QPEP: An Actionable Approach to Secure and Performant Broadband From Geostationary Orbit

James Pavur*, Martin Strohmeier†, Vincent Lenders†, and Ivan Martinovic*
* Oxford University
Email: first.last@cs.ox.ac.uk
†armasuisse
Email: first.last@armasuisse.ch

***Abstract*—Satellite broadband services are critical infrastructures, bringing connectivity to the most remote regions of the globe. However, due to performance concerns, many geostationary satellite broadband services are unencrypted and vulnerable to long-range eavesdropping attacks. This paper delves into the underlying cause of these issues, presenting the case that the widespread use of Performance Enhancing Proxies (PEPs) for TCP optimization has created a security/performance trade-off. A review of previous mitigation proposals finds limited real-world adoption due to a variety of factors ranging from misaligned commercial incentives to the prevalence of unverified "black-box" encryption products.**

To address these shortcomings, we design and implement a fully open-source and encrypted-by-default PEP/VPN hybrid, called QPEP. Built around the QUIC standard, QPEP enables individuals to encrypt satellite traffic without ISP involvement. Additionally, we present an open and replicable Docker-based testbed for benchmarking QPEP, and other PEP applications, through simulation. These experiments show that QPEP enables satellite customers to encrypt their TCP traffic with up to 72% faster page load times (PLTs) compared to traditional VPN encryption. Even relative to other unencrypted PEPs, QPEP offers up to 54% faster PLTs while also protecting communications in transit. We briefly discuss how QPEP might leverage bespoke modifications to the QUIC protocol for further optimization. Ultimately, our experiments suggest that QPEP's hybrid architecture represents a promising new technique for bringing both security and performance to satellite broadband while avoiding costly alterations to status-quo networks.

I. Introduction

Historically, security and performance have often traded-off in satellite broadband networks. As a result, many satellite internet service providers (ISPs) do not offer over-the-air traffic encryption, exposing sensitive customer data to eavesdropping attacks. This is because techniques used to optimize TCP connections in long-distance satellite links are often incompatible with commonly used encryption techniques, such as VPNs.

Since the early 2000s, academics and satellite operators have grappled with the challenge of offering both encrypted and performant TCP over satellite. In Section III we highlight notable proposals and discus why they have seen limited real-world adoption. While some encryption systems exist, these follow a "black-box" model and are inaccessible and costly for smaller organizational and individual customers. Moreover, their proprietary nature makes security and performance claims difficult to verify and most permit ISPs to eavesdrop on traffic.

No open-source encryption tool exists for performant TCP communications over satellite links. Although academic proposals are numerous, these are often purely theoretical or lack replicable source-code. As a result, interested researchers must either repurpose outdated code to incorporate modern encryption or reinvent PEPs from scratch. The combined requirements of cryptography and low-level network programming create steep barriers to entry. Moreover, the lack of standardized testing environments makes comparing approaches difficult without privileged access to satellite infrastructure.

The end result is that satellite broadband users have no good options. They (or their ISPs) must purchase expensive and unvetted proprietary applications, accept the substantial performance hit caused by general-purpose VPNs, or transmit sensitive data in clear text over massive satellite footprints.

This paper seeks to address both the lack of encryption options and high barriers to research in this domain. Its primary contribution is QPEP - an open-source and encrypted-by-default PEP. Unlike many proprietary encrypted PEPs, QPEP is designed for individual satellite customers and conceals traffic from both eavesdroppers and ISPs. Built around the open QUIC transportation protocol, QPEP benefits from robustly vetted cryptographic foundations and a broad technical community. The system is implemented in Go, an accessible modern language, to facilitate contributions in future research.

As a secondary contribution, the paper presents an open-source simulation testbed, built around the OpenSAND satellite networking engine [1]. This all-in-one dockerized environment is tailored towards rapid and replicable benchmarking for secure PEP applications. While it has immediate utility in our evaluation of QPEP, it is also designed to ease future system proposals and comparisons from others.

Within this environment, we demonstrate that QPEP achieves its design goals. It nearly halves average page load times compared to traditional VPNs and substantially improves on the performance of even unencrypted PEP applications. Additional simulations are conducted to assess QPEP's performance under various network conditions and in the presence of modifications to the QUIC standard. Ultimately, we find that

Network and Distributed Systems Security (NDSS) Symposium 2021
21-25 February 2021, Virtual
ISBN 1-891562-66-5
https://dx.doi.org/10.14722/ndss.2021.24074
www.ndss-symposium.org

QPEP represents a promising solution for performant over-the-air encryption in satellite networks while avoiding investment in new infrastructure or alteration of existing protocols.

II. Motivation

The prevalence of security and performance trade-offs in modern satellite broadband networks may initially seem unintuitive. After all, the dangers of unencrypted wireless communications are well understood and have been robustly mitigated in systems ranging from home wifi to cellular communications. Today, the decision of a terrestrial wireless ISP to offer unencrypted broadband services could, not unreasonably, be attributed to ignorance or incompetence.

This is not the case for long-range satellite communications. Severe security and privacy issues arising from the use of unencrypted broadband services from Geostationary Earth Orbit (GEO) have been known since at least 2005 [2]. However, our own experimental studies have found that, fifteen years later, tens of thousands of satellite customers still rely on unencrypted GEO links [3]–[5]. Deeply sensitive data is readily observed by eavesdroppers with access to simple home-television equipment - affecting customers ranging from individual home internet subscribers to massive corporations.

In the process of responsibly disclosing these vulnerabilities both to satellite ISPs and to their customers, our initial advice was simply to employ proven existing encryption techniques (e.g. IPSec). We quickly learned that this solution was somewhat naive and overlooked significant technical and cultural barriers unique to the satellite context.

In response to our disclosure efforts, satellite ISPs would often espouse the opinion that encryption was a duty which fell to individual customers. They were generally uninterested in deploying costly network-wide security protections, or already offered them, but only as a premium service add-on for particularly risk adverse clients (e.g. military customers). Occasionally, they would emphasize the importance of access to clear-text traffic headers in order to optimize network performance - increasing customer satisfaction.

When speaking with customers, they would acknowledge the value of encryption in the abstract, but were unwilling to accept substantial performance reductions caused by the use of end-to-end encryption tools such as VPNs. In some cases, they had already attempted to deploy VPNs but ended up removing them at the suggestion of their ISP to resolve these performance issues. Indeed, the support pages of many satellite ISP websites suggest the disabling of VPN software as a remediation for slow internet services [6], [7]. The most information-security conscious customers we contacted had attempted to employ piecemeal application-layer protections, such as replacing HTTP web-servers with HTTPS services. However, we found significant gaps in these defenses such as unencrypted DNS traffic or sensitive data from overlooked systems (e.g. a legacy FTP server or POP3 email service).

The principal motivation of this research is to present an approach which considers the unique technical and commercial requirements of these stakeholders. We design and implement a tool which empowers individual customers to encrypt the entirety of their satellite network connection by default, while maintaining performance that is on-par with, or better than, the unencrypted services they use today. Critically, our design requires no network changes or satellite ISP involvement.

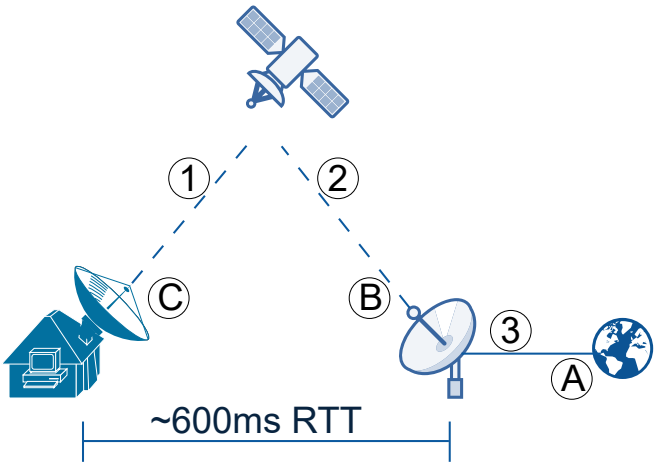


Fig. 1. Notional Overview of a GEO Network. A typical web request would travel from the customer to the satellite (Step 1) before being redirected down to the ISP's ground-station equipment (Step 2). From there it would be routed as normal IP traffic to the internet (Step 3). This process then occurs in reverse traveling back from the internet to the ISP's ground-station (Step A), up to GEO orbit (Step B), and down to the customer's dish (Step C). It's worth noting that the forward link signal (Step C) is a typically sent on a wide beam, with footprints measuring on the order of millions of square kilometers.

In addition to proposing this new system, the paper delves into many of the performance characteristics of our implementation. Our motivation in doing so is to demonstrate that our approach offers meaningful performance benefits over traditional VPN tools. Beyond this core hypothesis, significant additional detail is provided to facilitate replicability and comparative benchmarking. We provide open-source implementations not just of our tool, but also for each of the experiments run in the paper. This is because we recognize that, while our approach is a substantial and needed improvement, it is unlikely the only (or best) way to secure these networks. A key secondary motivation is thus to provide a framework and starting point for others interested in this topic area.

III. Background and Related Work

Understanding the security/performance trade-off requires a closer look at TCP behavior over satellite. This section provides an overview of our threat model, key defensive challenges, and prior work to address them.

A. The Eavesdropping Threat Model

Our focus is on broadband provided from platforms in geostationary earth orbit (GEO). The basic operation of GEO broadband can be thought of as a "bent pipe" (see Figure 1). As GEO is located more than 30,000 km away from the Earth's surface, a single satellite has line of sight to a vast area on the surface (theoretically as much as 40% of the Earth's surface, but practically closer to 20% for broadband communications). This has the advantage of making GEO broadband a relatively inexpensive mechanism of providing global service. Only a

half-dozen satellites are needed for almost complete Earth coverage (barring some polar areas).

An eavesdropping attacker is greatly aided by these coverage characteristics as emissions from GEO satellites are not targeted towards specific users. As a result, the radio waves reaching an attacker's antenna could be carrying traffic intended for an entire continent of satellite customers. Since these are consumer-oriented networks, the equipment necessary for eavesdropping on such signals is inexpensive and widely available [3], [4], [8]. With the rise of software defined radios, even more complex protocols are within the reach of relatively unsophisticated attackers [4], [9].

In light of this threat, it is not intuitively clear why status quo satellite broadband services fail to encrypt customer traffic. The main barrier is physical. Speed of light delays over the 30,000 km hop to GEO are substantial and round-trip latency can exceed 600 ms. Latency can be reduced with closer satellites in low Earth orbit (LEO) but this increases costs and complexity. While LEO offers as little as 50 ms in speed-of-light latency, satellites only maintain line of sight for a matter of minutes. Consistent global coverage thus requires hundreds of satellites. Status quo LEO constellations can still experience round-trip delays of up to 1,500 ms depending on the route a message travels [10]. Thus, while in-development constellations have made ambitious claims, satellite latency will likely remain relevant for some time [11], [12].

B. TCP Performance Over Satellite

To understand how latency discourages encryption, one must consider its impact on TCP performance. In this paper, we focus on standard TCP implementations on the assumption that forcing satellite customers to use alternatives (e.g. TCP-Hybla) is infeasible [13]. We outline two of the most prominent issues here but many others have been extensively characterized in prior work [14]–[17].

