European Commission — FP7

RITE

REDUCING INTERNET TRANSPORT LATENCY

Project acronym: **RITE** Project number: 317700 Work package: Use-case trials Deliverable number and name: Deliverable 3.4: RITE recommended use parameters report	**Title:** RITE recommended use parameters report **Work Package:** WP3 **Version:** 1 **Date:** November 16, 2015 **Pages:** 23
	Author: Per Hurtig
	Co-Author(s): Gorry Fairhurst, Bob Briscoe, Koen De Schepper, Andreas Petlund, Naeem Khademi, Nicolas Kuhn, Iain Learmonth **To:** Jorge Carvalho Project Officer

Status:		**Confidentiality:**		
[ ]	Draft	[X]	PU	— Public
[ ]	To be reviewed	[ ]	PP	— Restricted to other programme participants
[ ]	Proposal	[ ]	RE	— Restricted to a group
[X]	Final / Released to CEC	[ ]	CO	— Confidential

Revision:
(Dates, Reviewers, Comments)

Contents:
This report provides guidelines on how to set end-host and network parameters to minimise the latency experienced by applications.

Contents

Abbreviations

This section provides definitions of key terms and defines the abbreviations used in the remainder of the report.

3G 3rd Generation

3GPP 3rd Generation Partnership Project

AccECN Accurate ECN

ACK Acknowledgement

ADU Application Data Unit

API Application Programming Interface

AQM Active Queue Management

ARED Adaptive Random Early Drop

BE Best Effort

BCP Best Current Practice

BDP Bandwidth Delay Product

BoF Birds-of-a-Feather

CAIDA Cooperative Association for Internet Data Analysis

CC Congestion Control

CDF Cumulative Density Function

CDN Content Delivery Network

CE Congestion Experienced

CM Congestion Manager

CMT-SCTP Concurrent Multipath Transfer for SCTP

CoDel Controlled Delay

CUBIC Cubic function congestion control

cwnd Congestion WiNDow

CWR Congestion Window Reduced

CWV Congestion Window Validation

DASH Dynamic Adaptive Streaming over HTTP

DC Data Center

DCLC Data Center Latency Control (IRTF research group)

DCTCP Data Center TCP

DiffServ Differentiated Services

DNS Domain Name System

DOCSIS Data Over Cable Service Interface Specification

DoS Denial of Service

DRR Deficit Round Robin

DSCP Differentiated Services Code Point

DSL Digital Subscriber Line

E-UTRAN Evolved Universal Terrestrial Radio Access Network

E2E End-to-End

ECE ECN-Echo

ECN Explicit Congestion Notification

ECT ECN-Capable Transport

eNB eNodeB, Evolved Node B (LTE Base-Station)

FIFO First-In First-Out

FQ Fair queuing

FQ-CoDel Flow queuing CoDel

FSE Flow State Exchange

FTP File Transfer Protocol

GB Gigabyte

GPRS General Packet Radio Service

GRE Generic Routing Encapsulation

GTP GPRS Tunnelling Protocol

GUTS Getting up to Speed

HAS HTTP Adaptive Streaming

HSS Hybrid Slow-Start

HTTP HyperText Transfer Protocol

HULL High-bandwidth Ultra Low Latency

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

IP Internet Protocol

IRTF Internet Research Task Force

ISP Internet Service Provider

IW Initial Window

LAN Local Area Network

LEDBAT Low Extra Delay BAckground Transport

LKM Loadable Kernel Module

LTE Long Term Evolution

MAC Media Access Control

MB Megabyte

MPLS Multi Protocol Label Switching

MPTCP Multipath TCP

MSS Maximum Segment Size

NAT Network Address Translation

NS-2 Network Simulator 2

NV-GRE Network Virtualization using Generic Routing Encapsulation

NVO Network Virtualization Overlay

NVP Non Validated Period

OWD One Way Delay

PCI Packet Congestion Indication

PCN Pre-Congestion Notification

PDI Packet Drop Indication

PDPC Packet Discard Prevention Counter

PDV Packet Delay Variation

PGW Packet Gateway

PIE Proportional Integral controller Enhanced

PLT Page Load Time

PSP PipeACK Sampling Period

QV Queue View

RCP Rate Control Protocol

RED Random Early Drop

RFC Request For Comments

RITE Reducing Internet Transport Latency End-to-End

RMCAT RTP Media Congestion Avoidance Techniques (IETF Working Group)

RT Real-Time

RTCWEB Real Time Collaboration on world wide WEB (IETF Working Group)

RTO Retransmission Time Out

RTP Real-Time Protocol

RTT Round Trip Time

RW Restart Window

SACK Selective Acknowledgement

SAE System Architecture Evolution

SBD Shared Bottleneck Detection

SCTP Stream Control Transmission Protocol

SDO standards Development Organisation

SFQ Stochastic Fair Queuing

SGW Serving Gateway

SKB Socket Buffer

SQ Source Quench

SVD Singular Value Decomposition

SYN Synchronize packet

TFRC TCP Friendly Rate Control

TCP Transmission Control Protocol

TCP′ Paced and RTT Independent TCP (TCP-PRIme)

TCPM TCP Maintenance (IETF Working Group)

TOS Type Of Service

TRILL TRansparent Interconnection of Lots of Links

TSQ TCP Small Queue

UDP User Datagram Protocol

URG TCP Urgent flag

VCP Variable-structure congestion Control Protocol

VXLAN Virtual Extensible LAN

WiFi Wireless Fidelity

WG Working Group

WP Work Package

WRED Weighted Random Early Drop

Participant organisation name	**Participant Short Name**	**Country**
Simula Research Laboratory	SRL	Norway
BT	BT	UK
Alcatel-Lucent	ALU	Belgium
University of Oslo	UiO	Norway
Karlstad University	KaU	Sweden
Institut Mines-Télécom	IMT	France
University of Aberdeen	UoA	UK
Megapop	MEGA	Norway

1 Introduction

RITE has investigated methods of removing the root causes of unnecessary latency over the Internet. Whilst time-of-flight delay is inevitable, greater delays can result from interactions between transport protocols and buffers. It is these concerns that RITE has tackled.

This document summarises the recommended use of parameters for low latency Internet communications, both in end systems and in network equipment. The guidelines are offered to businesses and network operators for general equipment parameters' use, together with advice for application designers.

It identifies mechanisms and configuration parameters that can benefit the next generation low-latency applications. It describes a recommended set of parameters based on analyses performed by the RITE project and the results of experiments using mechanisms to reduce Internet end-to-end transport latency. This advice includes consideration of how configuration recommendations interact and the need for any joint parameter tuning. The document also offers advice on how to configure parameters depending upon application traffic type. For more information on how different traffic types are affected by configuration parameters, please see [1] and Deliverable 3.1.

RITE took a three-pronged approach to reducing latency — it included both end-system and network-based approaches:

- End system – for example, we believe that protocol interactions cause extra delay and that protocols can be optimised to improve latency with no, or limited, impact on throughput.
- Network – for example, we believe that large buffers throughout the network add to the end-to-end delay, and that they can be reduced or even eliminated.
- The interaction between end-system and network elements — for instance, where end systems can make latency-reduction decisions by utilising network hints.

On the basis of this work, RITE identified a number of areas within the network and end-systems where improvements can be made regarding current best practices. The policy recommendations that form the core of this deliverable are targeted at key stakeholders in the Internet communication ecosystem, namely network operators, equipment vendors and software vendors.

The recommendations will assist each of the stakeholders in furthering the goals of reducing Internet latency by providing recommendations for the enabling and tuning of technologies in the network and at end-hosts. These functions are suggestions to address and attend the central issues that are identified in RITE through the research work.

This report comprises:

- Design considerations for end-hosts
- Recommended configuration parameters for end-hosts
- Design considerations for networks
- Recommended configuration parameters for networks
- Future considerations from RITE research

2 End-host

This section describes recommended design considerations and parameter configuration of end-hosts, to reduce communication latency over the Internet. The design considerations and parameter configurations are the results from analyses and experimentations done within the RITE project.

