Topologies for the Provision of Network-Coded Services via Shared Satellite Channels

Ulrich Speidel 'Etuate Cocker Muriel Médard Péter Vingelmann, Janus Heide

The University of Auckland, Auckland, New Zealand
email:ulrich@cs.auckland.ac.nz

Spark Digital NZ Ltd. Auckland, New Zealand
email:Etuate.Cocker@spark.co.nz

RLE, MIT Cambridge, USA
email:medard@mit.edu

Steinwurf ApS Aalborg, Denmark
email:{peter|janus}@steinwurf.com

***Abstract*—Network traffic using the Transmission Control Protocol (TCP) across shared bottleneck satellite channels can suffer significant impairment due to TCP queue oscillation. Coding of such network traffic across multiple Internet Protocol (IP) packets allows packet loss to be masked from the senders, letting TCP senders sustain higher goodput rates. We argue that the concept of tunneling coded traffic across a satellite link is a flexible one and does not rely on a one-size-fits-all solution. This paper discusses a number of network topology options for the deployment of coding, from the perspective of satellite providers, Internet service providers, end users and third-party entities.**

Keywords–TCP/IP; network coding; network topologies; queue oscillation; tunneling.

I. Introduction

Consider the following scenario: An Internet Service Provider (ISP) on a small Pacific island receives its international connectivity via a geostationary (GEO) or medium earth orbit (MEO) satellite service. The capacity provisioned is in the range of several Mbps to several 100 Mbps, but always well below that of the networks connected at either end (assumed to be 1 Gbps or faster). The ISP services users on the island. The number of concurrently active client devices could be anywhere from a few dozen to a couple of thousand, and the ISP might observe up to a few thousand simultaneous TCP flows. For the purposes of this paper, a TCP flow is a set of TCP packets travelling in one direction and characterised by a unique combination of source and destination IP addresses and ports [1]. Each flow typically belongs to a single TCP connection (i.e., a connection typically consists of two flows in opposing directions).

The flows across the link will typically be a heavily skewed mix: Most flows on the link will contain at most a few hundred bytes and will be too small and short to have their rate controlled by TCP flow control (also known as congestion control). Long flows, which are subject to flow control, contribute the majority of bytes on the link, however.

Satellite links of this type present a significant challenge to TCP: The long latency bottleneck makes it difficult for TCP senders to find the correct congestion window [2]. Moreover, a large number of simultaneous connections face exactly the same congestion situation here. This causes the TCP senders involved to act in unison when adjusting their congestion windows, an effect known as *global synchronisation*. It can lead to *TCP queue oscillation*, where the input queue to the satellite link oscillates between empty and overflow, causing link underutilisation when the queue is empty. The resulting performance problem has been studied in the context of satellite links for over two decades (see, e.g., [3]) and remains essentially unsolved, despite the emergence of active queue management (AQM) techniques [4][5]. In large parts, this is due to senders overloading the queue based on feedback from the receivers that is already on its way when the queue shows signs of filling. Any feedback from explicit congestion notification or random early drops simply arrives too late to be useful.

Network coding [6] offers a potential part-remedy here: Error-correcting packet losses at the input queue can prevent premature back-off by the TCP senders, allowing some of the lost capacity to be reclaimed. The basic topology investigated in this paper is a tunnel, which operates across the link and both satellite input queues at either end. It accepts and delivers IP packets regardless of transport layer protocol involved, such that the end-to-end principle always remains intact. We have already demonstrated [7][8] that such a tunnel solution can improve goodput for individual TCP connections, even in the presence of a majority of legacy TCP traffic on the same link. Such tunnel-based solutions thus represent a potential alternative to and/or enhancement of performance-enhancing proxies (PEPs) [9], [10].

This paper investigates possible deployment scenarios for tunnel solutions of this kind. Section II explains the basic workings of the tunnel in a scenario where the ISP on the island provides the on-island tunnel endpoint and where the coded traffic between the tunnel endpoints travels in the payload of User Datagram Protocol (UDP) packets. On this basis, Section III discusses the question as to where the off-island tunnel endpoint could be located and presents a case for having multiple endpoints. Section IV describes a scenario in which a third-party entity operates the tunnel endpoint on the island. In Section V, we look at the advantages and drawbacks of offering coding-as-a-service tunnels to individual end users on an island. Section VI then looks at various options for non-UDP communication between the tunnel endpoints.

