ETSI TR 102 676 V1.1.1 (2009-11)

Technical Report

Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Performance Enhancing Proxies (PEPs)

Reference

DTR/SES-00294

Keywords

architecture, broadband, IP, management, multimedia, satellite

Contents

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Satellite Earth Stations and Systems (SES).

Introduction

The present document presents an overview of PEP issues over satellites and focuses on BSM-related issues. It is based on current ETSI BSM architecture documents [i.1] and [i.2]. Also it is aligned with the relevant IETF standards. The IETF documented general PEP issues are described in RFC 3135 [i.3]. However, RFC 3135 [i.3] is not satellite specific and, more importantly, is now seven years old.

Also the present document is aligned with the Satlabs group solution called Interoperable PEP (I-PEP) that is aimed at DVB-RCS systems [i.4].

1 Scope

The present document aims to describe the current solutions for Performance Enhancing Proxies in broadband multimedia satellite systems. The range of PEPs considered includes TCP accelerators, TCP header compression and HTTP proxies. The PEPs are classified in terms of ease of implementation, interworking capability with other PEPs and performance potential.

Analysis of various PEP types/mechanisms and recommendations are made for using PEPs in BSM networks. Also recommendations are made for further work to support the introduction of PEPs in satellite systems, and in particular their introduction into the BSM architectures and standards.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
    - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
    - for informative references.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

Not applicable.

2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area. For non-specific references, the latest version of the referenced document (including any amendments) applies.

[i.1] ETSI TS 102 465: "Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); General Security Architecture".

[i.2] ETSI TS 102 292: "Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM) services and architectures; Functional architecture for IP interworking with BSM networks".

[i.3] IETF RFC 3135 (June 2001): "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations".

[i.4] I-PEP specifications, Issue 1a. Satlabs group recommendations (October 2005).

NOTE: Available at http://www.satlabs.org.

[i.5] ETSI TS 102 463: "Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Interworking with IntServ QoS".

[i.6] ETSI TS 102 464: "Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Interworking with DiffServ Qos".

[i.7] ETSI TS 102 466, "Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Multicast Security Architecture".

[i.8] IETF RFC 4326: "Unidirectional Lightweight Encapsulation (ULE) for Transmission of IP Datagrams over an MPEG-2 Transport Stream (TS)".

[i.9] ETSI EN 301 790: "Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems".

[i.10] Xiplink.

NOTE: See http://www.xiplink.com/.

[i.11] Space Bel - SPB-FS-651-DS-001 (February 2004): "FastSat".

NOTE: Available at http://www.spacebel.be/FR/Space/FastSatDataSheet.pdf.

[i.12] HUGHES: "Delivering outstanding application performance over satellite".

NOTE: Available at:
http://www.direcway.com/HUGHES/Doc/0/SIKPBJS69O6KP42VCE4K4ER2BF/Hughes%20PEP-H35661-A4-LR-091206.pdf.

[i.13] IEEE A&E Systems Magazine (August 2007): "PEPsal: A Performance Enhancing Proxy for TCP Satellite Connections", C. Caini, R. Firrincieli, D. Lacamera.

[i.14] IETF RFC 5458 (March 2009): "Security requirements for the Unidirectional Lightweight Encapsulation (ULE) protocol".

[i.15] V. Obanaik (2006): "Secure performance enhancing proxy: To ensure end-to-end security and enhance TCP performance over IPv6 wireless networks". Elsevier Computer Networks 50 (2006) 2225-2238.

[i.16] S. Bellovin (February 1997): "Probable plaintext cryptanalysis of the IPSecurity protocols, Proceedings of the Symposium on Network and Distributed System Security".

[i.17] IETF RFC 5246 (August 2008): "The Transport Layer Security (TLS) Protocol Version 1.2".

[i.18] L. Moser, etal (February 2007): "Building Dependable and Secure Web Services". Journal of Software, Vol. 2, N . 1.

[i.19] G. Giambene, S. Kota (September - October 206): "Cross-layer Protocol Optimization for Satellite Communications Networks: A Survey", Int. Journal Sat. Communications and Networking, Vol. 24, pp. 323-341.

[i.20] G. Giambene, S. Hadzic: "A Cross-Layer PEP for DVB-RCS Networks", to be presented at the First International Conference on Personal Satellite Services 2009 (PSATS2009), March 18-19, 2009, Rome, Italy.

[i.21] P. Chini, G. Giambene, D. Bartolini, M. Luglio, C. Roseti: "Dynamic Resource Allocation based on a TCP-MAC Cross-Layer Approach for DVB-RCS Satellite Networks", Int. Journal Sat. Communications and Networking, Vol. 24, pp. 367-385, September-October 2006.

[i.22] C. Gomez, etal: "Web browsing optimization over 2.5G and 3G: end-to-end mechanisms vs. usage of performance enhancing proxies". Wireless Communications and Mobile Computing. 2008; 8:213-230. Wiley InterScience.

[i.23] ESA ARTES-1 programme, 2006: "Transport Protocol for DVB-RCS Interoperable PEP".

[i.24] C. Roseti and E.Kristiansen: "TCP behaviour in a DVB-RCS environment". In Proceedings 24th AIAA International Communications Satellite Systems Conference (ICSSC), San Diego, 2006.

[i.25] IETF RFC 4614 (September 2006): "A Roadmap for Transmission Control Protocol (TCP) Specification Documents".

[i.26] IETF RFC 2581 (April 1999): "TCP Congestion Control".

[i.27] Space Communications Protocol Specification (SCPS)-Transport Protocol (SCPS-TP). "Recommendation for Space Data System Standards, CCSDS 714.0-B-2". Blue Book. Issue 2. Washington, D.C.: CCSDS, October 2006.

[i.28] C. Dovrolis, P. Ramanathan, D. Moore, "What do packet dispersion techniques measure?", in Proceedings of IEEE INFOCOM, pp. 905-914, Apr. 2001.