2.1 Design considerations

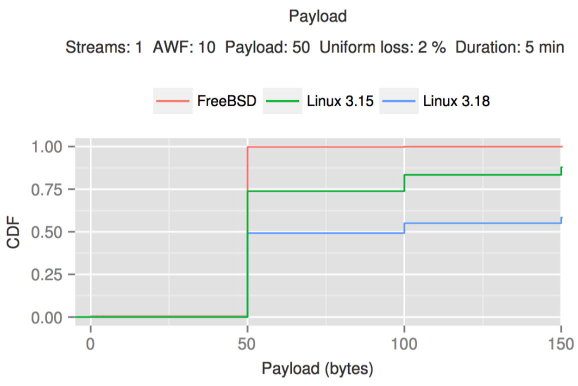
RITE has investigated the impact that common design choices have on the latency of end-host communication. In the coming section we summarise the impact of how the end-host accounts for transmitted and received data.

2.1.1 Packet-based vs byte-based accounting

When measuring network delay it is common to use packet capture tools like libpcap. Such tools capture the data as they leave the network stack and, consequently, do not include the waiting time from when the application pushes data to the kernel until it is transmitted. There are many circumstances that can contribute to the waiting time in the kernel (or the *sojourn time*) being much higher than what is necessary. This section discusses the sojourn time in the Linux kernel for thin streams and the ambiguities that arise for the TCP stack when having to do a hybrid between packet-based and byte-based accounting for data segments in the kernel. For more information and a wider discussion, see the technical report [2].

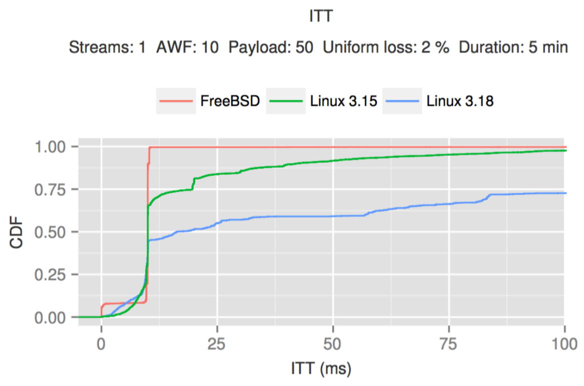
There are two dominant strategies for implementing accounting in the network stack of operating systems: 1) Maximum Segment Size (MSS) based accounting and byte-based accounting. Linux, for example, uses MSS-based accounting while FreeBSD uses byte-based accounting. For greedy traffic, MSS-based accounting works well, but for thin streams having small packets with high inter-transmission times (ITTs), MSS-based accounting opens up for a range of corner-cases and misunderstandings. This is because a large number of small application segments may be sent to the network stack without accumulating enough bytes to cross the 1 MSS limit, thus being registered in the network stack as a flow that never tries to grow the congestion window.

To compare performance between existing solutions, we have measured the sojourn time and total latency of flows that try to send more than one packet per RTT, but that will never generate enough data to cross the 1 MSS limit in the send buffer. Figure 2.1 shows statistics of packets captured with libpcap as



Name	Host	Streams	ITT (avg)	Payload (avg)	CWND
FreeBSD	fsender	1	9 / 9 / 9	50/50/50	3086/1448/2896/2898/2898/7242
Linux 3.15	rdbsender	1	18 / 18 / 18	96/96/96	10/1/7/9/13/20
Linux 3.18	rdbsender	1	51 / 51 / 51	264/264/264	3/1/3/3/3/32

(a)



Name	Sent p	Lost p	Retr p	RDB p	Payl B	Unique B	Rdn B	Retr B	Loss % (B/p)	DupACKs
FreeBSD	30.5 K	563	559	11	1.5 M	1.5 M	53.6 K	45.4 K	1.95 / 1.87	8527
Linux 3.15	15.8 K	323	322	0	1.5 M	1.5 M	32.9 K	32.9 K	2.17 / 2.05	2799
Linux 3.18	5.8 K	130	128	0	1.5 M	1.5 M	36.8 K	36.8 K	2.44 / 2.26	473

(b)

Figure 2.1: Experiment with an application producing a segment of 50 Bytes every 10ms. Packets are captured as the segments leave the network stack.

the data leaves the network stack. The plot shows ITT and payload size statistics. If the packets were sent immediately, all the packets should have a payload of 50 Bytes and an ITT of 10ms. The CDF, however, shows that many segments are held back in the send buffer and combined before being sent, thus adding to the total delay. When we inspect the sojourn time in figure 2.2, we can see that there is a

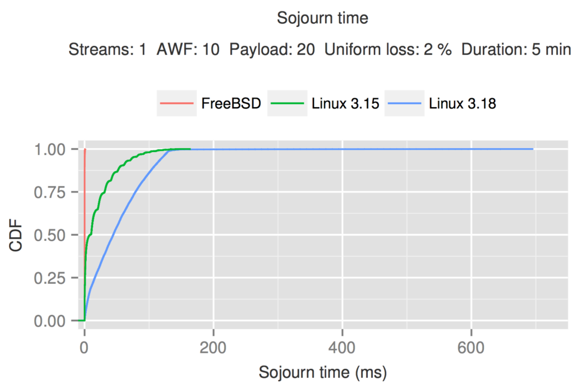


Figure 2.2: Send buffer sojourn time for the experiment with an application producing a segment of 50 Bytes every 10ms. Packets are captured as the segments leave the network stack.

big difference between recent Linux versions, as well as a noticable difference between FreeBSD and the Linux stack. FreeBSD allows for immediate delivery of the small segments, while Linux tends to hold them back to a larger degree. The reason why Linux behaves worse than FreeBSD for these scenarios is the interaction between mechanisms doing MSS-based accounting and mechanisms that to some degree try to do byte-based accounting.

Working on a hybrid between MSS-based and byte-based accounting makes it harder to handle cases where the segment sizes are lower than 1 MSS, like we often can see in interactive traffic like sensor networks, games, remote control and signal traffic. The delay added by holding segments back in the send buffer is critical to the applications involved. Using byte-based accounting allows for a flexible toolkit for dynamically adjusting the network stack's handling of thin-streams. We therefore recommend choosing byte-based accounting when designing transport mechanisms for thin streams. When performing measurements, it is also useful to instrument measurements of the send-buffer sojourn time, as it is not obvious that segments are dispatched as soon as they are delivered from the application.

2.2 Configuration parameters

There are various mechanisms available in common operating systems that can be configured to reduce transport latency. This section uses Linux to exemplify the recommendations for end-host configuration. Linux is used as example because of its prevalence in the sever market and its relative ease of configuration. However, not all mechanisms are available in the Linux mainline kernel. For mechanisms that are not available, we point to available patches or describe the mechanism in general as well as parameter tuning recommendations as a guideline to potential implementers.

2.2.1 Configuration of RITE mechanisms

While all mechanisms developed by RITE intend to reduce latency, they focus on different types of traffic. Below we describe how to optimise short flows (RTO and TLP Restart), thin flows (Redundant Data Bundling), bursty flows (New Congestion Window Validation) and general traffic (CAIA Delay Gradient and TCP Alternative Backoff with ECN) for low latency.