II. Coded Tunnels

We begin our tour of the basic tunnel model (Figure 1) by introducing our players and our components and following what happens during a TCP connection from an island end user client to a server off-island that the user wants to access:

The connection begins at an end user machine (client) on the island. In most scenarios, we will assume that we have no control over this machine, i.e., that we cannot assume anything beyond the existence of a TCP/IP stack and some TCP client program on the machine. It means in particular that we cannot

 ISBN: 978-1-61208-545-6

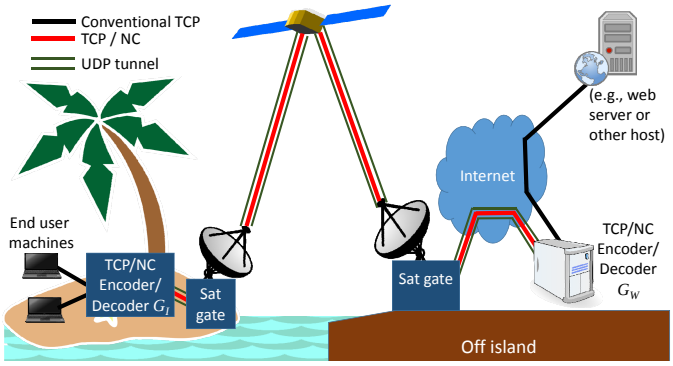

Figure 1. Network topology in a scenario where the on-island internet service provider operates the local encoder/decoder. The off-island encoder/decoder may be at an arbitrary off-island location on the internet.

install software on the machine or get the user to change machine settings. The client initiates the connection by sending a TCP packet to the off-island server, with the SYN flag set. Its TCP/IP stack encapsulates this packet in an IP packet whose source address is that of the end user machine. Its destination address is that of the off-island server.

The IP layer initially forwards this IP packet to a local gateway router on the island, and from there possibly along further gateway routers in the direction of the on-island satellite gateway. In our case, we replace one of these gateways by our on-island encoder/decoder G_I. Since our packet is heading off-island, we use G_I's encoder functionality here.

The encoding works as follows: G_I captures the original IP packet but does not forward it any further. Instead, G_I forms sets of n successive captured IP packets. Each such set is called a *generation* and n is the *generation size*. In random linear network coding (RLNC), G_I now creates $n + \omega$ byte-wise linear combinations of all packets p_j in the generation, using randomly (i.e., generally independently and uniformly) chosen coefficients c_{ij}. That is, the i'th linear combination r_i that G_I produces is given by:

$$c_{i1}p_1 + c_{i2}p_2 + \ldots + c_{in}p_n = r_i, \qquad (1)$$

The $n+\omega$ combinations thus produced form an overdetermined system of linear equations whose solution is the set of p_j. In doing so, G_I codes all bytes of the incoming packets, including the headers with the IP addresses of the island and off-island end hosts involved. G_I now communicates this system one equation at a time to G_W, located somewhere on the Internet on the off-island side of the satellite link. For this purpose, G_I sends $n + \omega$ UDP packets to G_I, with its own IP address as source and G_W's IP address as destination. Each UDP packet contains the equation for a particular i in the form of the c_{ij}'s and r_i.

These UDP packets now travel via the satellite link and the off-island Internet to G_W, which solves the system of linear equations. The solution consists of the p_j, of course, which G_W then forwards to their off-island destinations. Note that G_W generally only needs any n of the $n + \omega$ UDP packets in order to decode the p_j. The remaining ω UDP packets are surplus and may safely be dropped along the way – for example, at the input queue to the satellite gateway. The important point here is that we can leave it up to this queue to decide which packets to drop.