[i.29] M. Karaliopoulos, R. Tafazzoli, B.G. Evans, "Providing Differentiated Services to TCP Flows Over Bandwidth on Demand Geostationary Satellite Networks", IEEE Journal on Selected Areas in Communications, vol. 22, No. 2, Feb. 2004.

[i.30] M. Sooriyabandara, G. Fairhurst, "Dynamics of TCP over BoD satellite networks, International Journal of Satellite Communications and Networking", Vol. 21, No. 4-5, Jul. 2055, pp. 427-449.

[i.31] M. Luglio, C. Roseti, F. Zampognaro, "Performance evaluation of TCP-based applications over DVB-RCS DAMA schemes", International Journal of Satellite Communications and Networking, Vol. 27, Issue 3, pp. 163-191, Published online 2 Mar 2009, DOI: 10.1002/sat.930.

[i.32] IETF RFC 5374: "Multicast Extensions to the Security Architecture for the Internet Protocol".

[i.33] IETF RFC 793: "Transmission Control Protocol".

[i.34] IETF RFC 1122: "Requirements for Internet Hosts - Communication Layers".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

distributed PEP: PEP client and server are located at both ends of the satellite link (BSM ST and Gateway)

GateWay PEP (GW PEP): PEP server located near the BSM Gateway

integrated PEP: there is only one PEP entity residing with the satellite gateway (BSM Gateway)

interoperable PEP (I-PEP): functional architecture assumes a split-connection approach with the I-PEP server and a client both capable of supporting the I-PEP protocol

NOTE 1: The I-PEP protocol consists of a transport protocol heavily based on TCP and modified/augmented by SCPS-TP as well as a session protocol comprising several optional additions to support service and session management.

NOTE 2: Specified by the ESA/Satlabs [i.4] and aims to provide enhancement for satellite-based communications.

Performance Enhancing Proxy (PEP): network agents designed to improve the end-to-end performance of some communications protocol such as Transmission Control Protocol (TCP)

NOTE: More information on Transmission Control Protocol (TCP) is available at http://en.wikipedia.org/wiki/Transmission_Control_Protocol.

ST PEP: PEP client located near the BSM ST terminal

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ACCE	ACK-based Capacity and Congestion Estimation
ACK	ACKnowledgement
A-PEP	Application layer PEP
BDP	Bandwidth Delay Product
BSM	Broadband Satellite Multimedia
CCSDS	Consultative Committee for Space Data Systems
cwnd	congestion window (TCP)
DAMA	Demand Assignment Multiple Access
DNS	Domaine Name System
DVB-RCS	Digital Video Broadcasting - Return Channel for Satellites
ESP	Encapsulated Security Protocol
FSS	Fixed Service Satellite
FTP	File Transfer Protocol
GW PEP	GateWay PEP
HPEP	HTTP PEP
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
M-ESP	Modified ESP
MF-TDMA	Multi Frequency - Time Division Multiple Access
MIB	Management Information Base
ML-IPSEC	Multilayer IPSEC protocol
MPE	Multi Protocol Encapsulation
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NCC	Network Control Centre
PEP	Performance Enhancing Proxy
QID	Queue ID
QIDSPEC	Queue ID SPECifications
QoS	Quality of Service
RTO	Retransmission Time Out
RTT	Round-Trip Time
RWIN	Receive WINdow
SACKs	Selective ACKnowledgements
SCPS	Space Communications Protocol Specification
SCPS-TP	Space Communications Protocol Specification-Transport Protocol
SID	Security association IDentity
SIP	Session Initiation Protocol
SI-SAP	Satellite Independent - Service Access Point
SSL	Secure Socket Layer
ST	Satellite Terminal
TBTP	Terminal Burst Time Plan
TCP	Transmission Control Protocol
TCPN	TCP Noordwijk
TF-ESP	Transport Friendly - ESP
TLS	Transport Layer Security
T-PEP	TCP (transport) layer - PEP
UDP	User Datagram Protocol
ULE	Unidirectional Lightweight Encapsulation
UMTS	Universal Mobile Telephone System
URL	Uniform Resource Locator
VPN	Virtual Private Network
X-SAP	Cross Layer Service Access Point

4 Need for PEPs in BSM networks

4.1 Performance improvement using standard end-to-end techniques

The original Internet adopted an end-to-end architecture, where a transport connection was between a pair of hosts, bound to a globally unique IP address and locally meaningful transport port at each end host. The literature background for end-to-end improvements to TCP and HTTP (without using PEPs) is presented in clauses A.1 and A.2.

There are two main reasons in favour of using end-to-end mechanisms for improving performance over satellite links:

1) End-to-end mechanisms are based on standard options and maintain end-to-end semantics. Thus, they are fully compliant with the Internet architecture.

2) Empirical results demonstrate a significant improvement, especially when adequate HTTP settings are used.

However, end-to-end techniques have the following drawbacks:

1) The design criteria of Internet servers aim to optimize server throughput. Such goal might be difficult to achieve, because the configuration of many Internet servers limits the number of parallel transport connections per session.

3) Certain parameters cannot be optimized at the same time for different access technologies. For example, the Bandwidth Delay Product (BDP, see annex A) in satellite networks is much larger than UMTS. Moreover, servers are by default unaware of the access technology used by a client.

4) At least one TCP Slow Start phase will still take place during a web page download, unless persistent connections are used. However, the configuration of many Internet servers seeks to minimize the amount of memory consumed per session, a side-effect of this is that servers often unwilling to hold state for connections which become passive.

5) Should multiple objects be hosted under different domain names, DNS lookup overhead cannot be avoided or reduced using end-to-end options.

6) The performance of end-to-end mechanisms reduces over paths that experience gaps in connectivity (e.g. due to a link outages). The reason is that a server is unable to distinguish between congestion and radio link losses. This can lead to unwanted activation of TCP congestion control mechanisms or timeouts and thus significantly reducing performance.