RTO and TLP Restart Both TCP and SCTP use a retransmission timeout (RTO) timer [3, 4] as a last resort for data loss recovery. The standardised way in which this timer is restarted often unnecessarily extends the loss recovery time by at least one round-trip time (RTT). The extended loss recovery time is problematic as applications transmitting short/bursty time-sensitive traffic are often forced to use RTO-based loss recovery due to limited traffic feedback. The RTO Restart (RTOR) mechanism [5] can be used to remove this unnecessary delay. Further, Linux makes use of a non-standardised retransmission scheme called Tail Loss Probe (TLP) [6] that is designed to tackle the slow loss recovery in the end of flows. Like RTO, TLP also makes use of a timer that has the same restart problem. To mitigate this performance problem, RITE has developed a restart strategy for TLP, which we call TLP Restart (TLPR), that is similar to RTOR. Evaluations of the RTOR and TLPR [7] mechanisms show that they reduce loss recovery latency and complement each other. Therefore RITE recommends to enable both mechanisms on systems where they are available. The Linux mainline kernel does not incorporate these mechanisms, but these are available for download from [8].

```
Assuming the kernel is built with the patch referenced above:
Linux commands for enabling RTOR and TLPR:
sudo sysctl net.ipv4.tcp_rto_restart=1
sudo sysctl net.ipv4.tcp_tlp_restart=1
Note: This is the default setting when building the kernel with the patch.
```

Redundant Data Bundling (RDB) Flows that are time-dependent and/or interactive often transmit only a small amount of data with relatively large inter-packet interval times as compared to greedy traffic. Examples of such applications are sensor networks, game servers and remote control. Redundant data Bundling (RDB) is a sender-side only mechanism that tries to preempt the experience of packet loss by redundantly bundling all unacknowledged data as long as the resulting segment does not surpass 1 Maximum Segment Size (MSS). This allows for effectively avoiding high retransmission delays without sending any more packets into the network. Below, we have listed the recommended settings for the most important tunable parameters of the mechanism based on the findings described in Opstad et al. [9]. The Linux mainline kernel does not incorporate this mechanism, but it can be downloaded from [10].

```
Assuming the kernel is built with the patch referenced above:
Load the kernel module: "modprobe tcp_rdb"
To use RDB on a TCP connection, set the socket option TCP_THIN_RDB when setting up the socket.
Enable dynamic PIF limit with a min. inter-transmission time of 10ms.
sysctl -w net.ipv4.rdb.tcp_thin_dpif_lim=10
Limit RDB to bundling one old segment with each new:
sysctl -w net.ipv4.rdb.rdb_max_bundle_skbs=1
```

New Congestion Window Validation (new-CWV) RFC 7661 [11] specifies a mechanism to address issues that arise when TCP is used for traffic that exhibit periods where the sending rate is limited by the application rather than the congestion window. It provides an experimental update to TCP that allows a TCP sender to restart quickly following a rate-limited interval. This method is expected to benefit applications that send rate-limited traffic using TCP, while also providing an appropriate response if congestion is experienced. This document obsoletes a previous mechanism described in [11]. While this mechanism is not available in mainline Linux, it is available at [12].

```
Assuming the kernel is built with the patch referenced above:
Linux command for enabling new-CWV:
sudo sysctl net.ipv4.tcp_newcwv=1
```

CAIA Delay Gradient (CDG) CAIA Delay Gradient is a delay-based congestion control that takes its backoff decisions based on estimates of whether the queue is growing or shrinking based on the gradient of delay measurements within RTT periods. The mechanism includes a heuristic that switches the congestion control to ”New Reno” when it detects that it is competing with other loss-based congestion control mechanisms. The aim of this is to get the queueing delay benefits when alone on the bottleneck or competing with delay-based congestion controls, and still be able to achieve a fair throughput when competing with loss-based congestion control. When turning off the loss competition heuristics, CDG can be deployed as a less-than-best-effort congestion control. As of version 4.2, CDG is a part of the mainline Linux kernel and can be directly applied.

```
The default configuration in the Linux kernel is the currently recommended settings.
Scavenger traffic (less-than-best-effort) should disable coexistence heuristics
using parameters use_shadow=0 and use_ineff=0.
```

2.2.2 Configuration of non-RITE mechanisms

The mechanisms listed in this section all affect low-latency communication, although some were not designed for this purpose. Similarly to the configuration of the RITE mechanisms, we list them in order of the traffic they mostly affect. TCP Fast Open is solely designed for short flows. Next, Nagle's algorithm, TLP and ER, thin-stream fast retransmissions and linear-backoff, and the initial window of TCP all affect the latency of short and thin flows. The initial window also affects the latency of general flows. ECN can reduce head-of-line blocking delay and has the potential to allow other mechanisms to further reduce latency.

TCP Fast Open (TFO) RFC 7413 [13] describes an experimental TCP mechanism called TCP Fast Open (TFO). TFO allows data to be carried in the SYN and SYN-ACK packets and consumed by the receiving end during the initial connection handshake, and saves up to one full round-trip time (RTT) compared to the standard TCP, which requires a three-way handshake (3WHS) to complete before data can be exchanged. However, TFO deviates from the standard TCP semantics, since the data in the SYN could be replayed to an application in some rare circumstances. Applications should not use TFO unless they can tolerate this issue. RFC 7413 also describes further research and issues that need to be explored before this can become recommendation for general Internet use.

Many web applications can benefit from TFO because the HTTP protocols provide the required mechanisms to protect from replay. In particular, TFO is designed to deliver data safely during connection establishment without requiring the server to store per-client authentication information. A client obtains a TFO cookie from the server when accessing the server for the first time. This cookie contains encrypted information that authenticates the client. When the client wants to open a new connection to the same server, it returns the cookie to the server by inserting it into the payload. The server recognises the IP address of the client by deciphering the cookie, and therefore it can accept the TCP payload of the SYN segment. This reduces the delay of initial connection establishment, which is non-negligible in the overall web page download delay.

```
Linux command:
For servers, the change is very simple. Just set a socket option before issuing
the 'listen' system call on the socket:
int qlen = 5; // Incoming connections to queue
setsockopt(sfd, SOL_TCP, TCP_FASTOPEN, &qlen, sizeof(qlen));


For clients, the change is also simple, but more intrusive. For example, when
opening a new connection and performing a HTTP get request, replace calls to
connect() + send()/write() with:
sendto(sfd, data, data_len, MSG_FASTOPEN, (struct sockaddr *) &server_addr, addr_len);
```

Nagle's algorithm is traditionally enabled by default for most operating systems. This is a mechanism that trades latency for utilisation when a host sends small packets, by delaying the transmission of small packets in order to bundle several packets together to a larger segment before sending. For any time-dependent application, delaying the sending of packets will reduce the quality of service. Our recommendation is therefore to disable Nagle's algorithm if the application is time-dependent.

```
Linux command:
When creating a new socket, add TCP_NODELAY as a socket option.
```

Tail Loss Probe (TLP) and Early Retransmit (ER) are the mechanisms that were made in recognition of the fact that thin-streams and the tail of flows will not have the sufficient ACK feedback to be able to trigger a fast retransmit. ER [14] will reduce the number of ACKs needed to trigger a fast retransmission if the number of outstanding segment is less than four. TLP [6] tries to trigger loss recovery by inducing additional selective ACKs at the receiver, even if the last segment is lost. TLP transmits one probe segment after what is called a ”probe timeout” if the connection has outstanding data but is otherwise idle. This segment will then (in accordance with ER) trigger a retransmission of the outstanding segment without having to wait for a retransmission by timeout.

```
Linux command for enabling both TLP and ER:
sudo sysctl tcp_early_retrans=3
Note: This is the default setting in Linux, and may not need to be changed, usually.
```

Thin-stream fast retransmission If the need for low latency is larger for thin streams, it is possible to enable a mechanism that statically retransmits after 1 duplicate ACK. This will reduce recovery times compared to ER since the number of dupACKs needed to trigger a fast retransmission will not be dynamically adjusted based on outstanding segments.

```
Linux command for enabling thin-stream fast retransmit:
sudo sysctl tcp_thin_dupack=1
```

Thin-stream linear backoff For a thin stream experiencing a large number of recoveries by timeout, losing consecutive segments will lead to exponential backoff that massively increases the wait before retransmitting. If your application is sensitive to high maximum delays, you should turn on a mechanism that postpones the exponential factor until 6 retransmissions have been tried using a standard RTO.

```
Linux command for enabling thin-stream linear backoff:
sudo sysctl tcp_thin_linear_timeouts=1
```

Increased Initial Window RFC 6928 [15] proposes an experiment to increase the permitted TCP initial window (IW) from between 2 and 4 segments as specified in RFC 3390 [16], to 10 segments with a fall-back to the existing recommendation when performance issues are detected. It discusses the motivation behind this increase, the advantages and disadvantages of the higher initial window, and presents results from several large-scale experiments that show that the higher initial window improves the overall performance of many web services without resulting in congestion collapse.