Our SYN packet has now arrived at its off-island server destination, and the server wishes to send a SYN+ACK in response. At this point, the network topology becomes critical: Most data flows *to* the island and it is important that we encode this direction in particular, so we need to ensure that the response packet finds its way to G_W. However, in most island scenarios, the island hosts including the satellite gateways at either end of the link belong to a single IP subnet. From the world's perspective, the off-island satellite gateway is also the IP gateway to this subnet, so the SYN+ACK response (and any subsequent packets) from the server would be routed straight to the off-island satellite gateway, entirely bypassing G_W. This is unacceptable, of course.

The solution is to split the subnet on the island: End hosts in the islands become part of a new subnet A (this could also be several subnets), whereas the on-island satellite gateway is placed in a disjoint subnet B. One then configures routing to make G_W becomes the gateway for traffic to A, and traffic to B is routed straight to the off-island satellite gateway.

In this scenario, the off-island server receives the SYN packet with a source from network A, and thus responds by forwarding the SYN+ACK to G_W. There, G_W encodes the packet in the same way G_I encodes packets in the opposite direction. It then forwards the coded packets inside UDP to G_I for decoding and release to the island end user machine. This completes the round-trip handshake. Further packets between the hosts follow the same path. That is, the packets travel trough a coded UDP *tunnel* between G_I and G_W and vice versa.

This scenario requires the ISP on the island to either operate G_W off the island, or contract an off-island entity to operate G_W on their behalf. In many cases, it will also be desirable to make at least network A an autonomous system (AS) for routing purposes, in which case G_W needs to be duplicated for redundancy. The current experimental software that we have been working with is capable of supporting two instances of G_W.

In the next sections, we will consider variations of this base scenario.

III. Tunnel Endpoints and Their Locations

Our basic tunnel scenario above assumes that G_W is located at an arbitrary location on the Internet. As long as the tunnel that it spans with G_I covers the satellite link, it can fulfill its purpose of masking packet loss at the satellite gateway input queues. There are however good reasons to consider the placement of G_W carefully. The following options may deserve consideration:

- G_W could be placed in the path between the Internet and the off-island satellite gateway (Figure 2), or even be a part of the satellite gateway hardware. In this case, network B could use private IP addresses, and G_W simply acts as a gateway for network A as in the previous scenario. That is, all machines on the island could be in the same subnet of an upstream provider's network, save the off-island facing interface of the on-island satellite gateway. For the ISP and/or their upstream provider, this removes the cost of

maintaining a separate block of public IP addresses or even a separate AS.

However, a placement at the satellite gateway requires the competent cooperation of whichever party controls the off-island satellite gateway: It needs to install and assist in commissioning G_W, or permit terminals with equivalent built-in functionality to be installed. In practice, one encounters a variety of scenarios, however: One ISP owns and controls both satellite gateways, another ISP owns and operates the island side only and contracts to a satellite provider and upstream ISP off-island, and yet another buys a turnkey solution from a satellite provider who also controls and services the on-island satellite gateway. We note in this context that especially in the latter case, satellite providers often already provide Wide Area Network (WAN) accelerators with network memory, parity packets and various other optimisation functions – inserting G_W or even G_I as part of such a solution would thus not be without precedent.

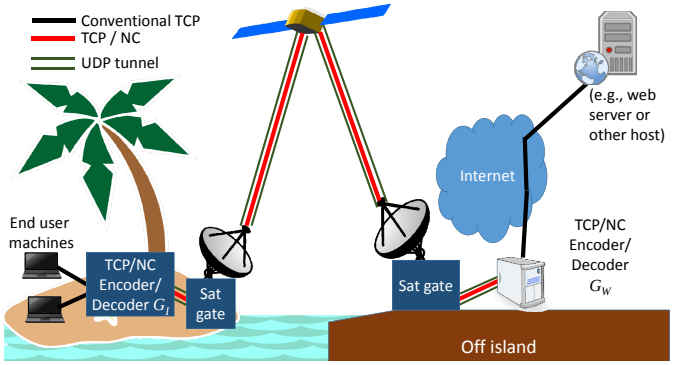

Figure 2. Network topology in a scenario where the on-island internet service provider operates the local encoder/decoder, and the off-island encoder/decoder is inserted in the path between off-island satellite gateway and the Internet.