1) Barriers to TCP in Satellite Networks: The first challenge to TCP performance in satellite networks arises from the requirement that TCP data packets are responded to with an acknowledgment (ACK) message [17]. The effect is compounded by the three-way handshake which, in the best case, takes upwards of 1,500 ms to complete over GEO. When visiting a website with embedded images and related files, many three-way handshakes may be required - compounding delays. Although modern implementations may employ various optimizations to bundle or reduce the total number of ACKs, these are not tailored for satellite networks [14]. Further, in some legacy devices, ACKs may be elicited by every packet, greatly increasing perceived latency [18], [19].

The second challenge arises from TCP congestion control and TCP "slow-start" initialization [14]. TCP slow-start gradually increases the ratio of data segments to ACKs until a desired congestion window is reached. The time this process takes is thus a function of round-trip times (RTT) over the satellite link. Even once a connection has reached optimal window size, packet loss can be misidentified as a sign of congestion and cause the slow-start sequence to restart. While modern satellites are more reliable than in the past, packet loss is still common compared to terrestrial networks. As a result, TCP sessions are both slow to maximize their bandwidth usage, and, once maximized, struggle to maintain that state.

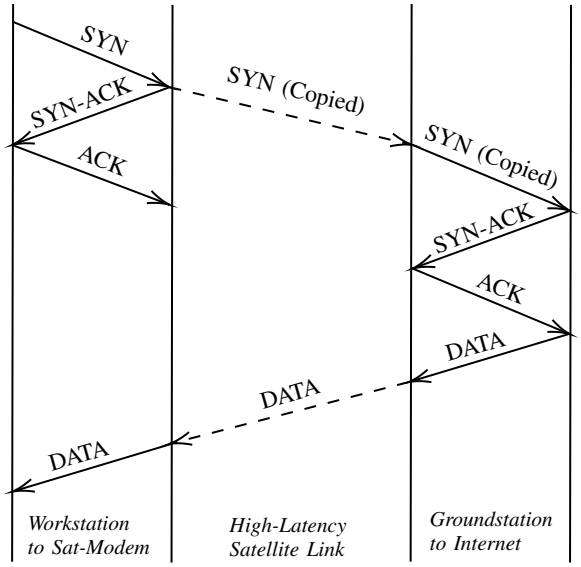


Fig. 2. Split Distributed PEP Handshake Example

These are but two factors among dozens, ranging from specific TCP option implementations to congestion control implications of link asymmetry [17]. Satellite network designs create a uniquely hostile environment for TCP.

2) PEPs: The most common approach to optimizing TCP traffic over satellite environments is the use of a class of appliances called "Performance Enhancing Proxies" or PEPs and loosely described in IETF RFC 3135 [20]. PEPs differ substantially and many implementations are proprietary and inaccessible to researchers. However, IETF RFC 3153 outlines a few basic principles that apply to most PEPs.

There are two typical PEP deployment options - integrated or distributed [21]. In integrated PEPs, a PEP appliance operates on a single endpoint - typically the ISP satellite gateway between the satellite network and the internet. In distributed PEPs, a PEP appliance operates on multiple endpoints - typically the customer satellite modem and the ISP gateway.

In either deployment, the PEP intercepts TCP traffic and applies optimizations in order to compensate for satellite performance issues. Typically, PEPs do this in a manner which is invisible to conversation endpoints so that no modifications are required on consumer hardware. This is referred to as a "transparent" PEP [20]. However, the concept of transparency is somewhat misleading as, in many cases, PEP modifications are still detectable (e.g. altered TCP sequence numbers).

Beyond this, PEPs vary quite broadly. Modifications made to TCP packets are often proprietary and implementation-specific. One common approach is to "split" incoming TCP connections prior to transmission across the satellite link and issue local ACK messages immediately for received TCP packets [20]. This allows three-way handshakes and congestion control to be negotiated locally before the satellite hop but requires the PEP developer to handle errors across the split.

In distributed PEPs, this splitting approach is extended

to create a tunnel between the individual PEP installations (see Figure 2). A TCP packet arriving at the client-side PEP (e.g. on the home satellite modem), is terminated locally as a TCP connection, and the payload is then forwarded through GEO using a modified TCP protocol (e.g. TCP-Hybla) or an alternative [13]. At the ISP gateway, a second PEP receives this modified packet, converts it back to normal traffic, and sends it along a locally-managed TCP connection to the internet.

Other PEP strategies can range from modifying TCP congestion control to bundling related packets into single transmissions. Commercial implementations often offer higher-level features such as inspecting HTTP payloads and combining requests for web-pages with their associated content [22]. A substantial body of existing work on PEPs covers these optimizations in detail not only for satellites, but also other latency sensitive environments (e.g. cellular networks) [14]–[16].

3) Security Consequences: PEPs have become a vital component of satellite broadband and customers have come to expect the performance characteristics of PEP-accelerated networks. This has created unintended tension between broadband performance and security.

As noted in RFC 3135, PEPs break the end-to-end semantics of IP connections [20]. Specifically, they require that the PEP appliance transparently modify packets - essentially acting as a benevolent man-in-the-middle on all TCP connections. This creates inherent compatibility issues with most commonly used VPNs as the PEP is unable to "snoop" into the VPN traffic flow and identify ACK messages. Even VPNs which leverage TCP for the transport layer are not correctly accelerated as ACK messages within the encapsulated connection are indistinguishable from other traffic. While most VPNs will function over PEPs, and in the case of TCP VPNs, observe modest improvements in initializing VPN sessions, functional browsing performance is roughly the same as if no PEP was deployed. The type of VPN employed may have marginal performance effects (e.g. UDP tunnels may receive better prioritization from the satellite ISP) but from a PEP-compatibility perspective, any VPN which does not leak the full TCP headers of a customer's connection faces the same issues. As a result, end consumers are faced with a choice between the security of VPNs and the performance of PEPs.

C. Existing Security Approaches

In this section, we will briefly consider some of the more consequential approaches proposed in academia and industry to enable satellite broadband encryption at each layer of the TCP/IP protocol stack. This analysis better characterizes how, despite a long history of research, PEP-compatible security remains unsolved in practice.

1) Physical and Link-Layer Approaches: Many techniques for over-the-air encryption focus on the lower layers of the networking stack - before TCP/IP becomes relevant. For example, physical-layer techniques such as frequency hopping patterns derived from cryptographic keys or direct sequence spread spectrum (DSSS) have been suggested as a mechanism for securing the entire satellite link [23]. Likewise the injection of artificial noise as an alternative to key-based encryption has been proposed [24]. These schemes tend to focus on military systems as they often necessitate expensive modifications to hardware that would be commercially unpalatable.

At the link layer, proposals still incur hardware costs but costs are often more manageable and restricted to hardware-based decapsulation. For example, the Consultative Committee for Space Data Systems (CCSDS) has proposed Space Data Link Security (SDLS), a protocol with built-in encryption for telemetry commands to scientific space missions [25]. Likewise, the proprietary Common Scrambling Algorithm (CSA) has long been used to restrict broadcast access to paying satellite television subscribers using smart-cards, albeit with notable security weaknesses [26].

One challenge for link and physical layer encryption systems like these is the multi-user environment. As it is rarely economically feasible to allocate each customer a unique satellite channel, customers with the same ISP will generally have a key which allows them to receive traffic from other broadcast subscribers. In such systems, the customer modem will determine which packets are relevant on the basis of header information and drop other traffic from the multiplex streams. An additional challenge with these systems it the process of key distribution and revocation over the broadcast medium.

2) Network and Transport-Layer Approaches: To provide over-the-air encryption with per-customer keys, a number of network-layer techniques have been proposed. In contrast with lower level approaches, interactions with TCP PEPs must now be considered directly. Many replicate traditional VPN software with bespoke modifications - for example by creating a modified IPSec with special encapsulating headers visible to PEP appliances [27], [28]. Proprietary "satellite VPNs" also exist which, while public information on their design is limited, are likely similar in design [29], [30]. Beyond concerns arising from proprietary encryption schemes, these non-standard layers can increase operator costs by limiting compatibility with existing networking equipment.

It makes intuitive sense to incorporate encryption within PEP appliances themselves - straddling the network and transport layers. This may be achieved by, for example, implementing an encrypted protocol over the satellite hop in a distributed PEP system. The transmitting PEP would first modify the TCP packets, then encrypt them. The receiving PEP would subsequently decrypt the received packets and forward them along the internet as normal. Many real-world PEP encryption products appear to employ this approach [31]–[33].

Most, if not all, encrypted PEPs are proprietary and not sold direct-to-consumer, making security claims difficult to verify. However, purported leaked manufacturer documents allude to built-in law-enforcement/intelligence back-doors in prominent examples [34]. Our own analysis of one satellite router with a pre-installed proprietary PEP found numerous cryptographic shortcomings, such as Diffie-Hellman implementations which are susceptible to man-in-the-middle attacks and key/IV reuse that permits replay attacks. Similar vulnerabilities have been alluded to in prior research [35]. Generally, the costs of adopting an encrypted PEP are undertaken by ISPs who may not perceive such purchases as value-for-money.

3) Application-Layer Approaches: An alternative approach would be the use of protocols which operate over the PEP-

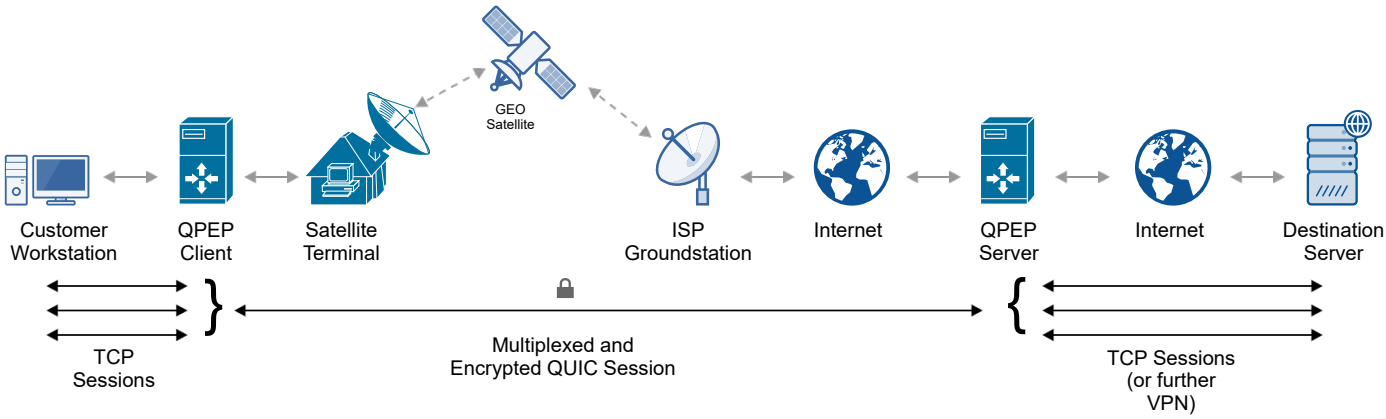


Fig. 3. Simplified Overview of QPEP Architecture. A traditional "splitting" TCP PEP is combined with a tailored over-the-air QUIC tunnel which offers encryption and further performance benefits compared to TCP. This architecture can be employed by either individual customers in the form of software on their computers and enterprise routers or by ISPs in the form of ground-station and modem software.

accelerated TCP connection. The widespread use of TLS encryption for websites, for example, has the effect of encrypting customer data over-the-air. However, this still leaks potentially sensitive data (such as the IPs a customer visits) over the massive radio-eavesdropping footprint of a satellite signal. Moreover, real-world observations of modern satellite traffic (see III-A) have found that, while customers could, in theory, use TLS, many do not. While this decision is the customer's, satellite ISPs may nevertheless have a duty of care.