Although increasing the TCP initial window is a small change, it has considerable impact on web transfer latency. According to the authors, this may be the only immediately applicable solution to remove the incentive for web developers to use multiple concurrent connections to reduce the download latency. However, results of large scale experiments also highlight the risks of congestion when using the method with very low-speed links. Since the extent of such networks is unknown, an initial experimental phase before the method could be universally deployed is recommended by the IETF.

Explicit Congestion Notification (ECN) [17] describes the benefits when applications use a transport that enables Explicit Congestion Notification (ECN) [18]. It outlines the principal gains in terms of increased throughput, reduced delay and other benefits when ECN is used over a network path that includes equipment that supports ECN-marking. It also discusses challenges for successful deployment of ECN.

Internet transports, such as TCP and SCTP, are implemented in endpoints and designed to detect and react to network congestion. Congestion may be detected by packet loss or, if ECN is enabled, by the reception of a packet with a Congestion Experienced (CE)-marking in the IP header. ECN may also be enabled by other transports: UDP applications that provide congestion control may enable ECN when they are able to correctly process the ECN signals (e.g., ECN with RTP [19]).

An ECN-capable network device can signal incipient congestion (network queueing) at a point before a transport experiences congestion loss or high queuing delay. The marking is generally performed as the result of various AQM algorithms, where the exact combination of AQM/ECN algorithms does not need to be known by the transport endpoints. Since ECN makes it possible for the network to signal the presence of incipient congestion without incurring packet loss, it lets the network deliver some packets to an application that would otherwise have been dropped if the application or transport did not support ECN. This packet loss reduction is the most obvious benefit of ECN, but it is often relatively modest. However, enabling ECN can also result in a number of beneficial side-effects for applications, some of which may be much more significant than the immediate packet loss reduction from ECN-marking instead of dropping packets. Several benefits reduce latency (e.g., reduced head-of-line blocking).

RITE, in collaboration with mPlane, has performed studies for the support for ECN in end-hosts and in the network [20], in order to assess the marginal risk of enabling ECN by default in client end-hosts. The Linux kernel has enabled “server-mode” ECN by default for some time, and so will negotiate ECN when it is requested. Over half of tested webservers in the Alexa top 1 million list were capable of negotiating ECN and less than half a percent of hosts showed ECN dependent connectivity failure when an ECN negotiation was attempted. RITE has also performed a study of selected consumer networking equipment, detailed in appendix A.

The marginal risk to enable ECN on client end-hosts has been shown to be minimal and therefore RITE recommends that ECN can now be enabled by default on client end-hosts. There will still be a small number of edge cases where ECN negotiation attempts may prevent TCP connections from being established, and so the option to disable ECN negotiation should not be removed at this time.

3 Network

This section describes recommended design considerations and parameter configuration of network components to further low-latency communication over the Internet. The design considerations and parameter configurations are results from analysis and experimentation work done within the RITE project.

3.1 Design considerations

Traditionally, networking equipment has been designed to enable as high throughput as possible, not considering latency problems caused by excessive buffering. RITE challenges this approach and has investigated a number of alternative design considerations including AQM guidelines, the repurposing of existing AQM equipment, ECN encapsulation guidelines and squared AQM.

3.1.1 AQM guidelines (IETF)

RITE has contributed to the production of RFC 7567 [21] on IETF Recommendations Regarding Active Queue Management (BCP 197). This presents recommendations to the Internet community concerning measures to improve and preserve Internet performance. There is a strong recommendation for testing, standardization, and widespread deployment of active queue management (AQM) in network devices to improve the performance of today's Internet. It also urges a concerted effort of research, measurement, and ultimate deployment of AQM mechanisms to protect the Internet from flows that are not sufficiently responsive to congestion notification. This memo replaces the recommendations in RFC 2309. It also explicitly obsoletes the recommendation that Random Early Detection (RED) be used as the default AQM mechanism for the Internet. This is replaced by a detailed set of recommendations for selecting an appropriate AQM algorithm.

It motivates the need for AQM, describes the combination of AQM and Multiple Queues, use of AQM and Explicit Congestion Marking (ECN), AQM and Buffer Size, and research issues concerning managing aggressive flows. These mechanisms can be implemented in network devices on the path between endpoints that include routers, switches, and other network middleboxes. The methods may also be implemented in the networking stacks within endpoint devices that connect to the network.

The key recommendations are:

1. Network devices SHOULD implement some AQM mechanism to manage queue lengths, reduce end-to-end latency, and avoid lock-out phenomena within the Internet.[1]
2. Deployed AQM algorithms SHOULD support Explicit Congestion Notification (ECN) as well as loss to signal congestion to endpoints.
3. AQM algorithms SHOULD NOT require tuning of initial or configuration parameters in common use cases.
4. AQM algorithms SHOULD respond to measured congestion, not application profiles. Other methods exist, e.g. Differentiated Services or Pre-Congestion Notification (PCN), that can be used to differentiate and police classes of applications.
5. AQM algorithms SHOULD NOT interpret specific transport protocol behaviors.
6. Congestion control algorithms for transport protocols SHOULD maximize their use of available capacity (when there is data to send) without incurring undue loss or undue round-trip delay.
7. Research, engineering, and measurement efforts are needed regarding the design of mechanisms to deal with flows that are unresponsive to congestion notification or are responsive, but are more aggressive than present TCP.

[1] In some situations tail drop allows a single connection of a few flows to monopolize the queue space, thereby starving other connections, preventing them from getting room in the queue.

3.1.2 Repurposing the existing Active Queue Management (AQM) equipment

Autotuning RED Hardware for Link Rate Changes

A considerable range of network equipment already provides support for Active Queue Management (AQM), typically in the form of an implementation of the Random Early Detection (RED) algorithm [22]. RED has often not been turned on by network operators, because it is renowned for being hard to configure correctly. One reason is that the queue thresholds of RED are defined in bytes so they have to be reconfigured for different link rates. For instance, if a RED threshold is configured at 60KB for a link rate of 10Mb/s it has to be scaled up to 600KB for a link rate of 100Mb/s.

More modern AQMs such as PIE [23] or fq_CoDel [24] measure the queue in time, which makes them insensitive to changes in link rate. Many equipment designers believe that they should implement one of these new algorithms, to make their AQM easier to manage. However AQM algorithms are often implemented in hardware, so this can involve considerable development expense and upheaval. Further, there is little performance difference between different AQMs—the mere existence of any AQM at all gives most of the benefit compared to no AQM.

Therefore, where RED hardware already exists, rather than embark on a complete hardware reimplementation, we recommend adding a management harness around the existing hardware. Then the benefits of AQM can be made available rapidly to all existing users, without having to wait for the next round of hardware purchases.

This management harness should capture any change to the link rate and automatically reconfigure all the RED parameters so that they stay constant when measured in time units, even though internally they are defined in byte units. There are two possible approaches to deployment:

- It could be implemented by the network operator in its management system. For instance, many operators of Digital Subscriber Line (DSL) access networks arrange for their DSL access multiplexer (DSLAM) to signal any change of line rate to the Operational Support System (OSS). In turn the OSS signals the change to the relevant Broadband Network Gateway (BNG) and reconfigures the drain rate of the packet shaper for the relevant line to just less than the new access line rate, so that the downstream bottleneck is always in the BNG. If the BNG is configured to operate the RED algorithm within the queue(s) feeding this shaper, the management system could simultaneously issue reconfiguration commands to scale the RED parameters.
- It could be implemented by the equipment vendor or chipset supplier as software encapsulating the RED hardware. For instance, this might be appropriate in a residential DSL or cable modem, so that the RED algorithm being applied to upstream traffic would automatically reconfigure to match the line rate.