- G_W (or several instances thereof, labelled G_{W1}, G_{W2}, etc.) could be placed close to the known primary sources of bulk data content sought by island clients (Figure 3). Such a placement would protect a longer portion of the paths between servers and clients by coding and would bridge other potential sources of loss. However, there is an obvious drawback: G_W is no ordinary server – as an AS gateway, it needs a significant amount of network configuration in its environment. Placing G_W in a site potentially far away from both ISP and satellite provider premises requires the cooperation of a third party able to host G_W and arrange for its routing needs. Such partners could potentially be difficult to recruit for an ISP based on a remote island.
- G_W could be placed at the premises of a specialised off-island provider, who could also own and operate the device and sell its encoding/decoding services to the ISP on the island. An obvious advantage of this model is that it allows a provider to specialise in this type of service and host the G_W for multiple island installations, achieving some economies of scale. A potential disadvantage is added latency: The latency between off-island satellite gateway, G_W and off-island data sources may be much higher than that between data sources and satellite gateway alone. This problem can be exacerbated by a failure to peer near the off-island satellite gateway. E.g., the authors are aware of a Pacific Island ISP whose off-island gateway is located in Hawaii. While the island has close cultural and economic links to New Zealand, lack of peering in Hawaii at the time of writing meant that all traffic between the island and New Zealand also has to travel between Hawaii and the U.S. mainland and back.

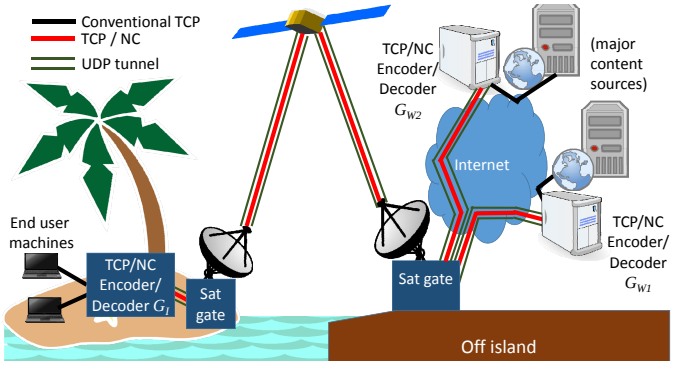

Figure 3. Network topology in a scenario where the on-island internet service provider operates the local encoder/decoder, and off-island encoders/decoders are placed close to the servers on the internet from which most of the download content originates.

IV. Tunnels Not Involving ISPs

In all our scenarios so far, the island ISP has played a core role as the operator of G_I if not G_W. However, in principle there is no reason why G_I cannot be operated by another party on the island. Assuming for the moment that the ISP and satellite provider will pass UDP in both directions, any of the ISP customers within the island network can operate a G_I to tunnel to some G_W located off-island. This customer can then spawn their own network (Figure 4) or – in the case of individual rather than institutional customers – simply run G_I on their own host or local network address translator (NAT) box.

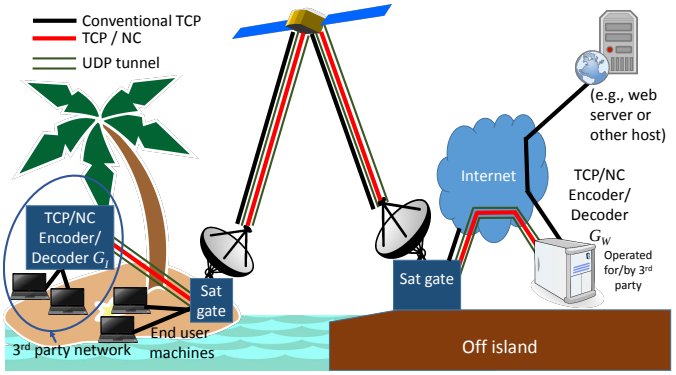

Figure 4. Network topology in a scenario where an on-island encoder/decoder is integrated into an end user machine on the island, and the off-island encoder/decoder is provided by a third party on the internet as a service, e.g., for a fee.

