Analysis of DCCP for Delay/Disruption Tolerant Networking

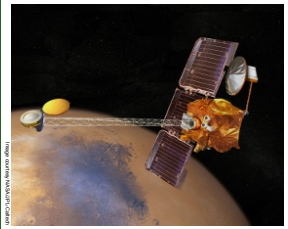
Samuel Jero
Internetworking Research Group, Ohio University

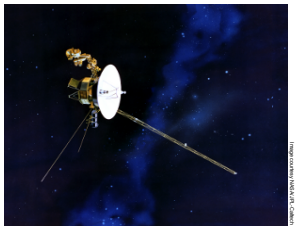
Delay/Disruption Tolerant Networks

Characteristics:

1. **Long round trip times across the network**
2. **Frequent interruptions in connectivity**
3. **High error rates**

Hence, normal TCP/IP protocols operate poorly





Ohio University, in partnership with NASA, is working to develop solutions to the problem of networking in deep space.

Deep space networking has all the characteristics of a Delay/Disruption Tolerant Network (DTN):

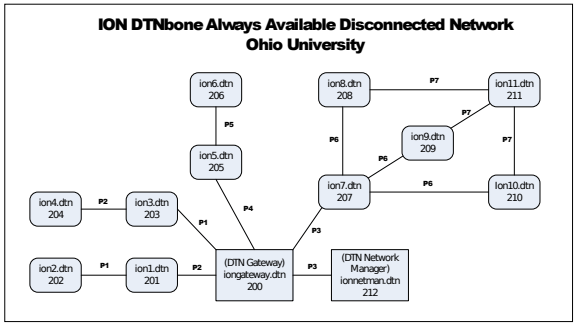
1. **Long Round trip times** -- from hours to seconds
2. **High error rates** -- on the order of 10^{-1} to 10^{-3}
3. **Interruptions in connectivity** -- as planets pass between endpoints

The Bundle Protocol [RFC 5050] and the Licklider Transmission Protocol (LTP) [RFC 5326] have been specifically designed to operate well in this environment.

The DTNBone

We are maintaining and testing an implementation of the Bundle Protocol and the Licklider Transmission Protocol known as the Interplanetary Overlay Network (ION).

For testing purposes, a group of organizations has put together a worldwide collection of ION/DTN2 nodes known as the DTNBone.



The Problem

We want to test DTN protocols across the Internet in order to get an understanding of how these protocols will operate in complex environments of many nodes and unexpected changes. However, in order to do that, we need to determine how to encapsulate DTN protocols to be able to connect DTNBone nodes across the terrestrial Internet.

We came up with three options:

1. **TCP** -- This is not a good choice because it is reliable and we need to test LTP, which provides reliable delivery.
2. **UDP** -- This is what was used for a long time. However, without congestion control, LTP will send faster than the connection can sustain causing congestion collapse.
3. **DCCP** -- Provides congestion control without reliability. This makes it particularly well suited for our application.

Datagram Congestion Control Protocol

The Datagram Congestion Control Protocol (DCCP) is a transport layer protocol designed to provide congestion control for applications that care more about receiving data regularly than about retransmitting lost data.

This protocol is ideal for VOIP, IPTV, and on-line games where congestion control is necessary, but reliable delivery could cause long delays or pauses in the content stream. DCCP is also perfect for our LTP tests because it makes no guarantee of reliable delivery; it only provides congestion control.

Key Features of DCCP:

- **Congestion control**
- **No guarantee of reliable or in-order delivery**
- **Connection-oriented protocol**
- **Multiple congestion control algorithms to choose from**
- **Explicit Congestion Notification capable**
- **Acknowledgments can be congestion controlled**
- **Highly extensible**
- **Implemented in the Linux kernel since 2005**

Congestion control algorithms currently implemented:

1. **"TCP-like congestion control".** This algorithm is extremely similar to SACK-based TCP's congestion control and is recommended for those applications that would like as much bandwidth as possible. This is the algorithm we chose to use to connect ION nodes.
2. **"TCP-friendly rate control".** This algorithm is designed for those applications that would like to minimize sudden changes in sending rate, for instance, VOIP applications. This algorithm is not window based. Instead it is based on a rate of sending packets which is reduced in response to congestion.