This management harness approach is only recommended where the line rate changes infrequently. For continual rapid link rate changes, e.g. in IEEE 802.11 WiFi, the implementation needs to integrate measurement of queuing time, to avoid continual parameter changes that are not coordinated with the state of the queue(s).

Note that RED has at least 4 parameters, and at least 3 of them should be related to the link rate:

- `min_th`, the minimum threshold queue length;
- `max_th`, the maximum threshold queue length (often defined as a multiple of `min_th` anyway);
- `w_q` or $2^{\texttt{exponential\_weighting\_constant}}$, the constant of the exponentially weighted moving average (EWMA) of queue length.

`min_th` and `max_th` should be scaled proportionate to link-rate. How to scale `w_q` depends on the implementation. If `w_q` or or $2^{\texttt{exponential\_weighting\_constant}}$ count in packets, they should also scale proportionate to link rate, so that they represent the same amount of time (assuming the distribution of packet sizes is unchanged). Of course, the value of `exponential_weighting_constant` will scale with the $\log_2$ of link-rate, because it is an exponent. For some hardware, the EWMA constant is defined in time units, in which case it should be invariant with link rate.

Allowing Distinct ECN configuration within Existing AQMs

Where ECN is supported by an existing AQM, previous advice has been to configure the ECN side of any AQM algorithm with identical parameters to the non-ECN side. The RITE project and others have shown that ECN has considerably more potential if its behaviour is not tied to loss behaviour, whether in the network [25] on the host [26] or both [27, 28].

Where the ECN behaviour is significantly different, a distinct identifier is needed for the packets. However, it has been shown that at least the burst tolerance in an AQM can be removed (or significantly reduced) for ECN packets to remove the smoothing delay without significantly affecting capacity sharing between flows.

Therefore, we have made sure that RFC 7567 [21] recommends that AQMs allow distinct configuration parameters for ECN and non-ECN packets. We also recommend the software drivers for existing hardware to be (re)designed to allow distinct ECN configurations. This applies particularly to a wide range of hardware that supports Weighted RED [29, 30, 31]. All that is necessary to support distinct ECN configuration is for the packet classification at the front-end of the WRED algorithm to be able to distinguish packets based on the ECN field, not just the Diffserv field. Given the ECN field comprises the last 2 bits of the 8-bit field that used to specify type of service (ToS), before it was split into Diffserv and ECN, this should not be difficult, and may already have been allowed on some systems.

Once such distinct packet classification is possible, it is only necessary to create two WRED configurations for the same traffic class but with differences between ECN and non-ECN packets.

3.1.3 ECN Encapsulation Guidelines

AQM is not only applicable to queues at the IP layer; it is applicable wherever there is a queue. Unless a packet is ECN-capable, there is nothing additional to do to make AQM work at any layer—dropping a frame at the link layer obviously also drops the packet encapsulated within the frame, so the end-to-end transport protocol will still detect the loss and respond accordingly.[2]

However, it is more difficult to make an AQM at a lower layer correctly interwork with ECN at the IP layer. First the lower layer has to somehow only mark frames that contain an ECN-capable IP packet(s), and secondly, the marking has to somehow be applied to any IP header(s) encapsulated within the frame.

The RITE project has not attempted to solve this problem for every specific link-layer technology. Instead, we have produced guidelines on how to correctly design ECN marking for any protocol that might encapsulate IP packets, so that it interworks correctly with IP [32]. We have also arranged for the IETF to liaise with other standards bodies, specifically the IEEE and the 3GPP, so that the designers of all lower layer protocols will use these guidelines as they consider how to add AQM and ECN to their specific link-layer protocols. This is necessary because when ECN was standardized in 2001, it represented a change to the IP protocol. Therefore all lower layer protocols that had been designed against the original lower interface of IP now need to be redesigned.

Recognizing that every link-layer protocol is different, the guidelines propose four arrangements of feedback, such that any lower layer protocol should match one of the patterns. It is then possible to give generic guidelines applicable to each arrangement, and therefore cover all known lower layer protocol approaches in one document.

As well as IETF liaison with other standards bodies, it has been necessary to liaise with certain working groups within the IETF that are standardizing new link-layer protocols, or new tunnelling encapsulations of IP. For instant the TRansparent Interconnection of Lots of Links (TRILL) WG has already used the draft guidelines, and presentations are planned for the Network Virtualization Overlays (NVO) Working

[2]If frames are larger than packets, dropping one frame will discard multiple packets. If frames (cells) are smaller than packets, dropping one cell will still cause a whole packet to need to be discarded.

Group and the Internet Area WG (int), which is responsible for the Generic Routing Encapsulation (GRE) and many other tunnelling protocols.

3.1.4 Squared AQM

One of the difficulties of defining and configuring AQM algorithms is caused by the non-linearity of the control signal (drop or mark) with respect to the rate that classic TCP endpoints are sending. To be ”TCP friendly” classic TCP implementations need to produce a sending rate that is inversely proportional to the square root of the signal probability. As a result the load or number of flows is proportional to the square root of the signal probability.

This non-linearity has a few consequences for the design and configuration of AQMs:

- Non-linear algorithms are usually too complex to handle at the packet rate. As a result complex algorithms are performed at a lower rate, or heuristic/empirical algorithms are defined that have no theoretical basis but experimentally behave ”good enough”.
- The operational range, with respect to the number of flows, of an AQM is bounded. It depends on the range where the configuration parameters are effective (producing results that are within the objectives of the AQM).
- Adaptation algorithms are defined to extend the operation range. Typically they are applied stepwise, potentially adding oscillation-like effects to the traffic.
- To handle square roots of numbers smaller than 1 (typically smaller than 1%), twice the precision is required for the internal variables, compared to the non-square-rooted values.
- The AQM cannot be used to control scalable congestion controllers.

From our work in the network signals activity on AQMs for scalable congestion controllers (meaning the number of flows or load is proportional to the signal probability) we came to the conclusion that classic AQMs can be linearised in a very simple way. Figure 3.1 shows the concept of ”Thinking Twice at the Output”. By applying a squaring function just before applying the signal probability, the AQM can work on the linear scalable probability, and the feedback to the congestion controller is adapted to the square root that is part of the response function for classic congestion controllers. A squared probability is very simple to implement. If p is compared with 2 random values instead of one, the probability is effectively squared. Generating (or lookup of pre-generated) random values is done at line speed today. Applying this simple extension removes all above mentioned disadvantages. From a configuraton point of view, configuring parameters that depend on RTT and delay were linear already. A squared AQM allows parameters that depend on the marking/dropping probability to become linear as well, and will apply to scalable congestion controllers too.

The following subsection briefly describe the benefits and simplifications when a squared AQM design is applied to a PIE AQM, but the application of this concept might be useful in future designs of AQMs.

3.1.4.1 A simpler PIE: PI^2

PIE [33] has been proposed as an improvement to the original PI controller based AQM. Several improvements on PIE have been proposed, but it can be both simplified and optimized by the ”Thinking Twice at the Output” concept.

As the PI controller needs to control a non-linear system (Classic TCP flows), it needs to adapt its proportional (β) and integral (α) gain factors to the level of congestion. The PIE implementation proposes to ”correct” α and β based on the current p value in a stepwise manner, to keep the control process in a stable regime.