Considering the issues discussed above, optimization of current end-to-end methods can provide improvements, but as yet cannot provide optimal performance for satellite systems. An alternative solution is the use of PEPs (see clauses 4.2 and 4.3).

4.2 Motivation for using PEPs

The present document focuses on the current work in defining the PEP architecture for BSM satellite networks.

In general, the Internet transport protocol (namely TCP) exhibits suboptimal performance due to the following satellite characteristics:

- Long feedback loops: Propagation delay from a sender to a receiver in a geosynchronous satellite network can range from 240 to 280 milliseconds.

- Large bandwidth*delay products: TCP needs to keep a large number of packets "in flight" in order to fully utilize the satellite link.

- Asymmetric capacity: The return link capacity for carrying ACKs can have a significant impact on TCP performance.

An alternative solution to clause 4.1 is to place an entity called Performance Enhancing Proxy (PEP) somewhere between the endpoints of a communication link. We focus on TCP PEPs (T-PEP) and Application PEPs (A-PEP). clauses B.1, B.2 and B.3 present details on PEP types, transport and application layer PEPs. As a summary of this approach, among the TCP PEP proposals, one solution is represented by the splitting approach [i.3]. The rationale of the splitting concept is to separate the satellite portion from the rest of the network. This approach can be further be divided into two categories: Distributed PEPs where the PEP client and server are located at each end of the satellite link. The other category is Integrated PEPs with only one PEP entity residing with the satellite gateway. Typical TCP PEP improvements are:

- TCP Spoofing: Eliminates effects of satellite delay on TCPs slow start and window sizing.
- ACK Reduction: Reduces unnecessary acknowledgements to improve bandwidth efficiency.
- Flow Control: Employs network feedback to intelligently control traffic flow.
- Error Recovery: Works closely with flow control to recover damaged or lost packets.
- Traffic Prioritization: Classifies traffic by application protocol, matching this to the MAC layer.
- Connection Establishment Spoofing: Intelligently spoofs the TCP three-way handshake to speed up establishment of a connection.
- PEPs can also compress protocol information, or change protocol characteristics to match specific characteristics of the satellite channel.

In addition to TCP PEPs (T-PEP), there are other complementary solutions such as application layer PEPs (A-PEP), where web browsing is the major target for application PEPs. Typical application layer PEPs improvements are:

- HTTP pre-fetching: Intercepting requested Web pages, identifying Web objects referred to by the Web pages, downloading these objects in anticipation of the next user requests.
- Browser Cache Leveraging: Caching some web pages not residing in browser cache, improving efficiency.
- Bulk Transfer Prioritization: Prioritizes bulk transfers to prevent adverse effect on other Web traffic.
- Cookie Handling: Ensures accurate painting of Web pages with the proper cookies.
- Compression: Payload compression provides increased transmission speeds. In addition, header compression for TCP, UDP, and RTP protocols results in additional bandwidth savings.
- DNS caching techniques, to further improve bandwidth utilization.

Commercial PEPs normally combine some or all of the T-PEP and A-PEP techniques together such the Hughes [i.12], XipLink [i.10], FastSat [i.11], Newtec, TAS-F and STM PEPs. A summary of the various techniques used in PEP products is presented in annex C.

4.3 PEP terminal architecture and components

There are two possibilities for the location of ST PEP: one is being internal to the BSM ST as shown in figure 1a, where the PEP run as a software process above the SI-SAP in the ST itself. The other possibility, as shown in figure 1b, is that ST PEP is external to the BSM ST and connected to the BSM ST with an Ethernet cable. Figure 2 shows the PEP protocol stack with the BSM Gateway terminal architectures, where the common location is that the Gateway PEP is external to the BSM Gateway.

The PEP residing on the BSM ST side is called ST PEP (PEP client) and the one on the BSM gateway side is called Gateway PEP (GW PEP, PEP server). Both PEPs have a similar architecture with two interfaces, one to the BSM satellite network and one to terrestrial networks. On the satellite side, the ST/Gateway PEP are connected to BSM ST/Gateway through an Ethernet LAN (except the internal ST PEP). On the terrestrial network side, normally, the PEP terminal connects to host/hosts on the same LAN, while the gateway PEP connects to a content server through the general Internet. However, the Gateway PEP can be located remotely from the BSM Gateway terminal (such as Gateway PEP run by a service provider), more details are presented in clause 4.4.

Also, figures 1a, 1b and 2 show the Satellite Independent Service Access Point (SI-SAP) interface. It enables the BSM system to abstract the lower layer functions. It allows the network protocols developed in the satellite independent layer to perform over any BSM family (specific satellite technologies). Moreover, the SI-SAP also enables the use of standard Internet protocols for example address resolution, QoS, security and network management, directly over the satellite system or with minimal adaptation to satellite physical characteristics. Finally the SI-SAP even makes it possible to envisage switching from one satellite system to another and to even a non-satellite technology while preserving the BSM operator's investment in upper layers software developments.

The transport protocol in the PEP is divided between standard TCP/UDP and PEP specific transport protocols. As shown in figures 1a, 1b and 2, the PEP specific transport protocol can be:

- A modified TCP (TCP+) such as the Hybla protocols [i.13], which is used in integrated PEP configurations, where only Gateway PEP will be used (no ST PEP).
- Standard Interoperable PEP Transport Protocol (I-PEP TP), recommended by Satlabs [i.4] and used in the distributed PEP configurations. The I-PEP TP is based on an extension set to TCP termed SCPS-TP, which was produced by the Consultative Committee for Space Data Systems (CCSDS).
- Proprietary distributed Transport Protocol (TP+), where company specific (non-standard) protocols are used.

The ST/Gateway PEPs can be managed either locally or remotely. For remote management, either SNMP or HTTP protocols can be used to communicate with the BSM management system. In both cases the PEP monitoring and configuration controls can be based on the standard MIB II and enterprise specific PEP MIBs.

