Routing Testing Service

Data Center TCP (DCTCP)
Test Plan

Technical Document

Version 1.0

University of New Hampshire
InterOperability Laboratory **https://www.iol.unh.edu**

Table of Contents

Modification Record

Version	Date	Editor	Modification
1.0	2020-06-01	Timothy Carlin	• Initial Document

Acknowledgements

The University of New Hampshire would like to acknowledge the efforts of the following individuals in the development of this test suite:

Timothy Carlin (UNH-IOL)
Kyle Ouellette (UNH-IOL)
Matthew Hartman
Timothy Winters

Introduction

The University of New Hampshire InterOperability Laboratory (UNH-IOL) tests networking and data communications products. Since 1988, the laboratory has fostered multi-vendor interoperability while preparing students for careers in the industry. The laboratory has grown steadily into one of the industry's premier independent proving grounds for new technologies.

Scope:

As described in RFC 8257, “DCTCP extends the Explicit Congestion Notification (ECN) processing to estimate the fraction of bytes that encounter congestion rather than simply detecting that some congestion has occurred. DCTCP then scales the TCP congestion window based on this estimate. This method achieves high-burst tolerance, low latency, and high throughput with shallow- buffered switches.”

This Test Plan describes a set of tests to exercise either the Sender or Receiver role of the DCTCP. These tests are designed to be run using a packet generator/protocol analyzer against an isolated DUT (Device, or implementation, Under Test).

Definitions

DCTCP	Data Center TCP as described by RFC 8257
DUT	Device, or implementation, Under Test
Sender	ECN Capable Implementation (see RFC 8257 Section 3)
Receiver	ECN Capable Implementation (see RFC 8257 Section 3)
Switch	Intermediate Device that signals to receiver when congestion detected

Possible Problems

RFC 8257 does not specify a precise algorithm for use in marking packets with CE, rather recommends an algorithm in section 3.1.

Test Organization

This document organizes tests by group based on related test methodology or goals. Each group begins with a brief set of comments pertaining to all tests within that group. This is followed by a series of description blocks; each block describes a single test. The format of the description block is as follows:

Test Label	The **Test Label** is the first line of the test page. Scripts implementing this test suite should follow this convention, and may also append a character in the set [a-z] indicating a particular test part.
Purpose	The **Purpose** is a short statement describing what the test attempts to achieve. It is usually phrased as a simple assertion of the feature or capability to be tested.
Test Setup	The **Test Setup** section describes the configuration of all devices prior to the start of the test. Different parts of the procedure may involve configuration steps that deviate from what is given in the test setup. If a value is not provided for a protocol parameter, then the protocol's default is used for that parameter.
Procedure and Expected Behavior	The **Procedure and Expected Behavior** table contains the step-by-step instructions for carrying out the test. These steps include such things as enabling interfaces, unplugging devices from the network, or sending packets from a test station. The test procedure also cues the tester to make observations of expected behavior, as needed, as not all steps require observation of results. If any behavior is expected for a procedure, it is to be observed prior to continuing to the next step. Failure to observe any behavior prior to continuing constitutes a failed test. Note, that while test numbers continue between test parts, each test part is to be executed independently (Following Common Test Setup and Cleanup as indicated), and are not cascaded from the previous part.
Possible Problems	The **Possible Problems** section contains a description of known issues with the test procedure, which may affect test results in certain situations.

References

Label	Citation
[DCTCP]	Data Center TCP (DCTCP): TCP Congestion Control for Data Centers. S. Bensley, D. Thaler, P. Balasubramanian, L. Eggert, G. Judd. October 2017. (Format: TXT, HTML) (Status: INFORMATIONAL) (DOI: 10.17487/RFC8257)

Common Test Setup

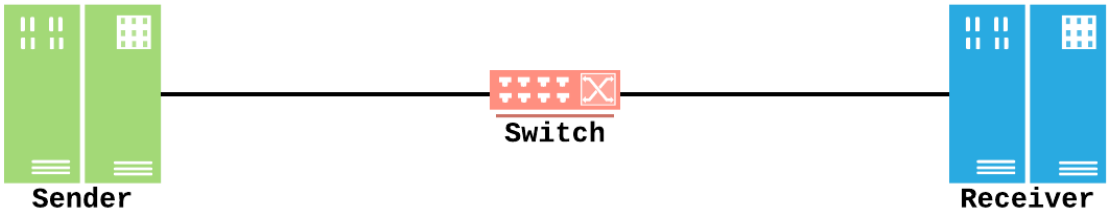
Sender Target

1. Configure the Sender to enable DCTCP.
2. Establish a TCP Connection between Sender and Receiver

Receiver Target

1. Configure the Receiver to enable DCTCP.
2. Establish a TCP Connection between Sender and Receiver

Common Test Topology



Section 1: DCTCP Sender and Receiver

Overview:

This section focuses on the DCTCP algorithm for Sender and Receiver implementations.