In the case of institutional customers, the corresponding G_W could be located at an organisation's off-island data centre or at the premises of a specialised off-island provider as discussed in the previous section. In the case of individuals, there could also be the option of G_W being provided on a subscription or pay-as-you-go basis by an off-island entity, as discussed in the next section.

Any such arrangement has a number of drawbacks, however. Firstly, it almost inevitably means that only some of the traffic on the link will be coded traffic, with the remainder being (mostly) conventional TCP. This residual uncoded traffic may still cause queue oscillation. While the coded traffic would be – at least to an extent – be protected from the associated packet loss and slow-down, the coding scheme involved would nevertheless have to provision sufficient overhead in order to cope with the potentially lengthy burst errors that queue oscillation causes. This would further increase the load on the link. However, this would be in parts offset by the fact that the link does not need to carry overhead for the uncoded part of the traffic.

Secondly, any overhead transmitted or received by a G_I under customer control increases that customer's data usage. In cases where the ISP on the island applies volume charges (a very common scenario in the Pacific), this results in additional cost for the customer. This may however be outweighed by data volume savings at the application layer as customers have to repeat fewer unsuccessful downloads.

V. Coding-As-A-Service Tunnels

Another possible scenario is to absorb G_I into a virtual network interface on the end user machine and provision G_W off-island on a subscription or pay-per-coded-volume basis (Figure 5). In this case, the end user would download an application which implements the client-side solution with G_I and interfaces with G_W off-island. The end user machine would then use two IP addresses: that assigned by the ISP, which appears in the header of the UDP packets between the machine and G_W, and an IP address assigned by the off-island provider of G_W, which belongs to the off-island provider's network and is not visible on the island to any host except G_I (which of course operates on the machine itself). This address is the source of IP packets departing G_W in the direction of off-island servers on behalf of the end user machine, and the destination of any packets that these servers send in response. In this respect, the service operates in a very similar fashion to a tunnelled VPN connection, except that the traffic across the tunnel is encoded rather than encrypted (it may of course also be encrypted in addition to the encoding).

An obvious advantage of this approach is that there is no need for dedicated on-island infrastructure, the ISP does not have to expend or upskill personnel resources (or even be aware of the tunnel operation), and there is no need for equipment or personnel to be sent to the island to install or support the system. These are significant factors as many Pacific islands with satellite connection are difficult to reach – air services may be infrequent or non-existent, and intervals between ship visits may be lengthy and freight is expensive. Similarly, many island ISPs struggle to hire and retain qualified personnel.

Naturally, there are also a number of drawbacks, which start with those discussed in the previous section. In addition,

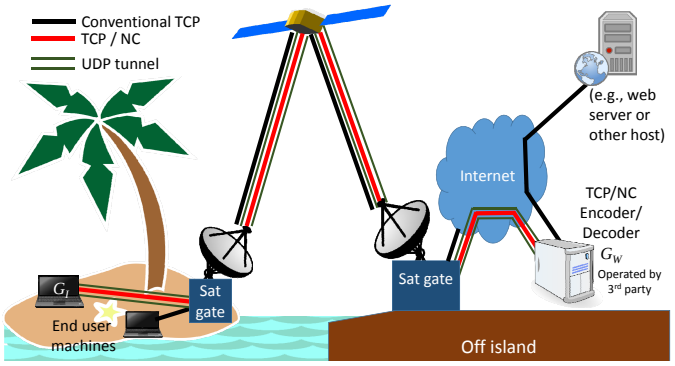

Figure 5. Network topology in a scenario where an on-island encoder/decoder is integrated into an end user machine on the island, and a third party provides the off-island encoder/decoder as a (potentially fee-paying) service on the internet.

there is now an additional challenge from the location perspective: As discussed in Section III, the latency between satellite gateway, G_W and data sources may be significantly higher than the latency between satellite gateway and data sources alone. If the specialised off-island provider of G_W implements a coding-as-a-service scenario at scale, it will inevitably find its client software used in multiple island locations, with satellite gateways in geographically dispersed locations: An island in the western Pacific can have its off-island GEO gateway in Canada, whereas an island in French Polynesia might opt for space segment terminating in Australia.