Figure 3.2 shows our PI^2 proposed implementation that avoids the need for the stepwise gain factor adaptations, which results in full load-spectrum support, while still using a normal PI controller. The PIE paper [34] proposed initially to divide the gain factors by 2 for p $<$ 10% and to divide by 8 for p $<$ 1%. Table 3.1 shows the changes in the output drop probability of d on different input deviations for both

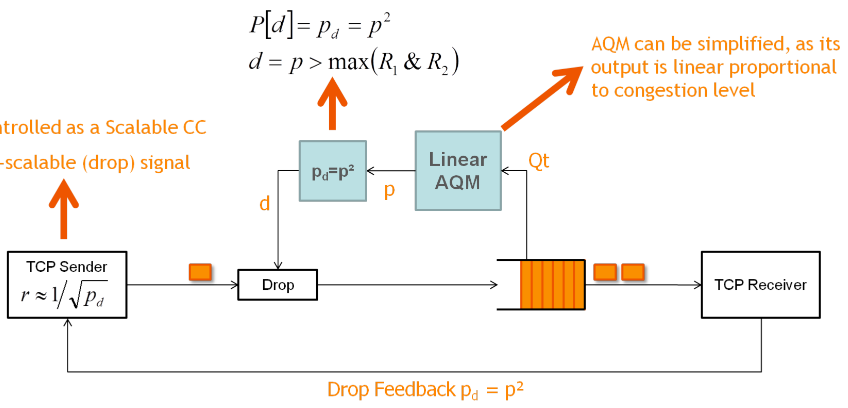

$P[d] = p_d = p^2$
$d = p > \max(R_1 \& R_2)$
AQM can be simplified, as its output is linear proportional to congestion level
Can be controlled as a Scalable CC with a non-scalable (drop) signal
$p_d=p^2$
Linear AQM
Qt
p
d
TCP Sender
$r \approx 1/\sqrt{p_d}$
Drop
TCP Receiver
Drop Feedback $p_d = p^2$

Figure 3.1: Linearity of Thinking Twice at the Output of an AQM

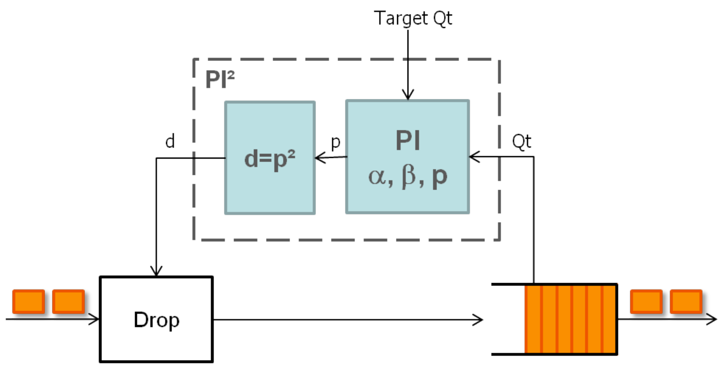


Figure 3.2: PI^2: no need to autotune α and β to level of drop probability

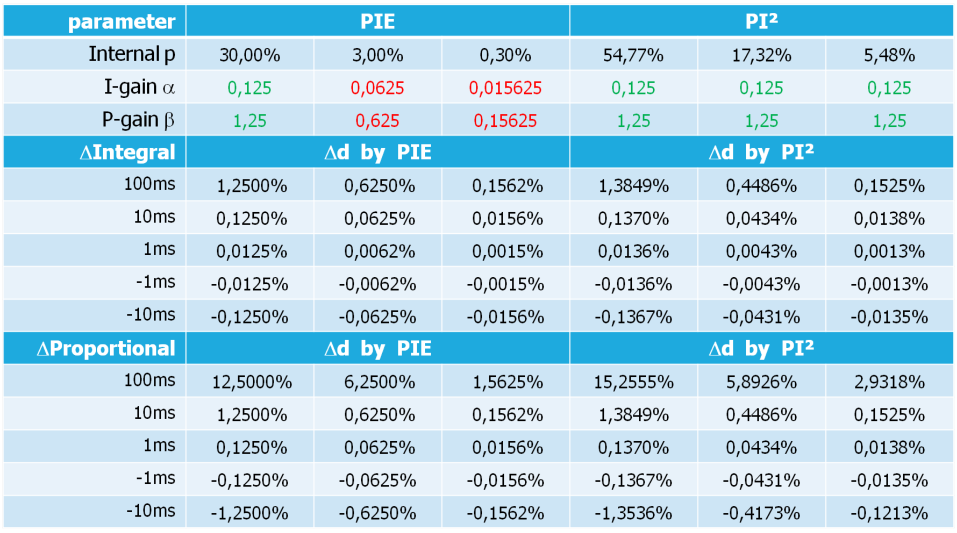
PIE and PI^2. As can be seen, the output values of PIE and PI^2 are very close together, suggesting that the PI^2 controller will work at least as efficiently as the PIE controller, and over a wider operating range. Additionally the output of the PI controller can be applied directly for marking scalable congestion controlled flows.

3.2 Configuration parameters

The problem of tuning AQM parameters correctly has likely hampered AQM deployment significantly over the years, as it is very hard to make general parameter suggestions or configuration guidelines. The RITE project has, through several studies, made an effort to mitigate this problem. This section presents configuration guidelines based on key observations from some of these studies.

3.2.1 AQM: Operating ranges and tunability

RFC 7567 [21] recommends deployment of a modern AQM technique (e.g. FQ-Codel or PIE) to manage the queues in network devices for the general Internet, where an AQM can auto-tune to network characteristics based on its default configuration parameters. This report confirms this recommendation. However, RFC 7567 also notes that although the default parameters of these AQM techniques can be applied across a range of deployment environments, and are applicable to a wide range of environments using default parameterisation, they are expected to require tuning for some specific types of links or networks. The need for tuning parameters is determined by the network characteristics (bottleneck capacity, path RTT) for which the AQM schemes are designed. These operating bounds have been



parameter	PIE			PI²		
Internal p	30,00%	3,00%	0,30%	54,77%	17,32%	5,48%
I-gain α	0,125	0,0625	0,015625	0,125	0,125	0,125
P-gain β	1,25	0,625	0,15625	1,25	1,25	1,25
ΔIntegral	Δd by PIE			Δd by PI²		
100ms	1,2500%	0,6250%	0,1562%	1,3849%	0,4486%	0,1525%
10ms	0,1250%	0,0625%	0,0156%	0,1370%	0,0434%	0,0138%
1ms	0,0125%	0,0062%	0,0015%	0,0136%	0,0043%	0,0013%
-1ms	-0,0125%	-0,0062%	-0,0015%	-0,0136%	-0,0043%	-0,0013%
-10ms	-0,1250%	-0,0625%	-0,0156%	-0,1367%	-0,0431%	-0,0135%
ΔProportional	Δd by PIE			Δd by PI²		
100ms	12,5000%	6,2500%	1,5625%	15,2555%	5,8926%	2,9318%
10ms	1,2500%	0,6250%	0,1562%	1,3849%	0,4486%	0,1525%
1ms	0,1250%	0,0625%	0,0156%	0,1370%	0,0434%	0,0138%
-1ms	-0,1250%	-0,0625%	-0,0156%	-0,1367%	-0,0431%	-0,0135%
-10ms	-1,2500%	-0,6250%	-0,1562%	-1,3536%	-0,4173%	-0,1213%

Table 3.1: Output comparison between the stepwise corrected PIE at its targeted midpoints and the full load-spectrum corrected linear PI^2 AQMs.

delimited by RITE (more information can be found in D2.3 public deliverable):

1 $base_{RTT} \geq 200$ ms;

2 $C_{bot} < 2$ Mbps;

3 $C_{bot} > 100$ Mbps.

PIE or CoDel can be tuned to obtain different trade-offs than the default, by changing τ and λ in CoDel and τ, alpha and beta in PIE.

As one example, they have been tuned to alleviate the impact of buffering on latency sensitive traffic, while preserving bulk throughput performance for cable modems networks: The CableLabs DOCSIS 3.1 specification includes requirements for cable modems [35, 36].