Another application-layer approach is tunneling traffic into an encrypted TCP stream and issuing local ACK messages before the data egresses from the client's computer. This differs from most SSL-VPNs which do not spoof the connection endpoint. Some commercial products appear to implement this approach [36]. However, the requirement of software installed on the client's computer limits compatibility with embedded devices and creates friction.

IV. QPEP Design Considerations

To address some of these shortcomings we have developed QPEP, an open-source and non-proprietary tool which can be used by both individuals and ISPs to both encrypt and accelerate satellite TCP traffic[1]. At its core, QPEP follows a distributed "snooping" PEP model similar to the methods described in Section III-C2. The QPEP client tunnels TCP traffic over the satellite link inside a stream that leverages the encrypted QUIC transport protocol. Tunneled traffic is decapsulated by a receiving QPEP server which then routes the decapsulated traffic over the internet as if it were the client. A high level overview of this architecture appears in Figure 3.

A. QPEP Design Objectives

QPEP's principal objective is to protect against forward link eavesdropping attacks without suffering the same performance reductions are traditional VPN software. Much like end-to-end VPN encryption, QPEP expands the threat model to include both wireless eavesdroppers and the ISP, seeking to avoiding leaking meta-data and routing information to service providers. These objectives, and their contrast with status-quo techniques, are summarized in Table I.

The fundamental security design of QPEP is deliberately straightforward. For example, we use QUIC - a proven and popular standard - instead of designing our own scheme for authentication, key exchange, and session management over the satellite link. As mentioned in Section II, our goal with this research is to adequately resolve an urgent issue affecting real-world networks. If a suitable combination of existing and trusted methods can meet this need, arbitrarily complex variations serve little purpose beyond academic diversion.

While simplicity and novelty often feel mutually exclusive, that is not the case here. QPEP arises from the interaction of two seemingly unrelated design factors which have not been considered together in prior work. The first is the recognition that PEP applications are the traditional prerogative of satellite ISPs for cultural and commercial, rather than technical, reasons; there is no underlying reason that customers could not choose to "bring their own PEP" other than its redundancy with the service their ISP already offers. The second is that the TCP-tampering phase of a PEP's operation represents an ideal opportunity for modifying not just the performance of a connection, but also its security properties.

Together, these factors give rise to QPEP's unique hybrid between customer-oriented tunneling VPNs and ISP-oriented PEPs. This is a significant deviation from status quo approaches which treat the application archetypes as distinct.

This paper focuses on the design and evaluation of a specific implementation of this hybrid approach. However, the core system architecture is a contribution which is agnostic to the encryption schemes and transport protocols employed. Indeed, the open-source code for both QPEP and its simulation testbed is designed to facilitate the swapping out of individual engineering components by researchers.

[1]Source code and documentation for both our QPEP implementation and our OpenSAND-based testbed environment are available publicly (https://github.com/ssloxford/qpep). Example python scripts used to run all of the simulation scenarios presented in this paper are provided.

TABLE I. QPEP COMPARISON TO STATUS-QUO PEP AND SECURITY OPTIONS

	Accelerated TCP Connections	Private From Wireless Eavesdropper	Private From Satellite Service Provider	Intended For Customer Deployment	Intended For Service Provider Deployment	Open Source Implementations
Plain Connection	No	No	No	Yes (Default)	Yes (Default)	Yes (Default)
Traditional PEP (e.g. [37])	Yes	No	No	No	Yes	Yes (Rare/Unmaintained)
Traditional VPN (e.g. [38])	No	Yes	Yes	Yes	No	Yes
Secure PEP Product (e.g. [39])	Yes	Partially (IP Headers Leaked)	No	No	Yes	No
QPEP (this paper)	Yes	Yes	Yes	Yes	Yes	Yes

B. *Use of the QUIC Protocol*

The principal strategy employed by QPEP to support over-the-air security is the use of the QUIC protocol for tunneling traffic over the satellite link. While the QUIC standard is still evolving, it has already seen wide adoption in terrestrial networks due to its performance and security advantages over TCP. Several of these benefits make QUIC intuitively promising for secure PEPs.

1) QUIC Security Benefits: QUIC is an encrypted-by-default transport protocol. Unlike TCP, the session initialization process for QUIC incorporates a modified version of the TLS 1.3 handshake (see Figure 4). This means QPEP's QUIC tunnel can readily provide both encryption for encapsulated payloads and built-in end-point authentication. The addition of encryption mitigates many of the basic security concerns in status-quo satellite broadband networks and the presence of end-point authentication avoids the aforementioned man-in-the-middle issues with some proprietary encrypted PEPs.

Our decision to incorporate QUIC contrasts significantly with other tunneling based PEPs. In the status quo, the dominant approach, used by commercial PEPs such as Tellitec's "Enhanced TCP" (ETCP) product, is to implement a bespoke network/transport-layer protocol [31]. This limits compatibility with other network infrastructure (e.g. firewalls, switches, QoS appliances) and thus may require direct ISP participation in deployment. QUIC, on the other hand, is notionally a transport-layer protocol but in practice is implemented as an application on top of UDP. This allows for out-of-the-box compatibility with existing networks. An additional benefit to QUIC as opposed to, for example, TLS 1.3 secure channel tunnels, is that ISP PEPs largely ignore UDP traffic, limiting the risk of unexpected interaction with existing infrastructure[40].

In short, QUIC allows us to create a secure PEP which does not require ISPs to operate any decapsulation software on their gateway in order to ensure traffic compatibility with the wider internet. This means individual customers can use QPEP to secure traffic to a QPEP server on the open internet without ISP knowledge or involvement.

This ability for customers to protect their traffic between two arbitrary endpoints without trusting their ISP makes QPEP's security properties most comparable to prior satellite VPN protocol proposals [27], [28]. However, QPEP's design differs from these tools which still reveal limited portions of the TCP header to ISP PEPs for optimization (e.g. destination IP, port numbers, and TCP flags). An ISP snooping on a customer's QPEP traffic would only see the IP address and UDP port of the customer's upstream QPEP server. All information regarding the true TCP connections are hidden inside the QUIC tunnel. This means QPEP can function in the presence of other ISP-installed PEPs without any special adjustments. From a service-provider perspective, it is no different than any other UDP-based application. Of course, QPEP could also be installed by ISPs on customer modems and network gateways just like with traditional PEPs, but trust in ISPs is no longer a design requirement.

2) QUIC Performance Benefits: Beyond these security benefits, the use of QUIC offers notable performance advantages.

First, the initial QUIC connection can be negotiated in a single round-trip, substantially shorter than the TCP three-way handshake. When compared with alternative encrypted tunnel schemes - such as TLS-based VPNs - QUIC offers a substantial reduction in round-trip transfers (see Figure 4). Indeed, for previously known QUIC servers, it is possible for a client to begin transmitting data from the very first packet. While the TLS session initialization process is particularly ill-suited to satellite environments, few tunneling approaches can initialize secure channels with comparable RTT requirements to QUIC. For example, Internet Key Exchange (IKE) initialization commonly used in IPSec VPNs will generally require three or more round-trips depending on configuration and version.

Additionally, unlike TCP, QUIC does not require that all packets in a stream be processed in a particular order - removing head-of-line blocking issues and permitting heavy multiplexing. This allows QPEP to encapsulate multiple TCP flows inside a single QUIC session, reducing the number of session-initialization round-trips.

Like TCP, QUIC has built-in support for the retransmission of lost and corrupted packets. This obviates intuitive concerns relating to UDP usage over low-reliability satellite links. Moreover, some draft proposals suggest the addition of built in forward error correction (FEC). These efforts have largely stalled due to minimal terrestrial performance gains [41]. However, satellite environments may represent a context for reviving research on QUIC-FEC.

3) QUIC Satellite Performance: As QUIC is a relatively recent protocol, its use in satellite environments has not been subject to much research. What does exist is largely inconclusive. Some preliminary assessments have found that QUIC facilitates a 100% increase in satellite broadband page load times compared to PEP-accelerated TCP [42]. However, others suggest that QUIC performs better, measuring up to a 50% decrease page load times [43]. Preliminary IETF discussions have lead to a number of proposed (but unimplemented) techniques for optimizing QUIC over satellite [44].

Relevant research has focused on real-world connections

to HTTP2 web-servers which support the QUIC protocol. In these cases, researchers only control the client-side QUIC configuration. However, under QPEP's distributed model, it may be possible optimize both server and client QUIC implementations for the satellite link. Much as many modern PEPs use modified TCP implementations, QPEP's architecture lends itself naturally to bespoke optimization of QUIC parameters relating to FEC, ACK decimation and congestion control.

V. QPEP Implementation

Reaping the theoretical benefits of QUIC as a transport alternative for satellite TCP connections raises several important engineering considerations. In this section, we focus on the specific implementation and architecture of QPEP, how it merges properties of standard VPN applications and TCP PEPs, and some of the challenges in doing so.

A. System Architecture

QPEP is implemented according to a distributed PEP application architecture in order to ensure compatibility with all web services rather than only those with native QUIC support. This distributed design allows QPEP to operate transparently, converting TCP conversations into QUIC streams over the satellite hop and then back into TCP conversations terrestrially. The benefit is that no special software or configuration is required on individual customer devices. QPEP-encrypted traffic appears identical to normal TCP traffic. This differentiates it from application-layer commercial PEPs and ensures compatibility with IOT systems.

The practical implication of this architecture is that a QPEP deployment consists of two independent appliances (see Figure 3): a QPEP client on the customer side of the satellite link and a QPEP server on the internet side.

The client application can be installed as software directly on a customer's device. However, it is also designed to operate transparently if placed along the network path between a customer's device and the satellite modem. For example, an enterprise user or satellite ISP might install a QPEP client on a router within a local area network in order to encrypt and optimize internet-bound traffic from all connected devices.

The server application is similarly flexible. It can be installed by the satellite ISP on their gateway, like a traditional PEP. However, it can also be installed anywhere else on the internet, with encrypted QPEP traffic traversing the ISP gateway en-route to a cloud server or other egress point, as with a traditional VPN.

When the QPEP client launches, it opens a QUIC tunnel with an upstream QPEP server. Unlike in normal QUIC services, where idle sessions are short-lived, QPEP sets the timeout for this tunnel to a relatively long period of time (5 minutes of link inactivity). QPEP does this because QUIC session initialization requires a full round-trip over the satellite link, so by re-using recently established QUIC tunnels, QPEP can save round-trips over creating a new QUIC tunnel for each connection.

Each TCP connection which is managed by QPEP is assigned to its own unique QUIC stream within this QUIC tunnel. This allows QPEP to multiplex concurrent TCP connections and avoid creating redundant session initialization handshakes over the satellite link. As discussed Section V-B, this also allows for better congestion control as losses in each stream can be handled independently.