A further challenge is the diversity in consumer operating systems. To be able to serve a large majority of users, the off-island provider would need to supply the application implementing G_I on multiple popular operating systems such as Windows, MacOS, IOS and Android. This represents significant additional effort compared to a tunnel application on a single operating system of the implementor's choice. It also carries the risk of leaving the end user machine disconnected: The software needs to modify the network configuration of the machine. There could be unintended consequences if, in doing so, the software interferes with any of the myriad of network configuration managers, tools and utilities which commonly inhabit these ecosystems. Given that the off-island provider has no control over what else may be installed on the end user's machine, this risk could be substantial. Another question that arises in this context is how an island user would pay the off-island provider: Not every islander has access to a credit card.

VI. Connecting the Tunnel Endpoints

On many islands, ISPs and/or satellite providers block UDP to keep traffic off their satellite link that does not back off under congestion. This can backfire, however, as many applications that have higher bandwidth efficiency using UDP do sense congestion and will switch to less efficient TCP when UDP is blocked. It is worth noting in this context that the UDP carrying our coded packets *will* back off as well: If too many packets of a generation are lost, the generation as a whole will become undecodable and the TCP packets it contains are lost as well, causing the contributing TCP senders to back off, too. Conversely, the notion that TCP will always back off is also not true: Small TCP flows often complete before the first ACK

 ISBN: 978-1-61208-545-6

reaches the sender, such that the ACKs do not get to influence packet transmission rates.

However, the communication between G_I and G_W need not rely on UDP. There are several options for this, two of which are discussed below:

A. Spoofing TCP

One option is to pass the coded combinations as TCP packets without actually running TCP at G_I or G_W. The only differences to the UDP variant are as follows:

- The packets carry a TCP header instead of a UDP header, with nominal sequence and acknowledgment numbers
- G_I and G_W acknowledge any packet received but do not attempt to retransmit any packets not received (and in fact ignore any ACK received)
- The first and second packet from G_I to G_W have their SYN and SYN+ACK flags set, respectively, and the first packet G_W to G_I correspondingly has SYN+ACK set.
- Either end ignores flags and ACK numbers upon receipt and concentrates on the packet payload instead.

To an outside observer, such flows are almost indistinguishable from genuine TCP and practically impossible to detect or block on a firewall with stateful inspection. Even in a real TCP connection, an observer somewhere along the path may not get to see all packets of the connection due to load balancing and asymmetric paths. One disadvantage of this approach is that it is a hack and, from the ISP's perspective, could be considered improper use. Another is that it adds encapsulation overhead, as the size of a TCP header is larger than that of a UDP header.

B. Multiple TCP Connections

Another option would be to open multiple TCP connections between G_I and G_W at the outset and communicate only a small number of linear combinations (or even just one) per generation as data across each connection.

In scenarios where G_I and G_W are the only significant users on the satellite link, this has the advantage of replacing what would otherwise be a mix of TCP flows of varying lengths by a fixed number of TCP flows with infinite length and more or less equal data rate. Since the arrival of each combination is now ensured by TCP, one could also set $\omega = 0$ and thus reduce overhead to zero. However, TCP also adds its own overhead. It is also possible to use TCP variants optimised for long latency networks, such as Hybla [12] or H-TCP [11].

One potentially significant problem occurs at G_W (and possibly G_I, too), however: In the UDP or spoofed TCP scenarios, data arrives at G_W at full Gbps network rates and leaves in the direction of the sat gate at the same high rate in encoded form. So G_W does not need to buffer or concern itself with keeping any form of state once the coded packets of a generation have left. If we connect G_W and G_I via TCP, we transfer at least a significant part of the sat link bottleneck and its associated queue to G_W. Since TCP sockets cannot queue drop, G_W would need to implement this functionality *before* the linear combinations are written to the TCP connections with G_I.