The optimum PEP performance is expected to require a close matching between the PEP configuration and the QoS provisioning of the associated lower layer bearer services. In some PEP implementations, there is a customized (proprietary) signalling between the PEP and the Satellite terminal. Such signalling can be used for QoS monitoring of the terminal queues and adjusting rate control parameters accordingly to maximize the use of the satellite capacity.

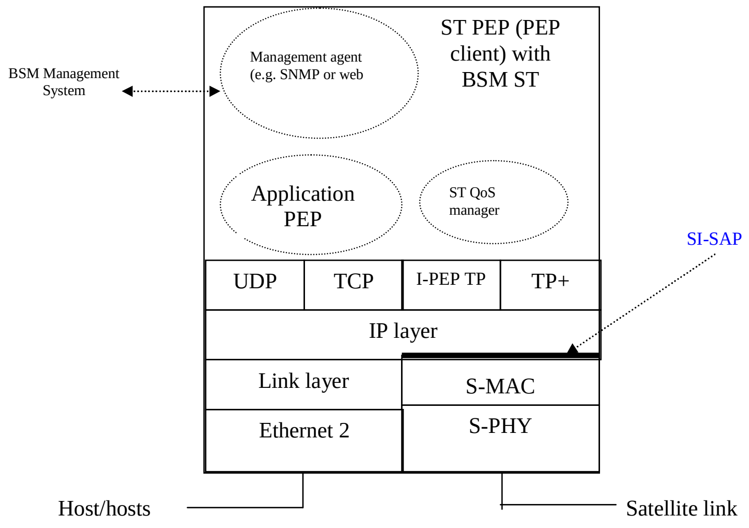

Figure 1a: BSM ST with internal PEP

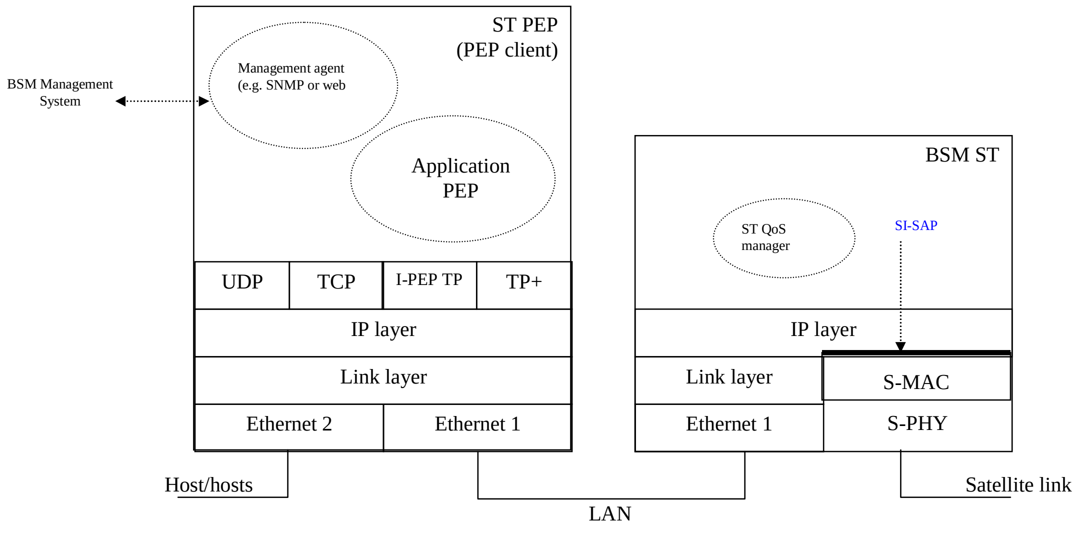

Figure 1b: BSM ST with external PEP

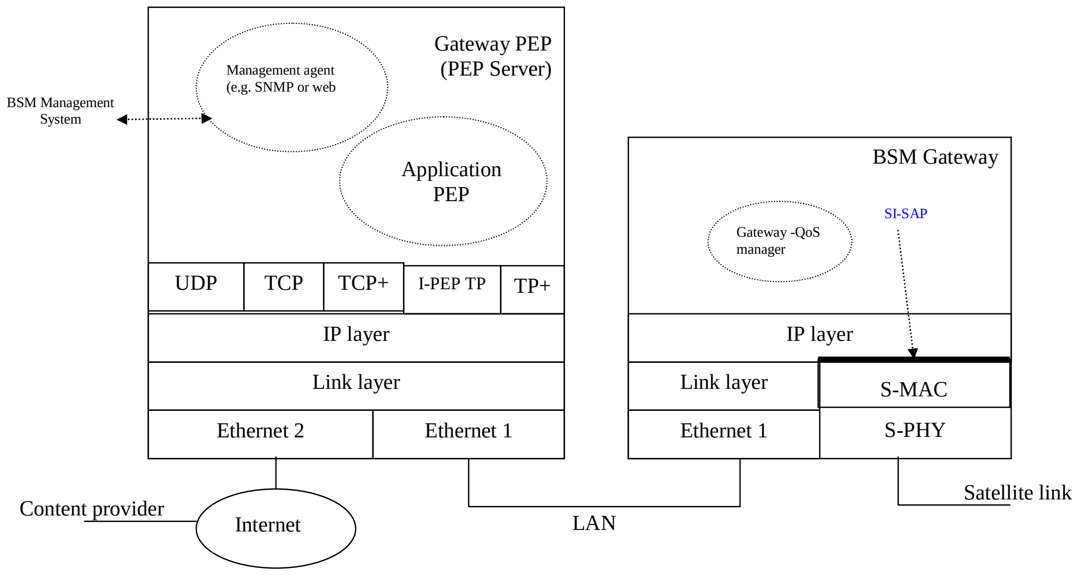

Figure 2: BSM Gateway PEP with external GW PEP

4.4 PEP scenarios

Several PEP usage scenarios are presented here following the Satlabs recommendations [i.4]. All scenarios apply to both star and mesh satellite link topologies.