Analysis of DCCP for Delay/Disruption Tolerant Networking

OHIO UNIVERSITY

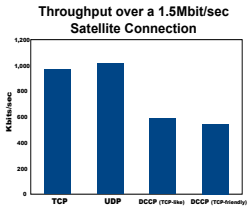
Samuel Jero
Internetworking Research Group, Ohio University

2/2

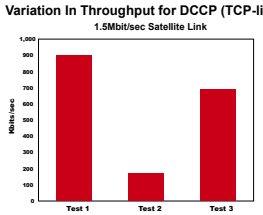
OHIO UNIVERSITY

DCCP and ION

Once we added DCCP encapsulation to ION's LTP implementation, we noticed some very interesting behaviors using the current DCCP development kernel (pulled 10/3/2010 -- based on Linux 2.6.36-rc3). These behaviors are best explained by the following graphs:



DCCP's throughput compares poorly with TCP's or UDP's



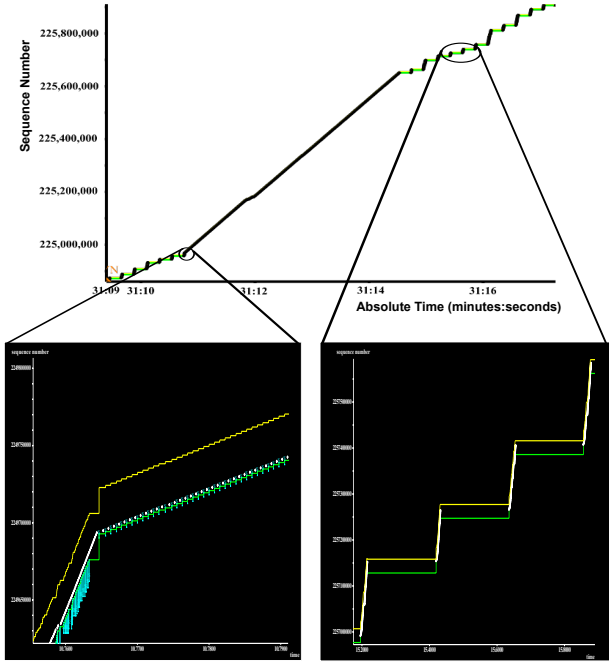
There are huge variations between connections on the *same* link

DCCP Analysis Methodology

1. Setup a bridged machine between two DCCP nodes
2. Capture packets on the middle machine while transferring data using DCCP between the other two machines
3. Use a program that we wrote to transform these DCCP packet captures into equivalent TCP packet captures (this only works because the DCCP congestion control is nearly identical to TCP's)
4. Run this TCP capture through Tcptrace to generate connection graphs for analysis

DCCP Analysis

Time Sequence Graph of a Typical Linux DCCP Connection



DCCP reduces congestion window to 1 packet per round trip time

DCCP randomly pauses for 200ms multiple times during the connection

We also determined that the Linux implementation of DCCP responds inconsistently to dropped packets. We have observed any of the following as reactions to dropped packets:

1. Reducing congestion window
2. Doing nothing
3. Increasing congestion window

Conclusions

DCCP was designed to provide congestion control for applications that care more about receiving data regularly than about retransmitting lost data. It seems well poised to provide that, but the only maintained implementation exhibits behavior that severely limits it's usefulness.

The Linux implementation of DCCP needs improvement before it is ready for practical deployment on the Internet.

The issues discovered here need to be overcome before we can practically use this protocol even for testing purposes.

These issues are:

1. Poor performance compared to TCP and UDP
2. Tendency to operate with congestion window of one packet
3. Random 200ms pauses in connections
4. Inconsistent response to lost packets

Future Work

1. Analyze Linux source code to find any bugs that might be causing these problems. Then develop and test fixes and inform the authors.
2. Any remaining issues should be taken up with the DCCP IETF working group.
3. Once these problems are fixed we can convert the DTNBone over to DCCP and continue DTN testing.

Acknowledgments

We thank Shawn Ostermann for guidance and advice in this research and Gilbert Clark for aiding in innumerable small ways.

For Further Information

Please contact Samuel Jero at sj323707@ohio.edu. More information on DTN Research at Ohio University is available at http://irg.cs.ohiou.edu