QPEP does not naively convert every incoming TCP packet to a QUIC packet. If it did so, we would expert performance akin to that of a traditional VPN. This is because the TCP three-way handshake would still occur over the latent satellite hop. Instead, QPEP must selectively terminate incoming TCP connections, drop spurious acknowledgements, and send only meaningful data across the satellite. This requires both the QPEP server and QPEP client to internally maintain state regarding each TCP connection.

When the QPEP client receives a TCP SYN packet, it immediately initiates a three-way handshake across the customer LAN - effectively "spoofing" the upstream destination TCP server. Upon finishing this handshake and receiving a TCP packet with payload data, it opens a new stream inside the QUIC tunnel session it established with the QPEP server at initialization. The client then strips away the TCP header information and encapsulates the payload data into a QUIC packet. A simple "QPEP header" consisting of a TCP four-tuple (*<src_ip, src_port, dst_ip, dst_port>*) is prepended to this packet. The client maintains a local state dictionary which maps the QUIC stream, this "QPEP header," and the associated TCP socket.

When the QPEP server receives an incoming QUIC payload, it checks its own state dictionary for any sessions associated with the received QPEP packet header. If no such entry is present, it opens a fresh TCP connection to the upstream TCP server on the basis of the received QPEP header and completes a three-way handshake across the internet - effectively "spoofing" the customer's device. It then updates its state dictionary to map this TCP session with the appropriate QPEP header and QUIC stream. From then on, each packet which the server receives in this QUIC stream will be converted into a TCP payload and then transmitted across the associated TCP socket to its destination. This same process happens in reverse for each response which comes from the internet, with the client extracting payloads sent by the server across the QUIC stream and then routing them to the appropriate TCP socket and onwards to the customer's device.

B. Error Handling and Session Management

The main challenge with this protocol splitting approach is correctly propagating errors which occur over one of the three network segments (*Customer* $\leftrightarrow$ *QPEP (client)*; *QPEP (client)* $\leftrightarrow$ *QPEP (server)*; *QPEP (server)* $\leftrightarrow$ *Internet*) to the others.

Over the satellite link, session management and congestion control is implemented within the QUIC session. This involves a modified version of the popular CUBIC congestion control algorithm for responding to losses which occur over the satellite hop. Congestion control is applied on a per-stream basis, preventing any individually troublesome stream from impacting the performance of other streams in the *QPEP (client)* $\leftrightarrow$ *QPEP (server)* session. While our implementation uses CUBIC for this purpose, only modest engi-

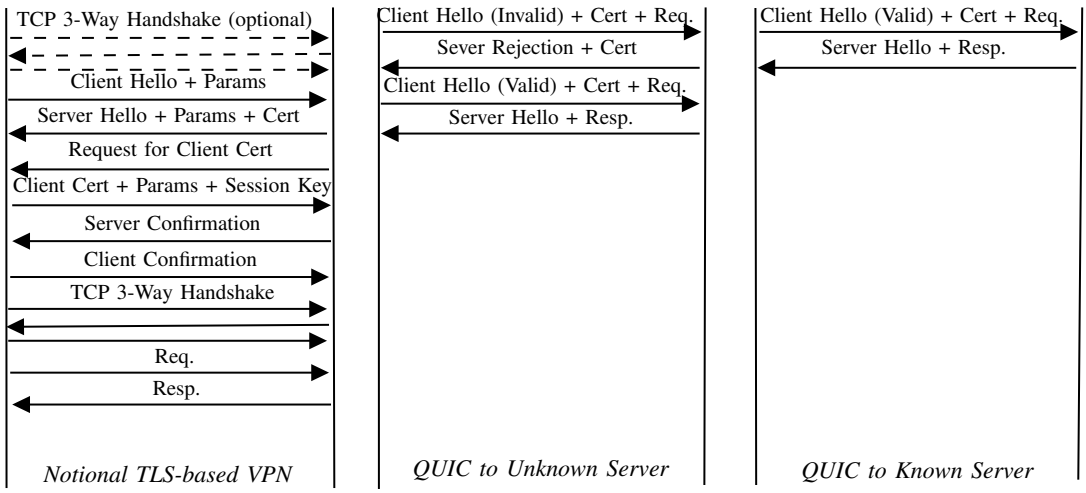


Fig. 4. Simplified Comparison of QUIC and VPN Initialization

neering effort would be required to replace it with any other QUIC-compatible algorithm. Adopting existing delay-tolerant algorithms (such as TCP-Hybla) to the QPEP architecture may thus represent one avenue for future research.

Over the terrestrial links, this process is handed down to the host's TCP stack. QPEP's "spoofed" endpoints resolve connection issues terrestrially just as any other TCP application. By design, this makes QPEP functionally transparent to users and ensures compatibility with upstream network appliances, such as firewalls or traditional VPN software.

The more difficult case is for errors which occur in one link and have implications for the others. For example, if a TCP connection on the *QPEP (server)* $\leftrightarrow$ *Internet* link fails, the error state of that TCP socket must be propagated up from the host's TCP/IP stack to the QPEP server application. The QPEP server will then issue a message to the QPEP client across the *QPEP (client)* $\leftrightarrow$ *QPEP (server)* segment designating the associated QUIC stream for closure. Upon receiving this message, the QPEP client will remove the stream from its session mapping dictionary and terminate the appropriate TCP connection on the *Customer* $\leftrightarrow$ *QPEP (client)* network. Finally, both the server and client will close the corresponding QUIC stream.

C. Limitations

In the implementation evaluated in this paper, QPEP only modifies TCP/IP connections. This is because our objective is to evaluate a secure alternative to traditionally unencrypted PEP appliances, which also focus exclusively on TCP connections. Only minor engineering modifications would be required to tunnel other protocols into the QUIC stream, such as UDP and ICMP. However, it is worth noting that we expect QPEP to have only marginal performance impacts on such protocols as they do not incur the same latency penalties as TCP over the satellite hop. Nevertheless, doing so may be desirable for end-users as it would bring over-the-air encryption by default to DNS queries and other non-TCP traffic.

It is also worth noting that we have not implemented QUIC's optional zero-round trip (0-RTT) session initialization handshake. While being able to further reduce the number of costly satellite round trips is an attractive prospect, prior work on QUIC's 0-RTT raises some security concerns with respect to replay attacks [45]. The potential harms of replay attacks are especially acute in the wide-footprint and high-latency context of satellite broadband. Indeed, a satellite eavesdropper may have closer physical proximity to a given QPEP server than the satellite ISP's gateway, which could allow them to even deliver "replay" messages faster than legitimate ones. Given that QPEP relies on long-lived QUIC sessions, the benefits of 0-RTT are likely marginal at best. This is because a QUIC handshake is only required for the very first connection a QPEP client makes, with subsequent streams re-using that session. Nevertheless, consideration of 0-RTT initialization dynamics may offer a route for some further optimization in future work.

D. Availability

An open-source reference implementation of QPEP, written in Go, is available in conjunction with this paper. Go was selected to increase accessibility without substantial performance sacrifices. To the best of our knowledge, only two non-proprietary PEPs exist [37], [46]. Both are implemented in C/C++, lack encryption capabilities, and have received only minimal development attention over the past several years. Other notable academic PEPs are either not publicly available or restricted to particular simulation tools [27], [47].

The QUIC implementation used by QPEP is based on the widely used quic-go library which roughly tracks the IETF QUIC proposal [48]. As discussed in Section VII-E, minor optional modifications to the QUIC implementation can be made to optimize performance in the satellite networking environment. Future work might also consider the suitability of other QUIC implementations such as Chromium's [49].

VI. Secure PEP Testbed

One challenge in developing and evaluating PEPs has been creating replicable simulations of system performance. While systems can be tested on live satellite networks, understanding performance under adverse conditions (e.g. poor reception

quality or network congestion) or creating experiments which others can verify often requires some degree of simulation. Simulating satellite IP networks involves more than the simple injection of artificial latency, which can result in misleading and inaccurate results [50].

The OpenSAND engine, previously Platine, is a long-standing satellite network simulation environment for more faithfully replicating satellite broadband [1]. The engine supports built in attenuation and modulation emulation, replicating conditions which can have significant implications for TCP performance. OpenSAND emulates satellite networks down to the link layer, simulating low-level protocol noise mitigations and creating realistic traffic routing behaviors.

However, the OpenSAND environment is somewhat difficult to configure - requiring multiple devices and precise network conditions. This has been noted in prior work as a barrier to its use, despite its relatively high degree of accuracy when validated against real-world networks [50].

In the process of assessing QPEP's performance, we have developed a simple dockerized deployment of the OpenSAND engine specifically tailored towards replicable PEP benchmarking. Our testbed models a basic GEO satellite network consisting of a single gateway and satellite terminal (akin to the networks shown in Figure 1 and Figure 3). This testbed is open-source and publicly available in the QPEP source repository (see Footnote 1). Its intention is to simplify the process for future researchers interested in making related contributions towards secure PEP development.

The testbed's gateway container is linked through the simulation's host machine to the broader internet, allowing a testbed user to open a web-browser and visit real websites as if they were using the simulated satellite link. We also connect the gateway container to a simulated LAN environment with a workstation containing several network benchmarking tools. A similar LAN environment is connected to the satellite terminal, replicating a satellite customer's devices.

On the satellite container, we include packet capture tools for real-time monitoring of simulated over-the-air transmissions. This allows for immediate verification that secure PEPs are not leaking sensitive data in clear-text.

Finally, pre-configured installations of QPEP, OpenVPN, and PEPsal are installed for both the gateway and satellite terminal networks. A set of example python scripts are provided to orchestrate the environment and run the experiments presented in this paper. These scripts are designed as modular benchmarks which can also be adapted to future secure PEP proposals to facilitate direct and replicable comparisons with QPEP in future work.

VII. QPEP Evaluation

In this section, we present an evaluation of the QPEP approach and its impact on the performance of TCP-based traffic within our testbed environment.[2]

[2]In this paper, QPEP is evaluated only through simulation. While this has benefits for reproducability, we originally intended to supplement these experiments with validation in a real-world VSAT network. Unfortunately, due to restrictions during the global coronavirus pandemic, this has not been possible. When real-world benchmarks are available, they will be added to QPEP's public source code repository (see Footnote 1).

No comparable encrypted satellite PEP is publicly available. As such, we selected PEPsal, one of the only open-source unencrypted PEPs, and OpenVPN, a popular VPN product without specific satellite optimizations, to provide some context to measurements made [37], [38]. Future work including commercial and proprietary PEPs might be of merit, although these are not readily available to researchers. It is worth noting that the particular VPN product (e.g. OpenVPN vs. PPTP vs. IPSec) is unlikely to have meaningful impact on performance benchmarks insomuch as all hide the true TCP headers of the customer's connection from ISP PEPs.

A. Experimental Setup

First, we consider preliminary results under ideal OpenSAND network conditions. Next, we present QPEP with various adverse network situations, assessing its performance in the presence of high rates of packet loss and under variable delay conditions such as those in LEO constellations. Finally, we briefly consider how performance modifications to the QUIC protocol itself may impact QPEP's behavior.