VII. Conclusion and Future Work

Network-coded tunnels carrying TCP/IP traffic in coded form across lossy bottlenecks in satellite networks have been shown to be able to improve goodput under TCP queue oscillation conditions even in the presence of a majority of flows using legacy TCP. The core insight that underpins the tunnel concept is that packet losses occur by queue drop at the input queue to the satellite link. As long as one can protect traffic against data loss at this location, the remaining system topology is a question of who will or can provide the tunnel service, and how cooperative the local ISP and satellite provider are. We have discussed a variety of potential topologies along with their advantages and drawbacks. All are feasible: Coding across satellite links does thus not rely on a single solution topology.

As a general rule, topologies in which ISP and/or satellite provider are not involved (or even actively oppose the use of coded tunnels) are bound to be less effective: The presence of legacy TCP connections forces coded traffic to use more overhead, so any parties on the island with coded traffic consume more data and bandwidth than necessary. Those not using coding are also put at a potential disadvantage as this may eat into their bandwidth as well. Active involvement of satellite providers and/or local ISPs thus seems advantageous.

At the time of writing, only experimental implementations of coded tunnels are available. These are based on a Debian/Ubuntu Linux kernel module. While they do not cater for the subscription model discussed in Section V at this point in time, they nevertheless represent a proof of concept for the remaining scenarios. Current work aims to demonstrate that the technology scales to whole-of-island coding.

Acknowledgments

The authors wish to thank the Information Society Innovation Fund (ISIF Asia/APNIC) and Internet NZ for their financial support for this work. We would also like to thank Brian Carpenter and Nevil Brownlee, among others, for their support and many helpful suggestions and discussions. We are also grateful for the support received from the Pacific Island Chapter of the Internet Society and a number of ISPs in the Pacific.

References

[1] Clark, D. D., "The design philosophy of the DARPA Internet protocols", Proceedings of ACM SIGCOMM'88, pp.16–19, 1988.

[2] Jacobson, V. and Braden, R. *TCP Extensions for Long-Delay Paths*, RFC1072, 1988.

[3] Jouanigot, J. M. et al., "CHEOPS dataset protocol: an efficient protocol for large disk-based dataset transfer on the Olympus satellite", International Conference on the Results of the Olympus Utilisation Programme, 20-22 April 1993, Sevilla, Spain, CERN CN/93/6.

[4] Braden, B. et al., *Recommendations on Queue Management and Congestion Avoidance in the Internet*, RFC 2309, 1998.

[5] Kim, J. and Yeom, I., "Reducing Queue Oscillation at a Congested Link", IEEE Transactions on Parallel and Distributed Systems, 19(3), pp. 394–407, 2008.

[6] Sundararajan, J. K. et al., "Network Coding Meets TCP: Theory and Implementation", Proc. IEEE, 99(3), pp. 490–512, 2011.

[7] Speidel, U. , Cocker, 'E., Vingelmann, P., Heide, J., and Médard, M., "Can network coding bridge the digital divide in the Pacific?", International Symposium on Network Coding (NetCod), Sydney, pp. 86–90, 2015.

 ISBN: 978-1-61208-545-6

[8] Speidel, U. et al., “Can Network Coding Mitigate TCP-induced Queue Oscillation on Narrowband Satellite Links?”, International Conference on Wireless and Satellite Systems, pp. 301–314, Springer International Publishing, July 2015.

[9] Caini, C., Firrincieli, R., and Lacamera, D., “PEPsal: a Performance Enhancing Proxy designed for TCP satellite connections”, IEEE 63rd Vehicular Technology Conference, pp. 2607–2611, 2006.

[10] Delannoy, G., “Design and Implementation of a Performance-Enhancing Proxy for connections over 3G networks”, Dublin City University, May 27, 2013, https://github.com/GregoireDelannoy/TCPeP/blob/master/Final_Report.pdf [accessed: 2017-03-05].

[11] Leith, D., “H-TCP: TCP Congestion Control for High Bandwidth-Delay Product Paths”, Internet Draft, IETF, April 7, 2008, https://tools.ietf.org/html/draft-leith-tcp-htcp-06 [accessed: 2017-03-05].

[12] Caini, C., Firrincieli, R., “TCP Hybla: a TCP enhancement for heterogeneous networks”, Int. J. of Satellite Communications and Networks, 5(22), pp. 547-566, 2004.

 ISBN: 978-1-61208-545-6