4.4.1 Scenario 1: Single user

Figure 3 shows the single user scenario, where there is a clear one-to-one mapping between users and PEP clients (ST PEP). The multi-user scenario expands beyond the single user variant in that several application clients are served by the same PEP client. This reflects the typical home user or home office scenario. The PEP client may be integrated with the BSM ST, or it may be a stand-alone entity separate from both the end user's device and the ST.

The end-to-end TCP connection is split into three connections:

- The first connection is between the content provider and the GW PEP (standard TCP).
- The second connection is between the GW PEP and the ST PEP. The transport protocol here can be either proprietary or standardized PEP such as the I-PEP [i.4].
- The third connection is between the host and the ST PEP (standard TCP).

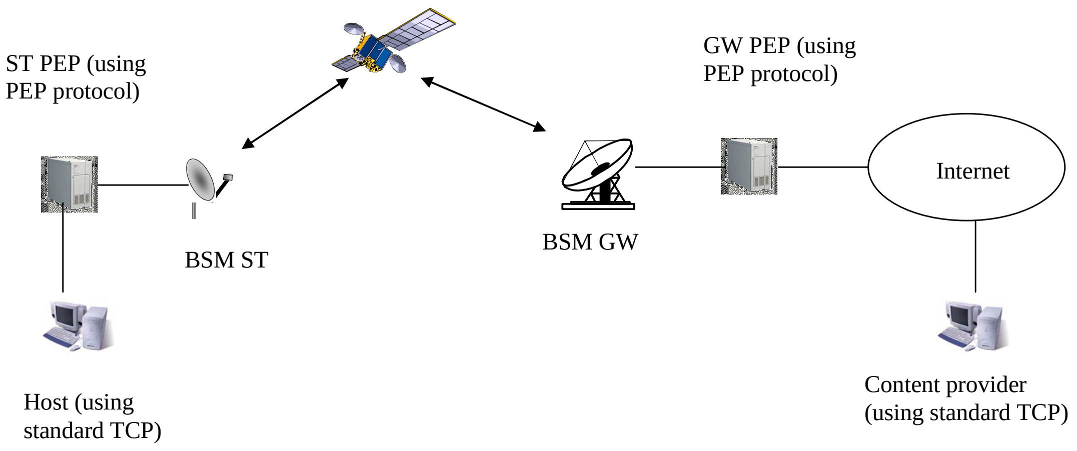
ST PEP (using PEP protocol)
GW PEP (using PEP protocol)
Internet
BSM ST
BSM GW
Host (using standard TCP)
Content provider (using standard TCP)

Figure 3: Scenario 1: ST PEP serving a single host with co-located BSM and PEP Gateways

4.4.2 Scenario 2: Independent satellite and PEP gateways

Scenario 1, assumed that the PEP server (Gateway PEP) is co-located with the BSM Gateway (which implies that the satellite service provider either operates the PEP or provides hosting facilities for the respective system components). In scenario 2, the PEP server is external to the BSM Gateway (see figure 4), motivating two different set-ups:

- PEP server may be run by a separate Internet Service Provider (ISP) on behalf of many users; or
- PEP server may be operated by an enterprise on its own behalf.

Similar to scenario 1, end-to-end TCP connection is split into three connections. However there are few difference:

- The communication link between the PEP server and BSM Gateway now extends through a wide area network that is not trusted and the satellite service provider may also not be trusted. A typical connection will be an IP tunnel (possibly with IPsec).
- Addressing schemes for PEP and BSM entities may be controlled by two different administrations.

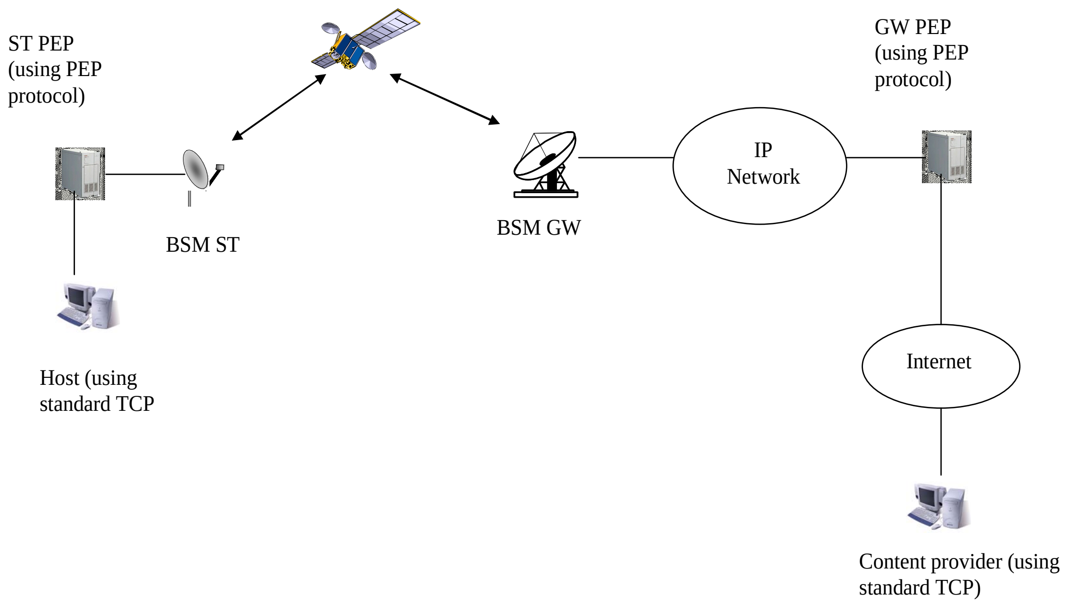

Figure 4: Scenario 2: Independent Gateway PEP from BSM Gateway

A variation in scenario 2 will be to have independent transport layer and application layer PEPs. So in figure 4, each PEP box (ST and GW PEPs) may include two functionally independent PEPs (e.g. T-TCP and A-PEP).