Unless otherwise noted, the OpenSAND network is configured to use the DVB-S2 protocol with GSE encapsulation for forward-link communications and DVB-RCS2 with RLE encapsulation for the return link. The clear-sky SNR is set to 20 dB and Adaptive Coding and Modulation (ACM) is used at the physical layer to provide quasi error free (QEF) communications at this SNR level. A constant speed-of-light delay of 125 ms is used from both the satellite terminal and the satellite gateway to the satellite (resulting in a 500 ms RTT). The forward-link carrier frequency is allocated 50.0 MHz of bandwidth with a roll-off factor of 0.25 and the return-link is allocated approximately 7.4 MHz of bandwidth. These bandwidth values are well within the simulation capabilities of the machines used to run the scenarios, reducing the risk of artificial network caps due to hardware limitations. While simulations were run on multiple hosts for efficiency reasons, comparisons made within a given experiment (e.g. all measurements shown in a single figure) were conducted on the same physical host.

Our configuration is intended to represent the characteristics of a typical GEO satellite broadband network. We also briefly touch on an alternative LEO network configuration in Section VII-C, which demonstrates the performance of QPEP in a situation where latency is variable, depending on the geographic location of the end user and the corresponding satellites and ground stations. The testbed supports arbitrary delay and bandwidth models which may be useful in future work considering more esoteric network designs, such as those involving space-to-space routing or polar orbits.

Simulations of QPEP are configured with a QPEP server sitting local to the satellite gateway network and listening for incoming QUIC tunnel connections. The QPEP client is hosted on the satellite terminal and listens transparently into all incoming TCP connections. The QPEP server is configured to accept up to 40,000 concurrent streams from a single host - substantially higher than quic-go's default of 100. This is to enable compatibility with concurrent download benchmarks.

OpenVPN simulations are deployed similarly to QPEP, with an OpenVPN client connected to the satellite terminal

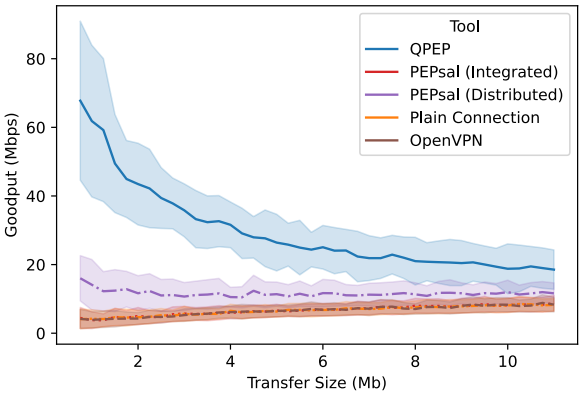


Fig. 5. Goodput Comparison by Iperf Transfer Size. The shaded zones represent standard deviation across 100 simulation runs at each file size. Note that QPEP performs well for small transfers and matches the performance of the unencrypted distributed PEP for larger transfers. Meanwhile, the traditional VPN, integrated PEP, and unencrypted satellite connection all perform relatively poorly throughout.

and an OpenVPN server connected to the satellite gateway. OpenVPN is configured to leverage a UDP tunnel as this is expected to perform better in the satellite environment.

PEPsal is evaluated under two different configurations - a distributed installation and an integrated installation. Evaluations of distributed PEPsal are implemented with a PEPsal endpoint transparently listening to all incoming TCP traffic on both the satellite gateway and the satellite terminal. In integrated PEPsal, a PEPsal endpoint listens to incoming TCP traffic on the satellite terminal but no endpoint is installed on the satellite gateway.

Diagrams summarizing these configurations can be found in Appendix A.

B. Baseline Performance

An initial comparative assessment of goodput can be made through the use of Iperf which attempts to provide consistent performance evaluations of network speed. For these benchmarks, an Iperf server is hosted on the satellite gateway network and is used to transfer data to an Iperf client connected to the satellite terminal. For each tool, one-hundred iterations of Iperf are run at data transfer sizes ranging from 0.5 to 10 MB in 250 KB intervals. Varying the volume of data transferred provides insights into the extent to which results are influenced by session initialization time. We would expect smaller transfers to demonstrate larger susceptibility to latent TCP handshakes as a proportion of total transfer time while larger transfers should be more heavily influenced by congestion control and total available bandwidth. The results of these experiments are summarized in Figure 5.

We see that QPEP is capable of making significantly greater use of bandwidth for small to moderate-sized downloads than any of the evaluated alternatives, even, surprisingly, the unencrypted PEPs. This make sense as QPEP is able to send data along with the stream initialization packets, allowing very small transfers to be completed in a single round-trip. As shown in Figure 5 this has a large effect on the measured goodput for small transfers, but diminishes at larger transfer sizes until QPEP approaches the performance of unencrypted distributed PEPs.

Integrated PEPsal offers little advantage here as it is constrained by head-of-line blocking over the satellite hop and the majority of download traffic originates on the un-optimized route from the gateway to the user. Distributed PEPsal performs much better as it is able to optimize both directions of the satellite conversation. However, it lacks QPEP's ability to encapsulate concurrent streams and to make use of the first few handshake packets for data delivery. Finally, as expected, OpenVPN performs much worse than QPEP, essentially matching, or slightly underperforming, an un-optimized satellite link.

This benchmark, while meaningful, is somewhat misleading. Iperf provides one important measure of goodput but the scenario it evaluates is not representative of real-world behavior. Specifically, opening a connection to a port, ramping it up to maximum speed, and then maintaining that speed for many file transfers is not how most web services operate. PEPs were explicitly invented to optimize web-browsing and visits to text and image-based services. Even if QPEP were well suited to encrypting certain types of file transfers, its adoption would likely hinge on its performance for web-browsing tasks.

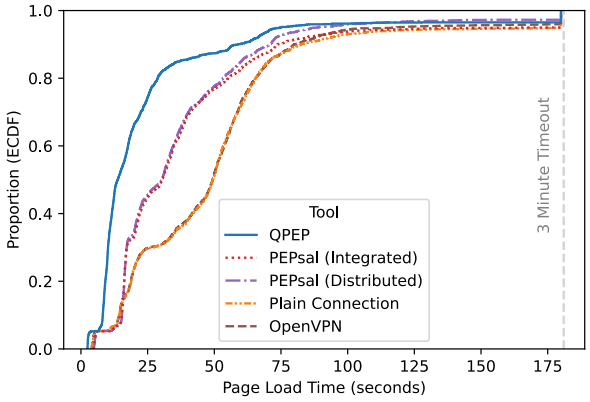


Fig. 6. ECDF Comparison of PLTs over Alexa Top 20. Note that QPEP shows significantly faster PLTs than traditional VPNs and marginally better PLTs compared to unencrypted PEPs. Each line represents 2,000 simulations with a connection timeout set to three minutes.

A more realistic sense can be found through the evaluation of the time it takes to visit actual websites. Unlike IPerf, web-browsing consists of the transfer of many small files (e.g. embedded images or style-sheets) over multiple TCP sessions. Often, these files can be hosted on a variety of servers. This makes web traffic more sensitive to latency effects.

Experimentally measuring page load times (PLTs) is an imprecise art. For our simulations, we used the open-source tool Browsertime [51]. Browsertime reports PLT as the number of elapsed milliseconds between the "navigationStart" and the "load" event of the browser's navigation timing API (as defined in [52]). This roughly translates to the amount of time between a user hitting the enter key in the browser's navigation bar and the moment when all page resources, including the DOM, images, and stylesheets, are loaded.

To conduct these experiments, we connected our simulated satellite gateway to a real terrestrial broadband network. This naturally induces measurement variability depending on network conditions at measurement time. To reduce this variability, we conducted 100 connections with each tool to each of the top 20 distinct domains listed by Alexa Internet Inc [53]. Between each visit, the browser (a headless version of Firefox) and the DNS cache were reset. Any page loads which took more than 3 minutes were terminated as timeouts. The results of these PLT measurements are summarized by means of an Empirical Cumulative Distribution Function (ECDF) in Figure 6.

This page load time comparison shows that QPEP is able to encrypt realistic web browsing traffic without undermining the performance users have come to expect from status quo unencrypted PEPs. QPEP's median page load time (PLT) across the Alexa Top 20 is 13.77 seconds. This is roughly 54% faster than distributed PEPsal's 30.16 seconds and integrated PEPsal's 30.5 seconds. It makes sense that both integrated and distributed PEPsal perform similarly here as PLTs are dominated by large numbers of client-initiated TCP handshakes for various web resources. In terms of mean PLTs, which are more heavily influenced by "worst-case" long-running connections, QPEP still significantly outperforms the traditional insecure PEP, with a mean PLT of approximately 25.80 seconds compared to distributed PEPsal's 37.61 second average and integrated PEPsal's 40.70 second average.

The most important benchmark comparison, however, is between QPEP and the status quo options for end-to-end web traffic encryption. In this case, we find that QPEP more than halves median PLTs when compared to OpenVPN's encryption, achieving 72% faster page loads than an OpenVPN-encapsulated connection's 49.42 second median PLT. In terms of mean PLTs, QPEP still roughly halves OpenVPN's mean PLT of 50.01 seconds. As expected, we further observe that OpenVPN roughly matches, or slightly under-performs, a basic unencrypted and unoptimized satellite link.

The relative disadvantage of using a traditional VPN for over-the-air encryption in GEO broadband is clear when considering this PLT metric. QPEP is functionally the same from a security perspective (eavesdroppers cannot interpret intercepted traffic), but significantly more performant by design. The surprising additional outcome that QPEP achieves significantly lower PLTs than established and architecturally similar insecure PEP appliances suggests that QUIC is particularly well-suited for the satellite tunneling use-case.

C. Performance Under Adverse Conditions

While these basic evaluations present a compelling case for the use of QPEP in a typical GEO environment, satellite networks can exhibit many atypical characteristics. Packet loss, rain-fade, and orbit altitudes can all have significant performance implications. As such, we have elected to evaluate the relative performance of QPEP under some of these conditions.

Intuitively, packet loss and rain fade conditions are significant threats to encrypted tunneling PEPs like QPEP. Loss of critical packets related to the key exchange process or session initialization could impose heavy additional round-trip costs not observed in clear-sky conditions. In a tunneling PEP, severe packet loss can even cause the tunnel between the PEP client and server to timeout or otherwise break. However, at mild loss levels, PEPs are expected to improve network performance by mitigating the impact of TCP congestion-control restarts as discussed in Section III-B.

Given these requirements, a series of simulations were run to assess QPEP's performance under adverse network conditions. For these experiments, losses are expressed in the form of "Packet Loss Rates" (PLR) between the satellite and the customer's satellite terminal. This represents the probability that any given DVB-S encapsulated packet is irrecoverably corrupted in transmission. Measuring "typical" PLRs in satellite networks is a deceptively complex task as definitions of both "packet" and "loss" are closely tied to the specific process by which IP transmissions are framed and fragmented by satellite ISP equipment [54]. A common worst-case upper bound often referenced in satellite broadband standards is 1×10^{-3}, but real world conditions run the gamut from "quasi error free" conditions (where nearly all packet errors are corrected by the DVB-S link layer) to rates upwards of 1×10^{-2} [55]–[57].