With appropriate tuning, the AQM techniques can be effective for other network characteristics, i.e., tuning to achieve a high bottleneck utilisation and a low queuing delay outside of their operating bounds. PIE has been tuned to work within data centres, where flows typically experience a low RTT [23]. RITE has proposed parameters for a rural satellite broadband network for PIE and CoDel [37].

AQM self-tuning techniques continue to evolve. The RITE project has offered guidance in two main ways on this evolving area:

- The RITE project has highlighted that tuning for RTT is only necessary for drop-based AQMs. This is because drop is an impairment so the network has to smooth out sub-RTT bursts of drop. In contrast, the network can send bursts of ECN signalling without any smoothing, because ECN is solely a signal, and not an impairment. Each host transport can smooth out these ECN bursts itself, and inherently knows the timescale to smooth over—its own RTT. This recommendation to avoid having to tune RTT by using ECN and not smoothing it in the network has been emphasised on at least two occasions [38, 39].
- The RITE project has published initial draft recommendations on how to scale the configuration of an AQM with link rate, and expected number of flows, RTT, and packet size (see § 3.2.2).

3.2.2 Recommendations on Scaling AQM Configuration (Analytical Model)

Late in the RITE project we conducted an initial analysis of how AQM configuration should scale with numbers of flows, link capacity, RTT and packet size [40, § 2.2]. The main insight in this report is that the balance between queuing delay and loss chosen for one scale should be invariant for all scales. Even though flow aggregation at high scale reduces variation in queuing delay, TCP performance drastically drops if one exploits that opportunity to reduce delay by causing drop to increase. This dilemma can be resolved by overprovisioning. The formula given for the necessary level of overprovisioning to be able to minimise queuing delay without TCP causing loss to increase is:

$$\frac{X'}{X} = \frac{(D_R + d_q^*)}{(D_R + d_q^*/\sqrt{n})} \ ,$$

where X' is the necessary overprovisioned capacity, X is the capacity without overprovisioning, D_R is the expected average base RTT, d_q^* is the target queueing delay and n is the number of flows.

For example, a link with an average base RTT of $D_R = 20ms$ could carry $100\times$ more flows than currently and have the target queueing delay reduced from 20 ms to 2 ms while keeping the loss at e.g. 2%, if the aggregated capacity is over-provided by $(20 + 20)/(20 + 2) = 1.8$. That is, $180\times$ more capacity for $100\times$ more flows.

4 Future considerations from RITE research

RITE has experimented with and developed a number of low-latency focused end-host mechanisms based on novel design considerations. However, these mechanisms are experimental and may be hard to evaluate due to lack of deployment of support mechanisms. The following section makes some recommendations for future considerations based on the acquired knowledge from the experiments performed within the project.

4.1 Scalable and frequent ECN marking

In the RITE project, network mechanisms were developed to support compatibility between classic TCP congestion controllers, e.g. New Reno or Cubic, and L4S TCP congestion controllers (Low Loss Low Latency and Scalable with throughput), such as TCP-Prague, DCTCP, Relentless TCP and Scalable TCP.

L4S TCP congestion controllers (CC) will support both low queuing delay and low loss rates, and the signal frequency will scale with future link capacity rates. A frequent signal allows to mitigate throughput unfairness between competing flows, to quickly estimate whether link capacity is reached, and accurately estimate the level of congestion within an RTT.

An L4S CC design is recommended when Explicit Congestion Notification (ECN) is available, network equipment can support high marking rates with very shallow queues, and if needed, network equipment can support compatibility with Classic CC. An L4S CC will induce a high frequency congestion signal that will result in high amounts of loss if ECN is not used (packets will be dropped instead of marked).

Experiments with DCTCP on low delay paths show that many application issues disappear. Applications that use TCP with an L4S CC can enjoy:

- Hardly any packet drops, avoiding many retransmission timer issues
- Low latency using just TCP, no need to use UDP with alternative application layer congestion controls
- Stable throughput rate for, e.g., HAS and DASH streaming
- Fast feedback when link throughput varies

4.2 Circuit breakers

As network devices and end hosts move towards implementation of mechanisms that are designed to reduce latency, it is important to also consider the implications of these changes to the Internet architecture, and how the architecture will respond under overload. RITE work has contributed to an IETF Best Current Practice (BCP) document that explains the design and operation of a "network transport circuit breaker" (CB) [41]. This will be published within the RFC series. A network transport Circuit Breaker (CB) is an automatic mechanism that is used to estimate congestion caused by a flow, and to terminate (or significantly reduce the rate of) the flow when persistent congestion is detected. This is a safety measure to prevent congestion collapse (starvation of resources available to other flows), essential for an Internet that is heterogeneous and for traffic that is hard to predict in advance. [41] describes three types of CB and defines requirements for building a CB and the expected outcomes of using one within the Internet.

A CB is recommended when using network tunnels, and other non-congestion controlled applications. Although a CB does not immediately impact the end-to-end latency of networks, use of CBs can prevent starvation of resources, restoring the low latency operational point of a network after it has experienced severe congestion. The long-term latency that flows experience when sharing with other flows can be protected when the other flows implement an appropriate circuit breaker mechanism.

5 Conclusions

This document has summarised recommendations for designing and configuring both end-hosts and network components to enable low-latency communication over the Internet. The recommendations are all outcomes of analysis and experimentation work done within work package 3 of the RITE project.

References

[1] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I.-J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing Internet latency: A survey of techniques and their merits," *IEEE Communication Surveys and Tutorials*, vol. PP, no. 99, 2014. [Online]. Available: http://dx.doi.org/10.1109/COMST.2014.2375213

[2] B. Opstad and J. Markussen, "Latency considerations for thin-stream Congestion Control," Simula Research Laboratory, Technical report, July 2015. [Online]. Available: https://riteproject.files.wordpress.com/2013/03/thin_wcnd_tech_report.pdf

[3] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer," RFC 6298 (Proposed Standard), Internet Engineering Task Force, June 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6298.txt

[4] R. Stewart, "Stream control transmission protocol," RFC 4960 (Proposed Standard), Internet Engineering Task Force, Sep. 2007, updated by RFCs 6096, 6335. [Online]. Available: http://www.ietf.org/rfc/rfc4960.txt

[5] P. Hurtig, A. Brunstrom, A. Petlund, and M. Welzl, "TCP and SCTP RTO restart," Internet Draft draft-ietf-tcpm-rtorestart, work in progress, October 2015. [Online]. Available: http://tools.ietf.org/html/draft-ietf-tcpm-rtorestart

[6] N. Dukkipati, N. Cardwell, Y. Cheng, and M. Mathis, "Tail Loss Probe (TLP): An algorithm for fast recovery of tail losses," Internet Draft draft-dukkipati-tcpm-tcp-loss-probe, work in progress, February 2013. [Online]. Available: http://tools.ietf.org/html/draft-dukkipati-tcpm-tcp-loss-probe

[7] M. Rajiullah, P. Hurtig, A. Brunstrom, A. Petlund, and M. Welzl, "An evaluation of tail loss recovery mechanisms for TCP," *ACM SIGCOMM Computer Communications Review*, vol. 45, no. 1, pp. 5–11, January 2015.

[8] R. project, "RTO-restart and TLP-restart implementations for Linux." [Online]. Available: http://riteproject.eu/projects/wp1-end-systems-and-applications/rto-restart/

[9] B. R. Opstad, J. Markussen, I. Ahmed, A. Petlund, C. Griwodz, and P. Halvorsen, "Latency and Fairness Trade-Off for Thin Streams using Redundant Data Bundling in TCP," in *Proceedings of IEEE LCN*, Clearwater Beach, 2015.