4.4.3 Scenario 3: Multiple PEP Gateways

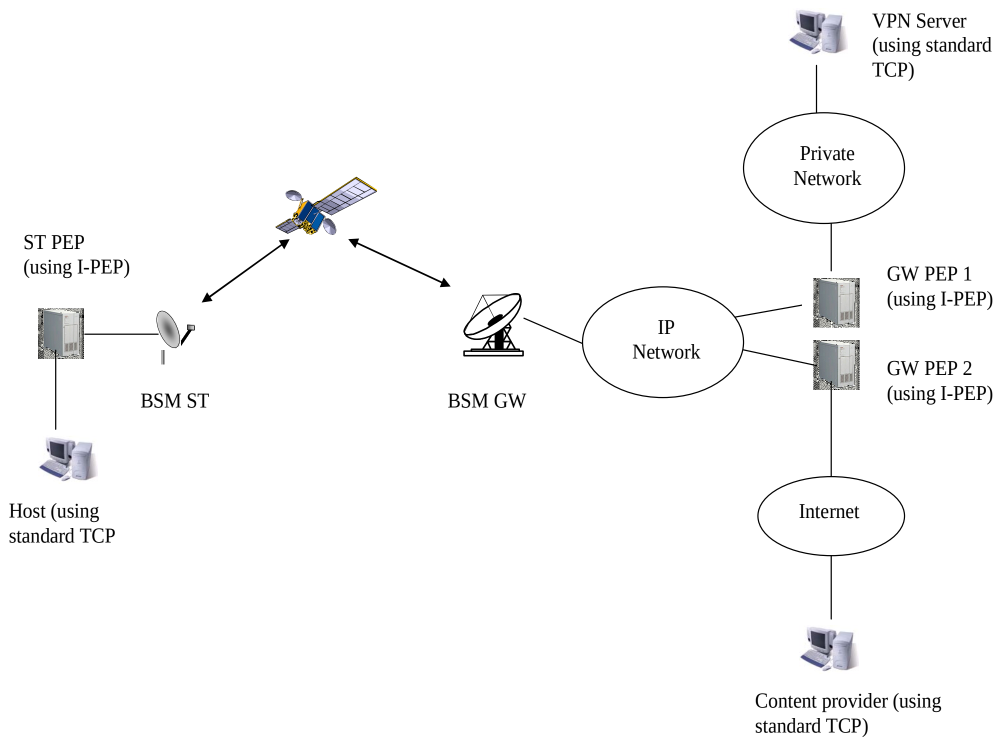

Figure 5: Scenario 3: ST PEP accessing multiple Gateway PEPs

This scenario is depicted in figure 5 and it is useful in cases where a remote terminal, has an IP tunnel for enterprise communications as well as a direct Internet connection for general communications. PEP acceleration is required for both and can only be operated independently.

In this scenario, there is no longer a single "centralized" Gateway PEP. Instead, multiple Gateway PEPs are used: either due to the presence of multiple ISPs or because performance enhancement is managed directly between user sites (VPN configuration).

Similar to scenario 1 and 2, end-to-end TCP connection is split into three connections. In comparison to scenario 2, the PEP client (ST PEP) needs to interoperate with multiple PEP servers from different vendors. Therefore, one possible solution is that the ST could house multiple PEPs. Another and more elegant solution is using the I-PEP protocol [i.4], where a single client PEP can communicate seamless with server PEPs from different vendors.

4.4.4 Scenario 4: Integrated PEP (single PEP)

The previous three scenarios showed various aspect of a distributed PEP (PEP client and server at both ends of the satellite link, like the ST and Gateway PEPs). Scenario 4 shows an example of an integrated PEP (see figure 6).

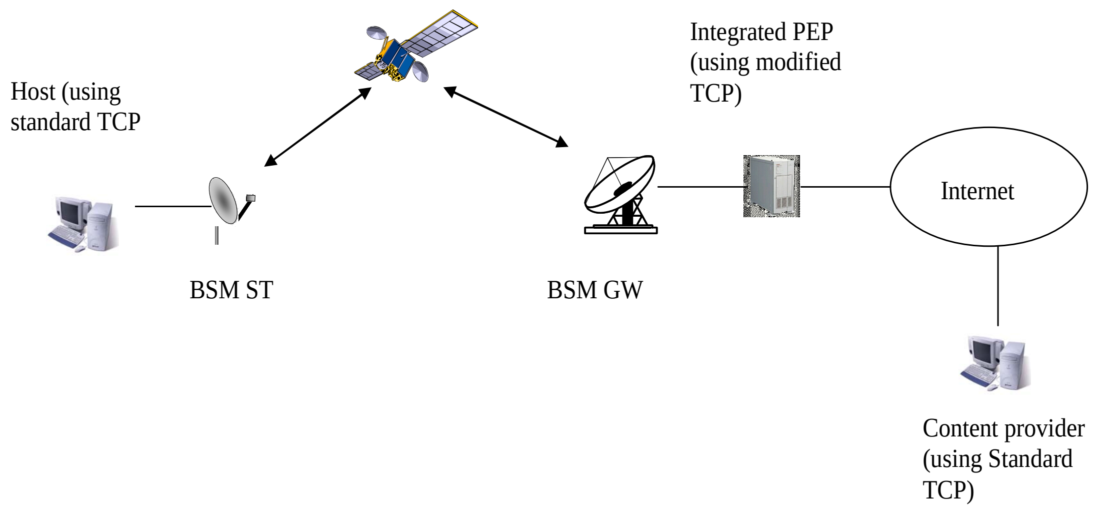
Host (using standard TCP
Integrated PEP (using modified TCP)
Internet
BSM ST
BSM GW
Content provider (using Standard TCP)

Figure 6: Scenario 4: Integrated PEP implemented at the BSM Gateway

Here the TCP connection established among the end hosts, is split in two separated connections, with the integrated PEP at the BSM Gateway. The first connection (between the web server and the integrated PEP makes use of the TCP standard and is terminated at the PEP. The second connection, between PEP and the final user, can exploit an enhanced TCP version compatible with a standard TCP receiver (such as the Hypla protocol [i.13]). In comparison to the distributed PEPs scenarios, integrated PEP is simpler but has limited enhancement capabilities.