Satellite networks leverage a variety of techniques to mitigate packet losses. For example, DVB-S forward error correction (FEC) can resolve modest bit errors at the link layer, resulting in error-free IP-layer transmissions. Many modern networks now implement adaptive modulation and coding (ACM) schemes at the physical layer, allowing the network to intelligently trade-off bandwidth in favor of reliability under adverse conditions. In our experiments, these network specific particulars are abstracted away to IP packet loss rates, which allows for direct focus on the final performance implications of a problematic link. However, future work may benefit from considering other dimensions of satellite link resilience, especially bandwidth variations. That said, it is not obvious that these lower-layer noise responses, to the extent that they manifest in the same ultimate PLR, would have significant impact on the relative performance of the tools evaluated in our simulations.

In these adverse condition simulations, we first measured IPerf goodput at 28 PLR levels distributed logarithmically from 1×10^{-9} (packet loss is very rare) to 1 (all packets are lost). At each of these levels, 100 IPerf simulations were conducted to transfer a file measuring 2MB in size. Our results are summarized in Figure 7.

As expected, we find that QPEP suffers at higher rates of packet loss. This makes sense as QUIC was not designed with lossy links in mind and, in particular, the loss of key cryptographic handshake packets during initialization can impose substantial RTT penalties. That said, QPEP outperforms distributed PEPsal at modest levels of packet loss and would be well suited to networks with strong signal-to-noise ratio (SNR) or physical-layer mitigations against packet loss. Generally though, this initial Iperf metric, suggests that QUIC's cubic congestion control mechanism is not as robust to PLR as PEPsal's TCP-Hybla based approach. Future work which adapts TCP-Hybla to the QUIC protocol may prove one avenue to maintain QPEP's performance edge under such conditions.

It is worth noting that, regardless of PLR, QPEP consistently meets or exceeds the performance of OpenVPN as an encryption tool. This suggests that, from the perspective of a

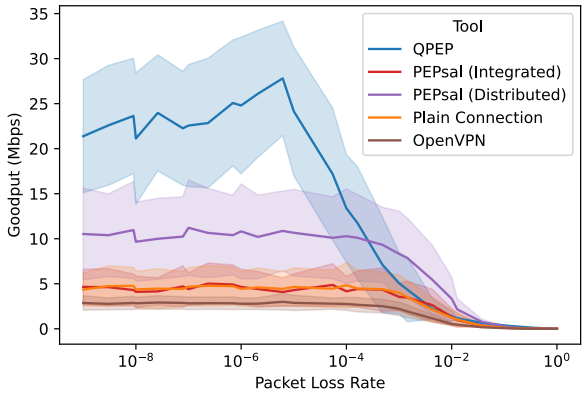


Fig. 7. IPerf Performance in Lossy Environments. The shaded intervals represent a standard deviation in measurements across 100 simulation runs for each PLR. Note that QPEP performance degrades rapidly in the presence of high PLRs, although it always meets or exceeds the performance of the only other encrypted tool (OpenVPN).

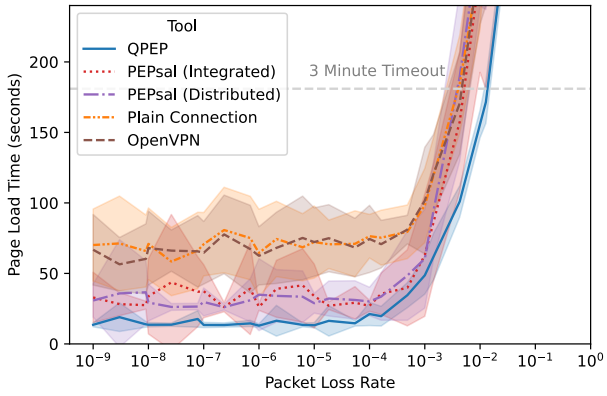


Fig. 8. Mean PLT of NASA.gov Homepage at Increasing PLRs. Lines which are lower and to the right demonstrate better PLTs at higher PLRs. Note that QPEP performs better here than in the IPerf case. This makes sense as connections are relatively short-lived and some errors may be resolved by the browser (e.g. by re-issuing failed requests).

security-conscious user, QPEP is net-beneficial compared to traditional VPN encryption.

Of course, as mentioned in Section VII-B, this Iperf benchmark only tells part of the performance story. In many cases, the short-lived data connections of web-browsing are likely more resilient to packet loss. To assess the impact of attenuation on page load times, a series of simulations were run measuring the average PLT of the NASA.gov homepage over fifty visits at each PLR interval (Figure 8).

Here, QPEP performs better, meeting or exceeding the performance of distributed PEPsal and substantially exceeding the performance of OpenVPN-based encryption throughout. This suggests that the goodput issues QPEP encounters at high PLRs may not necessarily translate to meaningful performance reduction for real web-browsing traffic, as QPEP's ability to rapidly deliver small images and text files over the latent satellite connection may counteract more error-prone delivery of larger transmissions.

In short, this preliminary look at packet loss effects suggests that QPEP is a better alternative than status quo VPN encryption under adverse conditions and performs reasonably well compared to insecure PEPs at low to moderate PLRs. However, our findings suggest that future work optimizing QUIC's response to packet loss could offer significant improvements, especially for file transfer operations.

D. Performance in LEO

While this paper has focused on GEO networks and performance under constant speed-of-light delays, some proposed "next-generation" satellite networks focus on the use of low earth orbit (LEO) to reduce transmission latency. While GEO broadband is likely to remain relevant for the foreseeable future due to its wide coverage and heavy industry adoption, it is worth considering QPEP's performance in future LEO systems as well. Unlike in GEO, latency from LEO can vary substantially due to the shifting relative locations of satellites and the geographic position of the customer. Additionally, as LEO is much closer to the Earth's surface (approximately 2,000 km), speed of light latency effects are reduced.

To emulate a LEO system, we implement an OpenSAND simulation model which replicates observed delay characteristics from a satellite terminal in the Atlantic Ocean connecting through the Iridium LEO constellation to a gateway in London [58]. In this particular network, one-way delay varies from as low as 25 ms to as high as 140 ms, depending on the time of transmission and the route a packet must take through the constellation. The same PLT benchmark from section VII-B was repeated in this environment. The results of these experiments can be found in Figure 9.

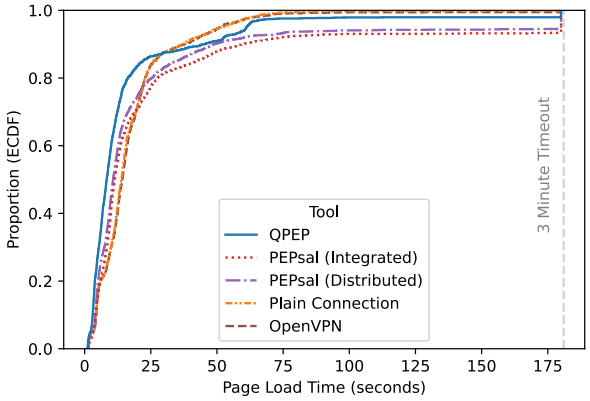


Fig. 9. ECDF of Alexa Top 20 PLTs in Iridium Simulation. As expected, QPEP offers very little benefit in this lower-latency environment. However, it also imposes smaller overheads than traditional unencrypted PEPs.

As expected, the performance benefits of PEPs are much less pronounced in LEO networks and VPNs represent a more viable encryption option. QPEP still generally outperforms OpenVPN in this context, with a median PLT of 8.3 seconds compared to OpenVPN's 14.2 seconds. However, this is counteracted by the fact that all three PEPs seem to struggle with more complex/slow-loading pages where the added overhead

of connection splitting is not always worth the benefits. It is worth noting, however, that relative to both PEPsal architectures, QPEP does appear to impose less overhead costs. Taken together, these measurements suggest that QPEP would be an adequate mechanism for providing encryption in LEO constellations but, unlike in GEO networks, the performance gains over more established VPN options are, at best, marginal.

E. QUIC Optimizations

One of the principal theoretical advantages of a distributed PEP configuration is the ability to adopt non-standard and environmentally tailored protocols over the satellite hop. In this section, we consider a demonstrative example as to how such optimizations might be identified and incorporated into QPEP.

One common strategy for improving the performance of TCP over satellite links is ACK decimation - the process of combining many ACK messages into a single transmission at regular intervals. Unlike TCP, the QUIC protocol is not ACK-clocked which diminishes the impact of ACK decimation on goodput [59]. Nevertheless, QUIC leverages ACKs for loss detection and QUIC ACK messages are relatively large compared to in TCP contexts. This means that excessive acknowledgments can potentially congest asymmetric links [60].

We conducted a set of initial experiments to determine if QUIC ACK decimation ratios had any impact on QPEP's measured goodput. In the default QUIC implementation, this ratio is set to 2 ACK eliciting packets per ACK for the first 100 packets, and 10:1 thereafter. Due to long satellite RTT's however, we observed in practice that the vast majority of ACKs were triggered by the QUIC implementation's default 25 ms ACK timeout window rather than decimation. In order to measure the effect of decimation in isolation, this timeout window was increased substantially to 8,000 ms and ACK decimation was set to begin after the 4th packet over the QUIC link. As a result, these experiments are not directly comparable to those which appear elsewhere in the paper. We further selected the IPerf benchmark as, based on Section VII-C, it is more sensitive to packet loss effects that are directly relevant to ACK decimation.

In these experiments, 100 IPerf benchmarks were conducted for 5 Mb transfers at each of 30 decimation ratios. These ranged from 1:1 to 30:1. Additionally, we conducted the evaluations at three different PLRs (error-free, 1×10^{-6}, and 1×10^{-4}). The results are summarized in Figure 10.

We observe a few relevant trends in these results. The first is that extremely low ACK decimation rations (e.g. 1:2) perform poorly. This makes sense as large portions of bandwidth are tied up with ACK messages at these levels. Higher ratios offer some benefit, with the default ratio of 10:1 roughly doubling goodput compared to "worst-case" 1:1 ratio. However, the benefit of ACK decimation is more limited and less consistent in lossier environments, as denoted by the relatively small goodput increase and high variance observed in our experiments. Finally, we observe that at a certain point, additional decimation has little to no effect. This may be the result of timeouts again gaining dominance. We found that, in practice, increasing the minimum ACK timeout much beyond the value used in our experiments (8 seconds), led to link

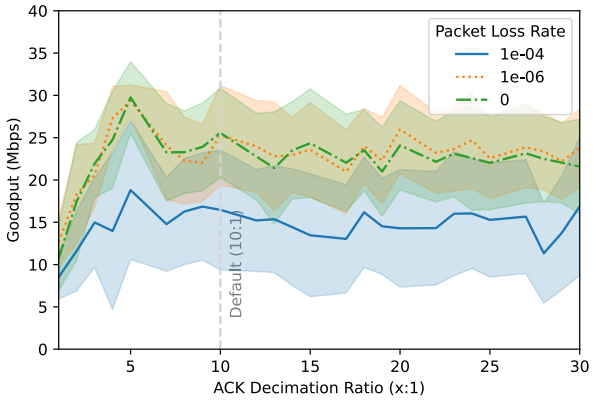


Fig. 10. IPerf Goodput vs ACK Decimation Ratio

instability. Specifically, unrecognized packet losses caused the IPerf client to perceive its connection to the server broken, causing it to terminate prematurely.