DCTCP.Conf.1.1: ECT Codepoint Set During Handshake

Purpose: Sender sets the ECT codepoint for SYN packets

Reference:
- [DCTCP] – Section 3.6

Target:
- Sender

Test Setup: No Common Test Setup is performed.

Procedure:

Part A: ECT Codepoint on SYN

Step	Action	Expected Behavior
1.	Initiate a DCTCP connection from the Sender.	The Sender transmits a TCP ECN-Setup SYN with the ECT codepoint set.

DCTCP.Conf.1.2: ECT Codepoint Set During Handshake

Purpose: Receiver sets the ECT codepoint for SYN-ACK and RST packets

Reference:
- [DCTCP] – Section 3.6

Target:
- Receiver

Test Setup: No Common Test Setup is performed.

Procedure:

Part A: ECT Codepoint on SYN-ACK

Step	Action	Expected Behavior
1.	Begin listening for DCTCP connections on the Receiver.	The Receiver is actively listening for DCTCP connections.
2.	Initiate a DCTCP connection from the Sender by sending a TCP ECN-Setup SYN.	The Receiver transmits a TCP ECN-Setup SYN-ACK with the ECT codepoint set.

Part B: ECT Codepoint on RST

Step	Action	Expected Behavior
3.	Initiate a DCTCP connection from the Sender by sending a TCP ECN-Setup SYN. Ensure that the Receiver is not listening for DCTCP connections.	The Receiver transmits a TCP RST-ACK with the ECT codepoint set.

DCTCP.Conf.1.3: Echoing Congestion Information

Purpose: Receiver detects the state of the codepoint and reacts accordingly.

Reference:
- [DCTCP] – Section 3.2

Target:
- Receiver

Test Setup: Perform Common Test Setup

Procedure:

Part A: CE codepoint is not set and DCTCP.CE is false

Step	Action	Expected Behavior
1.	The Sender transmits a TCP data packet. Switch *does not* set the CE codepoint.	The Receiver acknowledges all data with ECE=0.
2.	The Sender transmits a TCP data packet. Switch *does not* set the CE codepoint.	The Receiver acknowledges the data. DCTCP.CE remains set to false. Any ACKs have ECE=0.

Part B: CE codepoint is set and DCTCP.CE is false

Step	Action	Expected Behavior
3.	The Sender transmits a TCP data packet. Switch *does not* set the CE codepoint.	The Receiver acknowledges all data with ECE=0.
4.	The Sender transmits a TCP data packet. Switch sets the CE codepoint.	The Receiver sets DCTCP.CE to true and sends an immediate ACK with ECE=1.

Part C: CE codepoint is not set and DCTCP.CE is true

Step	Action	Expected Behavior
5.	The Sender transmits a TCP data packet. Switch *does not* set the CE codepoint.	The Receiver acknowledges all data with ECE=0.
6.	The Sender transmits a TCP data packet. Switch sets the CE codepoint.	The Receiver sets DCTCP.CE to true and sends an immediate ACK with ECE=1.
7.	The Sender transmits a TCP data packet. Switch *does not* set the CE codepoint.	The Receiver sets DCTCP.CE to false and sends an immediate ACK with ECE=0.

Part D: CE codepoint is set and DCTCP.CE is true

Step	Action	Expected Behavior
8.	The Sender transmits a TCP data packet. Switch *does not* set the CE codepoint.	The Receiver acknowledges all data with ECE=0.
9.	The Sender transmits a TCP data packet. Switch sets the CE codepoint.	The Receiver sets DCTCP.CE to true and sends an immediate ACK with ECE=1.
10.	The Sender transmits a TCP data packet. Switch sets the CE codepoint.	The Receiver sets DCTCP.CE to true and sends an immediate ACK with ECE=1.