4.5 Cross layer improvements

Due to the bandwidth-limited (and sometimes dynamic) nature of the radio channel, there is tight interdependence between the performance of protocol layers in satellite systems. "Cross-layering" is the mechanism that exploits interactions between protocols at adjacent or non-adjacent layers (i.e. exchanging of information/commands related to the 'internal state' of protocols, meaning here the 'variables' that are typical of a layer, representing its state or behaviour; for instance:

(i) queue length for MAC layer or IP layer;

(ii) congestion window value for TCP at transport layer) to improve system performance.

Cross-layer signalling exchange can be in downward or upward directions in the protocol stack (i.e. following the same direction of the application data flow or in a feedback direction, respectively). Downward signalling could be used to notify information such as: application and audio/video codec type, QoS requirements, priority, protocol type and internal protocol state. Upward signalling could be employed to send information concerning: available air interface options, propagation conditions, handover preparation measures, congestion notification, policing options, and a request to change the codec type (the reference is here to a scalable application layer) depending on the varying capacity offered by the PHY layer, referring to an air interface supporting adaptive modulation and coding. In general, two methods could be used to support the exchange of signalling across layers [i.19]:

- *In-band signalling* with the use of enriched packet headers to notify internal state variables to either other layers within a given host (internal cross-layering) or in a peer-to-peer case with another host or gateway (external cross-layering). This method needs the redefinition of packets headers (e.g. unallocated header fields or new message type) and can be used for signalling going in the same direction of related data (downward/forward signalling).

- *Out-of-band signalling* via the control plane, that is the use of new primitives and suitable Service Access Points (SAPs) to allow the dialogue between protocol layers.

Cross-layering requires to adopt mechanisms to allow the exchange of signalling also among non-adjacent layers. The coordination of signalling could be made by a protocol layer (horizontal approach) or by an external controller that is common to all the layers (vertical approach). In the first case, as shown in figure 7a, the coordinating protocol layer can have direct interfaces (SAPs) with adjacent layers and cross-layer interfaces (X-SAPs) between non-adjacent layers (e.g. creating holes in the protocol stack by means of the Internet Control Message Protocol (ICMP) [i.29]. In the second case, a global coordinator of different layers has interfaces (X-SAPs), with all the protocol layers and can have control on their internal state variables, reading and modifying them, depending on triggering events (see figures 7a and 7b).

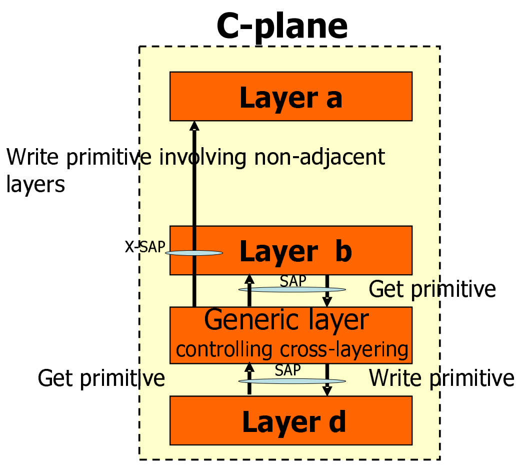

Figure 7a: Diagram of cross-layer signalling exchange among protocol layers horizontal approach with a protocol having the control

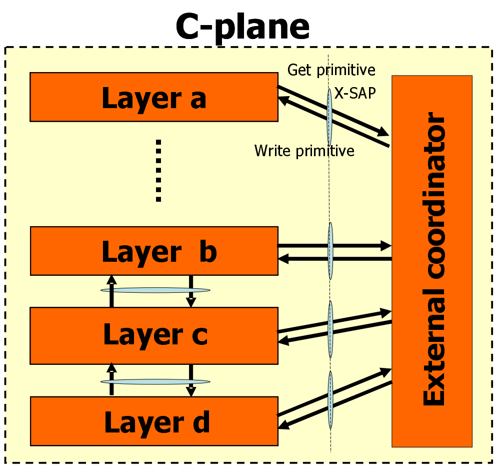

Figure 7b: Diagram of cross-layer signalling exchange among protocol layers vertical approach with an external coordinator

A classical transport-layer PEP does not use cross-layer information to manage TCP flows. This non-cross-layer type of PEP is called sometimes "Blind-PEP". Alternatively, a PEP that exploits cross-layer signalling is called "Sighted PEP".

The cross-layer PEP scheme envisaged here is a transport-layer integrated PEP operated on the NCC side and based on the horizontal cross-layer signalling approach, where MAC layer has the control of signalling exchange [i.19]. In this study the NCC/PEP is collocated with the BSM Gateway. In satellite networks, the bottleneck link is the satellite one. Hence, the NCC/gateway can have a direct control on it via the resource allocation decided at layer 2: the NCC/PEP can thus anticipate congestion events and use an upward cross-layer signalling (out-of band message) to notify its transport layer when the capacity available in the satellite network is close to be saturated. Then, modified ACK* (in-band signalling) could be used to notify the TCP sender to stop the traffic injection increase. In a Blind PEP this is not possible, thus causing large packet losses with consequent possible time out events with a significant drop of the TCP congestion window for a long time interval. These additional layer functionalities of the Sighted PEP need that the NCC/PEP supports a cross-layer signalling dialogue between TCP and MAC layer. Annex E presents a detailed description of the functionalities needed to support the "Sighted PEP" for File Transfer Protocol (FTP) data flows from the ST (TCP sender) through the gateway towards the Internet, where an integrated PEP is co-located with the gateway.