The high variance of these preliminary experimental results makes it difficult to definitively pinpoint an optimal ACK decimation ratio. However, one clear takeaway is that increasing the minimum ACK timeout period from 25 ms allows for better exploitation of the QUIC's ACK decimation feature in the presence of high-latency networks. We observed a roughly 25% increase in mean clear-sky goodput (from 19.25 Mbps to 25 Mbs) as a result of doing so, even when the ACK decimation ratio itself remained at its default of 10:1. Finding an ideal ACK decimation radio for the satellite use-case, and potentially setting it dynamically in response to noise and traffic characteristics, represents a possible avenue for further performance tuning in future work.

The ACK decimation ratio is but one of many QUIC protocol constants which may be tuned to have a meaningful impact on proxy performance. Changes in congestion window parameters, congestion control algorithms, session timeouts, and multiplexing limits may all also represent avenues for further tuning. The search space for such an optimization problem is enormous and exceeds the remit of this paper - especially given that default QUIC implementations already offer substantial security and performance benefits over the status quo. Nevertheless, the approach presented through this case study demonstrates how the testbed environment and benchmarks we developed for evaluating QPEP might be leveraged more broadly for protocol performance research.

VIII. Future Work

The QPEP implementation presented here is a proof-of-concept and productive use would benefit from additional features. In Section V-C we outline a few intuitive starting points, such as support for non-TCP protocols and the implementation of 0-RTT session initialization which is robust to replay attacks. Beyond 0-RTT, other QUIC feature proposals, such as forward error correction or alternative congestion control protocols to CUBIC, may provide routes for additional performance gains.

Under our threat model, ISPs are considered completely untrusted. This means that QPEP conceals the nature of customer traffic from ISP Quality of Service optimizations. Adding additional header layers which communicate QoS relevant metadata to ISPs, while preserving privacy, may further facilitate ISP-level integration of QPEP into customer routers.

As mentioned in Section II, the principal objective of this research was to develop an encryption tool which could be used to protect TCP traffic by default in satellite networks without meaningful reductions in performance. Even default QUIC implementations meet or exceed this baseline requirement without any modification. However, future work which considers the significant but surmountable engineering challenge of optimizing the performance of a QUIC tunnel over satellite represents a logical next step.

Beyond the design of secure PEPs, the testbed presented here may be useful for more general investigations of QUIC performance over satellite. Thus, although unlikely in near-term SATCOMs environments, if TCP ends up being phased out in favor of QUIC or if "TLS-everywhere" transitions from aspiration to reality, our contributions may be of enduring use for general performance research.

IX. Conclusion

In this research, we have challenged the historical assumption that security and performance must trade off in high-latency satellite networks. The result of this assumption has been that tens of thousands of satellite customers, from individuals to corporations, continue leak sensitive data to potential wireless eavesdroppers in the status quo. By delving into the underlying causes of inadequate encryption from GEO, we isolated key physical and commercial dynamics which have prevented the adoption of terrestrial encryption tools to the SATCOM domain.

We have presented a new approach to encrypting TCP satellite communications over-the-air through the use of QPEP - a PEP/VPN hybrid which leverages the open QUIC protocol standard to provide an encrypted UDP tunnel for the satellite hop. QPEP is evaluated through replicable simulations in an open-source benchmarking test suite we developed. These tests allow for direct comparisons between PEP and satellite encryption techniques and for targeted adjustments to various physical conditions.

Through these simulations, we find that QPEP is able to provide satellite users with over-the-air encryption while reducing page load times (PLTs) by more than 70% compared to status-quo VPNs. Moreover, we find that the use of QPEP is unlikely to result in TCP performance reductions for users who already employ insecure PEP products. Indeed, under certain network conditions, we found that QPEP may offer up to 50% performance improvement over such tools while also offering significant security benefits. We present the case that future work might expect to further bolster these gains and provide demonstrative case studies of two underlying QUIC protocol implementation characteristics as a starting point.

QPEP is the first open source PEP with support for the encryption of arbitrary TCP traffic. Moreover, unlike many commercial offerings, QPEP is fully independent, allowing individuals to run their own QPEP servers without sharing sensitive metadata with ISPs or convincing their ISPs to implement costly modifications to their existing network infrastructure. This offers an actionable near-term solution for customers interested in protecting their privacy. In the longer-term, QPEP's architecture is also suited to ISP deployment on modem equipment, allowing it to serve as a drop-in replacement for proprietary TCP PEPs. QPEP is entirely software-based and compatible with existing networking equipment and protocols. This means the practical costs for a customer implementing QPEP are on the same scale as any with any other open-source VPN. That is to say, the main implementation costs are those of renting a cloud host to run a QPEP server and paying for the desired amount of bandwidth to connect that server to the internet. This contrasts substantially with many existing PEP implementations which are implemented as physical "black-box" devices along the network path.

As the next generation of satellite broadband launches, ensuring the privacy of TCP communications without sacrificing performance is more important than ever. The QPEP proof-of-concept presented here demonstrates how careful consideration of the unique physical dynamics of outer space can leverage open and verifiable standards to meet this need.

References

[1] Viveris Technologies, *OpenSAND*, https://forge.net4sat.org/opensand/opensand, Apr. 2019.

[2] A. Adelsbach and U. Greveler, "Satellite communication without privacy–attacker's paradise," *Sicherheit 2005, Sicherheit–Schutz und Zuverlässigkeit*, 2005, ISSN: 3885793911.

[3] J. Pavur, D. Moser, V. Lenders, and I. Martinovic, "Secrets in the Sky: On Privacy and Infrastructure Security in DVB-S Satellite Broadband," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '19, New York, NY, USA: ACM, 2019, pp. 277–284, ISBN: 978-1-4503-6726-4. DOI: 10.1145/3317549.3323418.

[4] J. Pavur, D. Moser, M. Strohmeier, V. Lenders, and I. Martinovic, "A Tale of Sea and Sky: On the Security of Maritime VSAT Communications," in *To Appear in 2020 IEEE Symposium on Security and Privacy (S&P)*, Oakland, CA: IEEE, May 2020.

[5] J. Pavur, *Whispers Among the Stars: A Practical Look at Perpetrating (and Preventing) Satellite Eavesdropping Attacks*, https://www.blackhat.com/us-20/briefings/schedule/index.html\#whispers-among-the-stars-a-practical-look-at-perpetrating-and-preventing-satellite-eavesdropping-attacks-19391, Conference Briefing, Aug. 2020.

[6] Hughes, *What Should I Do if I have Slow or No Connectivity?* https://support.hughesnet.com/en/faq/internet/slow-no-connectivity.

[7] Viasat, *Answers to your Questions relating to your Viasat Internet service during this difficult time*, https://help.viasat.com/internet/articles/Denver_FAQ/Answers-to-your-Questions-relating-to-your-Viasat-Internet-service-during-this-difficult-time, 2020.

[8] A. Laurie, *$Atellite Hacking for Fun & Pr0fit!* http://www.blackhat.com/presentations/bh-dc-09/Laurie/BlackHat-DC-09-Laurie-Satellite-Hacking.pdf, 2009.

[9] C. Moore, "Spread Spectrum Satcom Hacking Attacking the Globalstar Simplex Data Service," *DEF CON*, vol. 23, 2015.

[10] M. M. McMahon and R. Rathburn, "Measuring latency in Iridium satellite constellation data services," NAVAL ACADEMY ANNAPOLIS MD DEPT OF COMPUTER SCIENCE, Tech. Rep., 2005.

[11] SpaceX, *Starlink Mission*, en, https://www.spacex.com/news/2020/01/07/starlink-mission, Text, Jan. 2020.

[12] OneWeb, *OneWeb*, http://oneweb.world/.

[13] C. Caini and R. Firrincieli, "TCP Hybla: A TCP enhancement for heterogeneous networks," en, *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, pp. 547–566, 2004, ISSN: 1542-0981. DOI: 10.1002/sat.799.

[14] S. Oueslati-Boulahia, A. Serhrouchni, S. Tohmé, S. Baier, and M. Berrada, "TCP over satellite links: Problems and solutions," *Telecommunication Systems*, vol. 13, no. 2, pp. 199–212, Jul. 2000, ISSN: 1572-9451. DOI: 10.1023/A:1019192022690.

[15] C. Caini, P. Cornice, R. Firrincieli, M. Livini, and D. Lacamera, "TCP, PEP and DTN performance on disruptive satellite channels," in *2009 International Workshop on Satellite and Space Communications*, Sep. 2009, pp. 371–375. DOI: 10.1109/IWSSC.2009.5286336.

[16] I. Bisio, M. Marchese, and M. Mongelli, "Performance Enhanced Proxy Solutions for Satellite Networks: State of the Art, Protocol Stack and Possible Interfaces," en, in *Personal Satellite Services*, K. Sithamparanathan and M. Marchese, Eds., ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Berlin, Heidelberg: Springer, 2009, pp. 61–67, ISBN: 978-3-642-04260-7. DOI: 10.1007/978-3-642-04260-7_8.

[17] T. R. Henderson and R. H. Katz, "TCP performance over satellite channels," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-99-1083, Dec. 1999, http://www2.eecs.berkeley.edu/Pubs/TechRpts/1999/5306.html.

[18] P. Hurtig, W. John, and A. Brunstrom, *Recent Trends in TCP Packet-Level Characteristics*, en, https://www.semanticscholar.org/paper/Recent-Trends-in-TCP-Packet-Level-Characteristics-Hurtig-John/6fef36d3285997391a9d7d5259dd78c9d63a81aa, 2011.

[19] K. Pentikousis and H. Badr, "Quantifying the deployment of TCP options-a comparative study," *IEEE Communications Letters*, vol. 8, no. 10, pp. 647–649, 2004, ISSN: 1089-7798.

[20] IETF, *RFC 3135 - Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*, https://tools.ietf.org/html/rfc3135, 2001.

[21] C. Caini and R. Firrincieli, "DTN and Satellite Communications," en, in *Delay Tolerant Networks: Protocols and Applications*, CRC Press, Apr. 2016, ISBN: 978-1-4665-1301-3.

[22] A. Pirovano and F. Garcia, "A New Survey on Improving TCP Performances over Geostationary Satellite Link," en, *Network and Communication Technologies*, vol. 2, no. 1, p1, Jan. 2013, ISSN: 1927-064X. DOI: 10.5539/nct.v2n1p1.

[23] L. Li, X. Tian, G. Chen, K. Pham, and E. Blasch, "Secure spectrum-efficient frequency hopping for return link of protected tactical satellite communications," in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, Nov. 2016, pp. 254–258. DOI: 10.1109/MILCOM.2016.7795335.

[24] G. Zheng, P.-D. Arapoglou, and B. Ottersten, "Physical Layer Security in Multibeam Satellite Systems," *IEEE Transactions on Wireless Communications*, vol. 11, no. 2, pp. 852–863, Feb. 2012, ISSN: 1558-2248. DOI: 10.1109/TWC.2011.120911.111460.

[25] CCSDS, "Space Data Link Security Protocol - Summary of Concept and Rationale," Green Book, Jun. 2018, https://public.ccsds.org/Pubs/350x5g1.pdf.

[26] W. Li and D. Gu, "Security Analysis of DVB Common Scrambling Algorithm," in *The First International Symposium on Data, Privacy, and E-Commerce (ISDPE 2007)*, Nov. 2007, pp. 271–273. DOI: 10.1109/ISDPE.2007.63.