[10] R. project, "Redundant data bundling implementation for Linux." [Online]. Available: http://riteproject.eu/resources/redundant-data-bundling/

[11] G. Fairhurst, A. Sathiaseelan, and R. Secchi, "Updating TCP to support rate-limited traffic," RFC 7661 (Experimental), Internet Engineering Task Force, October 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7661.txt

[12] R. project, "New congestion window validation (new-CWV) implementation for Linux." [Online]. Available: https://github.com/rsecchi/newcwv

[13] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain, "TCP fast open," RFC 7413 (Experimental), Internet Engineering Task Force, December 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7413.txt

[14] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig, "Early retransmit for TCP and Stream Control Transmission Protocol (SCTP)," RFC 5827 (Experimental), Internet Engineering Task Force, April 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5827.txt

[15] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing tcp's initial window," RFC 6928 (Experimental), Internet Engineering Task Force, Apr. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6928.txt

[16] M. Allman, S. Floyd, and C. Partridge, “Increasing tcp’s initial window,” RFC 3390, Internet Engineering Task Force, October 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3390.txt

[17] M. Welzl and G. Fairhurst, “The benefits to applications of using Explicit Congestion Notification (ecn),” Internet Draft draft-ietf-aqm-ecn-benefits-05, work in progress, June 2015. [Online]. Available: http://tools.ietf.org/html/draft-ietf-aqm-ecn-benefits-05

[18] K. Ramakrishnan, S. Floyd, and D. Black, “The addition of explicit congestion notification (ECN) to IP,” RFC 3168 (Proposed Standard), Internet Engineering Task Force, Sep. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3168.txt

[19] M. Westerlund, I. Johansson, C. Perkins, P. O’Hanlon, and K. Carlberg, “Explicit congestion notification (ECN) for RTP over UDP,” RFC 6679 (Proposed Standard), Internet Engineering Task Force, Aug. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6679.txt

[20] B. Trammell, M. Kühlewind, D. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger, “Enabling internet-wide deployment of explicit congestion notification,” in *Proceedings of the 16th Passive and Active Measurement Conference*, New York, NY, USA, 2015, pp. 193–205.

[21] F. Baker and G. Fairhurst, “IETF recommendations regarding active queue management,” RFC 7567 (Best Current Practice), Internet Engineering Task Force, Jul. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7567.txt

[22] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993. [Online]. Available: http://dx.doi.org/10.1109/90.251892

[23] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, “Pie: A lightweight control scheme to address the bufferbloat problem. further studies– pie for data centers,” in *IETF 86*, 2013.

[24] T. Hoeiland-Joergensen, P. McKenney, D. Täht, J. Gettys, and E. Dumazet, “Flowqueue-codel,” Internet Draft draft-hoeiland-joergensen-aqm-fq-codel, work in progress, Jun. 2014. [Online]. Available: http://tools.ietf.org/html/draft-hoeiland-joergensen-aqm-fq-codel

[25] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, and Y. Zhang, “Tuning ECN for Data Center Networks,” in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT ’12. New York, NY, USA: ACM, 2012, pp. 25–36. [Online]. Available: http://doi.acm.org/10.1145/2413176.2413181

[26] N. Khademi, M. Welzl, G. Armitage, C. Kulatunga, D. Ros, G. Fairhurst, S. Gjessing, and S. Zander, “Alternative Backoff: Achieving Low Latency and High Throughput with ECN and AQM,” Swinburne University of Technology, CAIA Technical Report CAIA-TR-150710A, Jul. 2015. [Online]. Available: http://caia.swin.edu.au/reports/150710A/CAIA-TR-150710A.pdf

[27] M. Kühlewind, D. P. Wagner, J. M. R. Espinosa, and B. Briscoe, “Using Data Center TCP (DCTCP) in the Internet,” in *Under submission*, Jul. 2014.

[28] K. de Schepper, O. Bondarenko, I. Tsang, and B. Briscoe, “Data Center to the Home,” RITE Project, Technical report, Jun. 2015. [Online]. Available: http://riteproject.eu/publications/

[29] D. D. Clark and W. Fang, “Explicit allocation of best-effort packet delivery service,” *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, Aug. 1998.

[30] C. Systems, “Distributed Weighted Random Early Detection,” Cisco Systems, Release Note Cisco IOS Release 11.1 CC and Feature Modules, 2002. [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf

[31] Alcatel-Lucent, “7750 SR OS 9.0.R1; Quality of Service Guide,” Alcatel-Lucent, Manual Document Part Number: 93-0077-08-01, Mar. 2011. [Online]. Available: https://infoproducts.alcatel-lucent.com/cgi-bin/dbaccessfilename.cgi/9300770801_V1_7750%20SR%20OS%20QUALITY%20OF.pdf

[32] B. Briscoe, J. Kaippallimalil, and P. Thaler, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP," Internet Draft draft-ietf-tsvwg-ecn-encap-guidelines-04, work in progress, Oct. 2015. [Online]. Available: http://tools.ietf.org/html/draft-ietf-tsvwg-ecn-encap-guidelines

[33] R. Pan, P. Natarajan, F. Baker, G. White, B. VerSteeg, M. Prabhu, C. Piglione, and V. Subramanian, "PIE: A lightweight control scheme to address the bufferbloat problem," Internet Draft draft-ietf-aqm-pie, work in progress, October 2015. [Online]. Available: http://tools.ietf.org/html/draft-ietf-aqm-pie

[34] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A lightweight control scheme to address the bufferbloat problem," in *Proceedings of the IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*, 2013, pp. 148–155.

[35] C. Greg White, "Active queue management algorithms for DOCSIS 3.0: A simulation study of CoDel, SFQ-CoDel and PIE in DOCSIS 3.0 networks," *Cable Television Laboratories*, 2013.

[36] G. White and R. Pan, "A pie-based aqm for docsis cable modems," 2014, iETF. [Online]. Available: http://tools.ietf.org/html/draft-white-aqm-docsis-pie-00

[37] C. Kulatunga, N. Kuhn, G. Fairhurst, and D. Ros, "Tackling bufferbloat in capacity-limited networks," *European Conference on Networks and Communications (EUCNC)*, 2015.

[38] B. Briscoe, M. Kühlewind, D. Wagner, and J. M. R. Espinosa, "Immediate ECN," Presentation in IETF Proceedings, URL: http://www.ietf.org/proceedings/88/slides/slides-88-tsvwg-20.pdf, Nov. 2013.

[39] B. Briscoe, M. Kühlewind, D. Wagner, and K. de Schepper, "ECN; the identifier of a new service model," Presentation in IETF Proceedings, URL: www.ietf.org/proceedings/89/slides/slides-89-tsvarea-3.pdf, Mar. 2014.

[40] B. Briscoe, "Insights from Curvy RED (Random Early Detection)," BT, Technical report TR-TUB8-2015-003, May 2015. [Online]. Available: http://www.bobbriscoe.net/projects/latency/credi_tr.pdf

[41] G. Fairhurst, "Network transport circuit breakers," Internet Draft draft-ietf-tsvwg-circuit-breaker, work in progress, October 2015. [Online]. Available: http://tools.ietf.org/html/draft-ietf-tsvwg-circuit-breaker

Appendices

A ECN Support in Consumer Devices

RITE tested a selection of modern wireless routers targeted at home use to determine if they correctly passed ECN signalling. The routers tested were:

- D-Link Wireless AC750 Dual Band Cloud Router
- NewLink Wireless Lite-N 150 Access Point Router
- Netgear N300 Wireless Router with External Antennas
- Netgear RangeMax Wireless Cable Router (108 Mbps)
- TP-Link 150Mbps Wireless N Nano Router

These routers were from a range of manufacturers and ran diverse firmware based on Linux, vxWorks and custom kernels.

It was discovered that all the routers did successfully allow negotiation of ECN for TCP connections and that ECN signalling information was passed successfully also. The Netgear N300 router even negotiated ECN when connecting to the web-based configuration interface, and this has been attributed to "server-mode" ECN being enabled by default in recent Linux kernels.