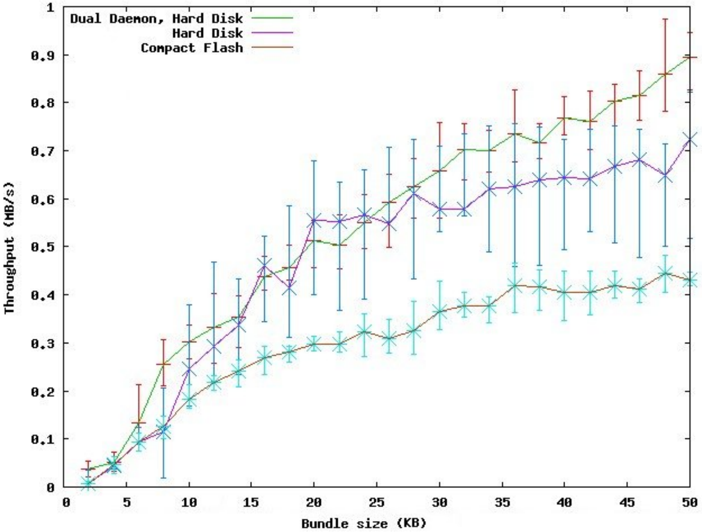
- Enqueue a fixed amount of data at *beta*
- Simulate a 20 second opportunistic connection
- Measure data remaining in queue after connection is closed



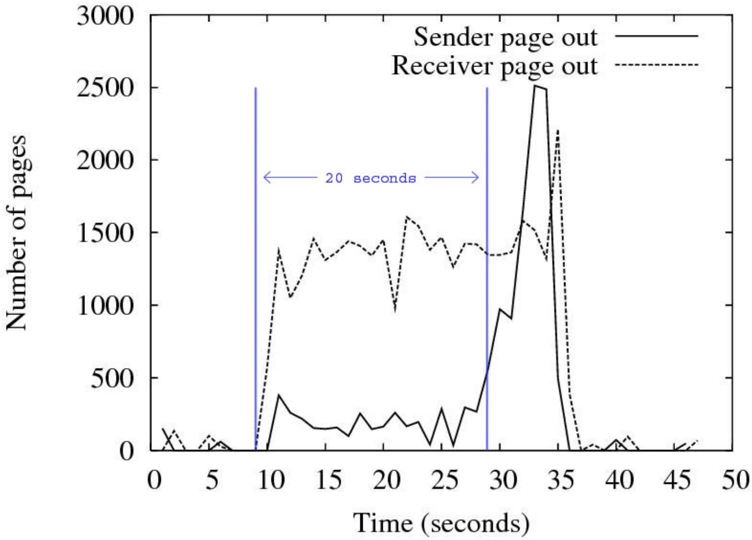
CPU activity during a wireless opportunistic connection



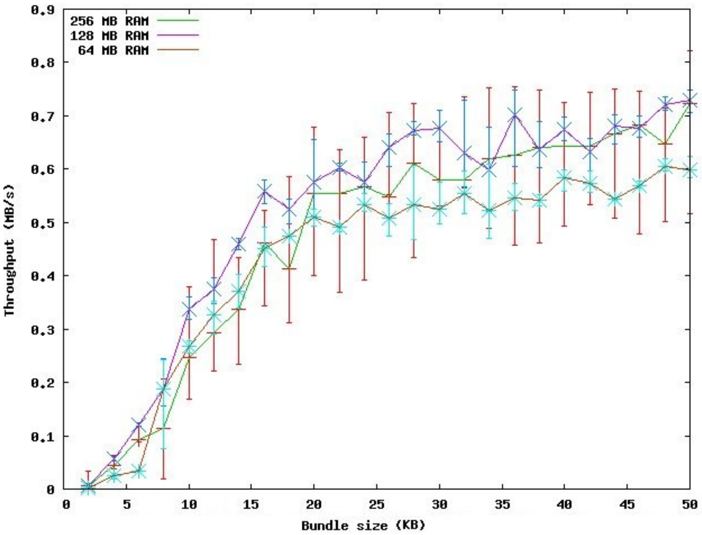
Data throughput vs. bundle size



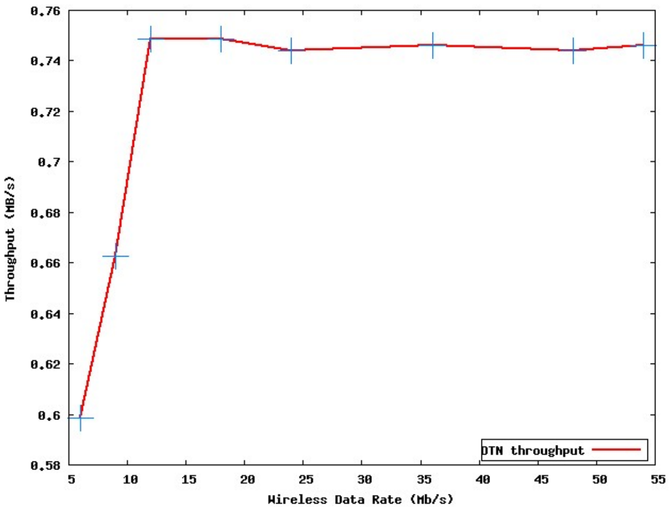
Disk block I/O during a wireless opportunistic connection



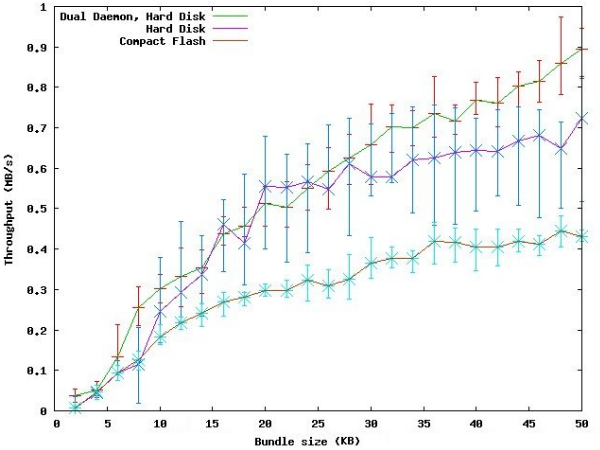
Data throughput vs. bundle size using a portion of memory



Data throughput vs. wireless data rate



Parallelism



- 16% higher throughput (despite being CPU bound)

Why?

- 8.28% less time spent on spinlocks
- For example, in "servlib/bundling/BundleList.cc" line 31:

```
BundleList::BundleList(const std::string& name)
    : Logger("BundleList", "/dtn/bundle/list/%s", name.c_str()),
      name_(name), lock_(new oasys::SpinLock()), notifier_(NULL)
{
}
```

Summary

Conclusion

- The primary bottleneck to the DRI performance on common hardware is **CPU**
- Bundle size highly affects DRI performance
- Methodology for evaluating the performance of other mobile systems using opportunistic connections

Recommendations

- **Application developers**: use the largest possible bundle size for the DRI in-memory API
- **DRI developers**: restructure DRI to increase parallelism or remove spinlocks
- **DRI users**: invest more in the CPU

Summary

Conclusion

- The primary bottleneck to the DRI performance on common hardware is **CPU**
- Bundle size highly affects DRI performance
- Methodology for evaluating the performance of other mobile systems using opportunistic connections

Recommendations

- **Application developers**: use the largest possible bundle size for the DRI in-memory API
- **DRI developers**: restructure DRI to increase parallelism or remove spinlocks
- **DRI users**: invest more in the CPU