[27] L. Duquerroy, S. Josset, O. Alphand, P. Berthou, and T. Gayraud, "SatiPSec : An Optimized Solution for Securing Multicast and Unicast Satellite Transmissions," in *22nd AIAA International Communications Satellite Systems Conference & Exhibit 2004 (ICSSC)*, American Institute of Aeronautics and Astronautics, May 2004. DOI: 10.2514/6.2004-3177.

[28] L. Djeddai and R. K. Liu, "IPSecOPEP: IPSec over PEPs architecture, for secure and optimized communications over satellite links," in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Aug. 2016, pp. 264–268. DOI: 10.1109/ICSESS.2016.7883063.

[29] Encore Networks, *BANDIT 2*, en-US, https://www.encorenetworks.com/products/bandit-ii/.

[30] DSD Telecom, *DSD Satellite VPN*, http://www.dsdtelecom.com/index.php/support/satellite-vpn.

[31] Newtec, *EL810 Mobile PEP-Box Terminal*, http://www.tellitec.be/tellinet/Newtec-Elevation-El810\%20Mobile\%20PEP-Box\%20Terminal\%20-\%20web\%20print.pdf, Oct. 2008.

[32] RigNet, *CyphreLink - End-to-End Data Protection*, en-US, https://www.cyphre.com/cyphrelink/.

[33] Hughes, *HN7000S Modem Product Page*, https://www.hughes.com/technologies/broadband-satellite-systems/hn-systems/hn7000s.

[34] Tellitec Gmbh., *TC-Spy Documentation*, https://wikileaks.org/wiki/Tellitec_Tellinet_Sat_Spy_manual,_6_Mar_2006, Mar. 2006.

[35] A. Adelsbach and U. Greveler, "Insider Attacks Enabling Data Broadcasting on Crypto-Enforced Unicast Links," en, in *Computer Security – ESORICS 2007*, J. Biskup and J. López, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2007, pp. 469–484, ISBN: 978-3-540-74835-9. DOI: 10.1007/978-3-540-74835-9_31.

[36] Ground Control, *IG-VPN: Using Applicaiton Layer Technology to Overcome the Impact of Satellite Circuit Latency on VPN Performance*, https://www.groundcontrol.com/IG-VPN_Intro.pdf, 2003.

[37] D. Lacamera and S. Ammirata, *PEPsal: A TCP Performance Enhancing Proxy for Satellite Links*, https://github.com/danielinux/pepsal, Sep. 2016.

[38] OpenVPN Inc., *OpenVPN*, https://openvpn.net/, 2020.

[39] Newtec, *TL100 TelliNet V2.6 Tellitec Product Family*, https://www.newtec.eu/frontend/files/leaflet/nop1800-nop1805-acceleration-and-compression-software.pdf, 2009.

[40] S. Chen, S. Jero, M. Jagielski, A. Boldyreva, and C. Nita-Rotaru, "Secure Communication Channel Establishment: TLS 1.3 (over TCP Fast Open) vs. QUIC," en, in *Computer Security – ESORICS 2019*, K. Sako, S. Schneider, and P. Y. A. Ryan, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2019, pp. 404–426, ISBN: 978-3-030-29959-0. DOI: 10.1007/978-3-030-29959-0_20.

[41] I. Swett, *QUIC FEC v1 - Google Groups*, https://groups.google.com/a/chromium.org/forum/\#!topic/proto-quic/Z5qKkk2XZe0, Message Board, Feb. 2016.

[42] L. Thomas, E. Dubois, N. Kuhn, and E. Lochin, "Google QUIC performance over a public SATCOM access," en, *International Journal of Satellite Communications and Networking*, vol. 37, no. 6, pp. 601–611, 2019, ISSN: 1542-0981. DOI: 10.1002/sat.1301.

[43] J. P. Rula, J. Newman, F. E. Bustamante, A. M. Kakhki, and D. Choffnes, "Mile high wifi: A first look at in-flight internet connectivity," in *Proceedings of the 2018 World Wide Web Conference*, International World Wide Web Conferences Steering Committee, 2018, pp. 1449–1458, ISBN: 1-4503-5639-7.

[44] N. Kuhn, J. Border, and E. Stephan, *QUIC for SATCOM*, https://www.potaroo.net/ietf/idref/draft-kuhn-quic-4-sat/, Nov. 2019.

[45] M. Fischlin and F. Günther, "Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates," in *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, Apr. 2017, pp. 60–75. DOI: 10.1109/EuroSP.2017.18.

[46] G. Delannoy, *TCPeP*, https://github.com/GregoireDelannoy/TCPeP, May 2013.

[47] D. Velenis, D. Kalogeras, and B. Maglaris, "SaTPEP: A TCP Performance Enhancing Proxy for Satellite Links," en, in *NETWORKING 2002: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, E. Gregori, M. Conti, A. T. Campbell, G. Omidyar, and M. Zukerman, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2002, pp. 1233–1238, ISBN: 978-3-540-47906-2. DOI: 10.1007/3-540-47906-6_113.

[48] L. Clemente, *Quic-go*, https://github.com/lucas-clemente/quic-go, Aug. 2019.

[49] Google, *QUIC, a multiplexed stream transport over UDP - The Chromium Projects*, https://www.chromium.org/quic, 2020.

[50] A. Auger, E. Lochin, and N. Kuhn, "Making Trustable Satellite Experiments: An Application to a VoIP Scenario," en, in *IEEE 89th Vehicular Technology Conference: IEEE VTC2019-Spring*, https://doi.org/10.1109/VTCSpring.2019.8746404, 2019, pp. 1–5.

[51] SitespeedIO, *Browsertime*, sitespeed.io, https://github.com/sitespeedio/browsertime, Dec. 2019.

[52] Mozilla.org, *PerformanceTiming.navigationStart*, en, https://developer.mozilla.org/en-US/docs/Web/API/PerformanceTiming/navigationStart, Documentation, Sep. 219.

[53] Alexa, *The Top500 Sites on the Web*, https://www.alexa.com/topsites, Jan. 2020.

[54] U. Speidel, *Simulating satellite Internet traffic to a small island Internet provider*, en-US, https://isif.asia/simulating-satellite-internet-traffic-to-a-small-island-internet-provider/, Jan. 2017.

[55] ETSI, "ETSI TR 102 768: Digital Video Broadcasting (DVB); Interaction channel for Satellite Distribution Systems; Guidelines for the use of EN 301 790 in mobile scenarios," Tech. Rep. ETSI TR 102 768, 2009, https://www.etsi.org/deliver/etsi_tr/102700_102799/102768/01.01.01_60/tr_102768v010101p.pdf.

[56] ——, "ETSI TR 102 376-1 Digital Video Broadcasting (DVB); Implementation guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 1: DVB-S2," Technical Report ETSI TR 102 376-2, 2015, https://www.etsi.org/deliver/etsi_tr/102300_102399/10237601/01.02.01_60/tr_10237601v010201p.pdf.

[57] ITU, "ITU-T Y.1541 Network performance objectives for IP-based services," Y-Series Recommendation, Dec. 2011, https://www.itu.int/rec/T-REC-Y.1541/en.

[58] A. Boubaker, *OpenSAND Example of Delay Variations*, https://wiki.net4sat.org/, Jun. 2019.

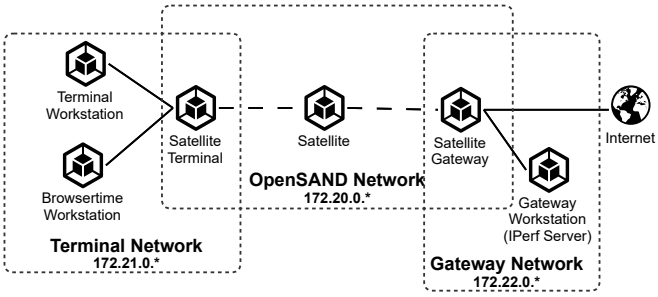
[59] A. Custura, T. Jones, and G. Fairhurst, "Rethinking ACKs at the Transport Layer," in *2020 IFIP Networking Conference (Networking)*, Jun. 2020, pp. 731–736.

[60] T. Jones, A. Custura, and G. Fairhurst, "Changing the Default QUIC ACK Policy," en, IETF Informational Draft Memo, Sep. 2020, https://tools.ietf.org/html/draft-fairhurst-quic-ack-scaling-03.

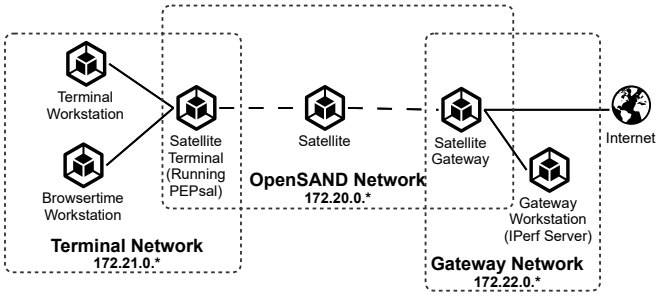
X. Appendix A - Testbed Network Configurations

For clarity, we have provided a number of network configuration diagrams to detail the testbed configuration used for each of the five scenarios commonly referenced in our benchmark comparisons. Further technical details can be found in the QPEP source repository.

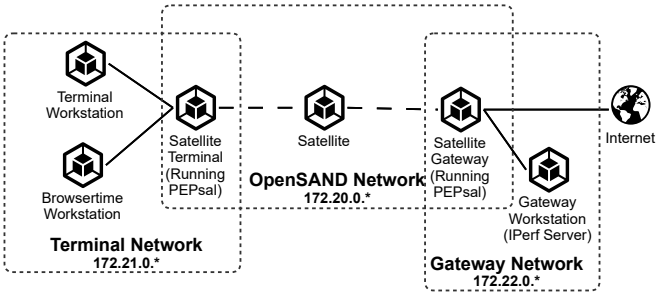
Fig. 11. Testbed Configurations



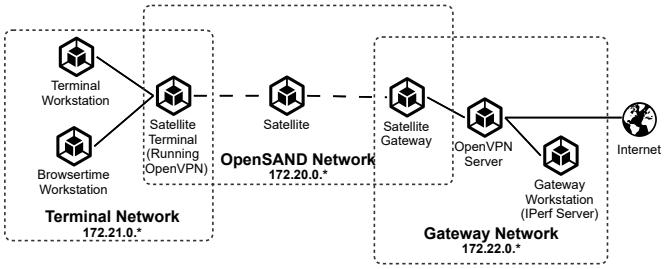
(a) Plain Satellite Connection Testbed Architecture



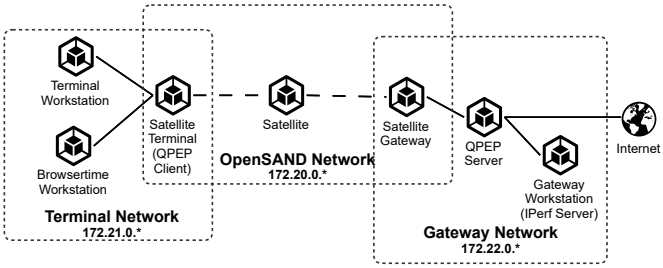
(b) Integrated PEPsal Testbed Architecture



(c) Distributed PEPsal Testbed Architecture



(d) OpenVPN Testbed Architecture



(e) QPEP Testbed Architecture