Possible Problems:

- An implementation MAY choose to send two ACKs: one for previously unacknowledged packets and another acknowledging the most recently received packet.

DCTCP.Conf.1.4: Processing Echoed Congestion Indications

Purpose: Sender processes the indication of congestion by the receiver and reacts accordingly.

Reference:
- [DCTCP] – Sections 3.3, 3.4

Target:
- Sender

Test Setup: Perform Common Test Setup

Procedure:

Part A: Reduce TCP Congestion Window Size

Step	Action	Expected Behavior
1.	Transmit data from the Sender.	The Sender transmits a TCP data packet.
2.	The Receiver acknowledges all data with ECE=0.	
3.	Take note of the TCP congestion window (cwnd) size.	
4.	Transmit data from the Sender.	The Sender transmits a TCP data packet.
5.	The Receiver transmits an immediate ACK with ECE=1. This indicates that the Switch encountered congestion and set the CE codepoint which causes the Receiver to set DCTCP.CE to true.	The Sender computes a congestion estimate and reacts by reducing the TCP congestion window (cwnd) size.
6.	Take note of the TCP congestion window (cwnd) size.	The newly computed congestion window (cwnd) size should be smaller compared to that of Step 3.

Part B: Increase TCP Congestion Window Size

Step	Action	Expected Behavior
7.	Transmit data from the Sender.	The Sender transmits a TCP data packet.
8.	The Receiver transmits an immediate ACK with ECE=1.	The sender computes a congestion estimate and reacts by reducing

	This indicates that the Switch encountered congestion and set the CE codepoint which causes the Receiver to set DCTCP.CE to true.	the TCP congestion window (cwnd) size.
9.	Take note of the TCP congestion window (cwnd) size.	
10.	Transmit data from the Sender.	The Sender transmits a TCP data packet.
11.	The Receiver transmits an immediate ACK with ECE=0. This indicates that the Switch is no longer experiencing congestion and unset the CE codepoint which causes the Receiver to set DCTCP.CE to false.	The sender computes a congestion estimate and reacts by increasing the TCP congestion window (cwnd) size.
12.	Take note of the TCP congestion window (cwnd) size.	The newly computed congestion window (cwnd) size should be larger compared to that of Step 9.

Possible Problems:

- It may take several TCP Data Packets to see a difference in the TCP Window Size.

DCTCP.Conf.1.5: Processing Loss Indications

Purpose: Sender reacts to loss episodes by utilizing fast retransmit and fast recovery algorithms.

Reference:
- [DCTCP] – Section 3.5

Target:
- Sender

Test Setup: Perform Common Test Setup.

Procedure:

Part A: Reduce Congestion Window Size on Loss Event

Step	Action	Expected Behavior
1.	Send data from the Sender	The Sender transmits a TCP Data Packet.
2.	The Receiver acknowledges all data with ECE=0.	
3.	Take note of the TCP congestion window (cwnd) size.	
4.	Send data from the Sender	The Sender transmits a TCP data packet.
5.	The Receiver acknowledges only some of the data, indicating loss occurred.	The sender reacts to the loss event and computes a congestion estimate and reacts by reducing the TCP congestion window (cwnd) size.
6.	Take note of the TCP congestion window (cwnd) size.	The newly computed congestion window (cwnd) size should be smaller compared to that of Step 3.

Part B: Increase Congestion Window Size when Loss has Cleared

Step	Action	Expected Behavior
7.	Send data from the Sender.	The Sender transmits a TCP data packet.
8.	The Receiver acknowledges only some of the data, indicating loss occurred.	The sender reacts to the loss event and computes a congestion estimate and reacts by reducing

		the TCP congestion window (cwnd) size.
9.	Take note of the TCP congestion window (cwnd) size.	
10.	Send data from the Sender	The Sender transmits a TCP Data Packet.
11.	The Receiver acknowledges all data with ECE=0.	The sender detects the conclusion of the loss event and computes a congestion estimate and reacts by increasing the TCP congestion window (cwnd) size.
12.	Take note of the TCP congestion window (cwnd) size.	The newly computed congestion window (cwnd) size should be larger compared to that of Step 11.

Possible Problems:

- It may take several TCP Data Packets to see a difference in the TCP Window Size.