Finally, Excessive use of cross-layering can increases implementation complexity and dependence on the protocol details of other layers, classically regarded as layer-violation. This may raise cause difficulties when the protocol specifications is updated, or new functions are added. Hence most of these techniques are still in research stage and are not widely implemented in existing PEP products.

5 Security impact on Performance Enhancing Proxies

5.1 Previous research work related to PEP security

Interworking between PEPs and security system has been researched in the past [i.15]. For example, many researchers had addressed the issue of interworking between IPsec and PEPs. One solution was the use of an intelligent switch at the PEP. As such, the PEP provides acceleration for the unencrypted packets, while the encrypted packets are allowed to bypass the PEP. With this approach, the applications can choose between security and performance, but both are not obtainable together.

Some modifications to IPsec had been proposed. Transport Friendly Encapsulated Security Protocol (TF-ESP) or Modified ESP (M-ESP) [i.15] proposes a modification to ESP header to accommodate the necessary TCP header information in the ESP header outside the scope of encryption. The mechanism proposes that the unencrypted TCP header information in ESP should be authenticated for integrity. Although this method addresses the performance issues, it exposes enough information to make the connection vulnerable to security threats [i.16].

The Multilayer IPSEC Protocol (ML-IPSEC) proposes a multi-layer encryption scheme. The IP datagram payload is divided into zones; each zone has its own security associations and protection mechanisms. For instance, the TCP data part can be a zone, using end-to-end encryption with the keys only shared between end-hosts. The TCP header could be another zone with security associations between the source, destination and a few trusted nodes (such as PEPs). The trusted nodes can decrypt the transport layer headers to provide the performance enhancements. This mechanism ensures security and can accommodate existing performance solutions. Though the requirements are satisfied, but the IPsec complexity increases. Also, the assumption that intermediate nodes are trustworthy may not be acceptable for users preferring end-to-end security.

Some other solutions explore the use of transport layer security. Secure Socket Layer (SSL) was proposed by Netscape and later standardized by IETF as Transport Layer Security (TLS) [i.17]. It is a transport layer mechanism that provides data security. It encrypts the user data, but not the transport layer headers, such as TCP headers. Since the transport layer headers are in plaintext, the intermediate nodes (such as PEPs) can access or modify them; thereby the performance related issues can be resolved. However, it is not recommended to have TCP headers in plaintext due to security concerns [i.16]. Suggestions were also made to use SSL/TLS with IPsec in order to protect the header information. Again there is increase in security complexity and overheads.

In summary, there is a requirement that security can be implemented in such a way that allows ST and Gateway PEPs to access the transport protocol headers (such as TCP). The most negative implication of using PEPs is breaking the end-to-end semantics of a connection which conflicts with the end-to-end security usage of IPsec and TLS.

5.2 Security solutions for BSM PEPs

The following clauses examine the detailed impact of transport, network and link layers security on PEP operations. All security solutions apply to distributed and integrated PEPs.

5.2.1 Interworking between PEPs and transport/application layer security systems

As shown in figure 8, security can be implemented above the transport layer such as using SSL or its variant TLS. Also application layer security can be applied such as secure web services [i.18].

Transport/application layer security will work with TCP PEPs (as described in clause I and clause B.2) because the TCP header is not encrypted by the security system. As such, the TCP PEP will function properly and seamlessly. However, if HTTP acceleration is used (A-PEP), then there is a problem regarding interworking with security. The reason is that application layer data will be encrypted by the security system. Hence, it will not be possible to perform techniques such as HTTP prefetching, caching and header and payload compressions (as described in clause I and clause B.3).

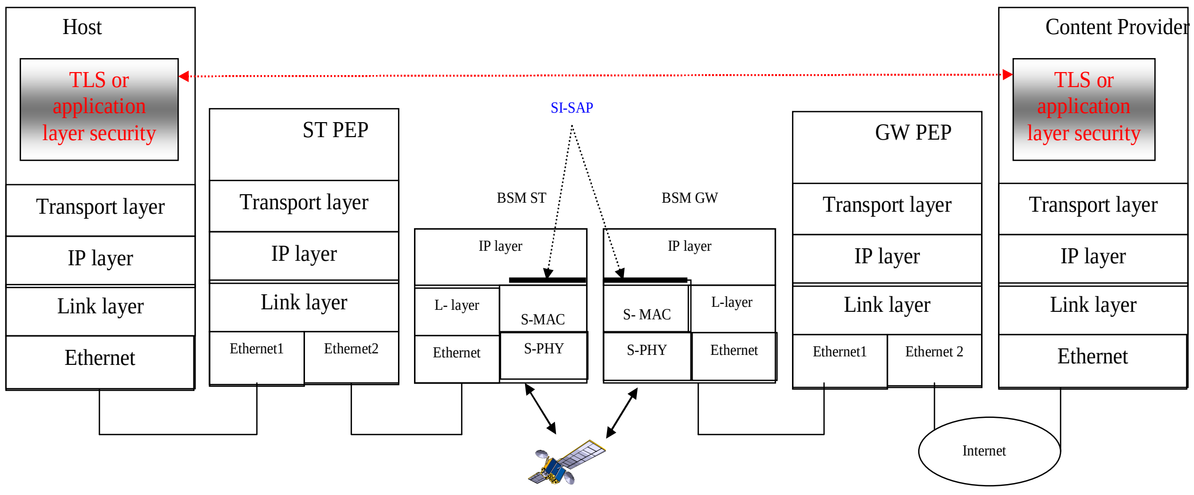

Figure 8: Distributed PEPs implementation with end-to-end transport/application layer security

5.2.2 Interworking between PEPs and IPsec

End-to-end network layer security (such as IPsec) will encrypt the TCP header and user data. Therefore TCP PEPs will not be able to perform techniques such as TCP spoofing, ACK reduction and flow control. In addition, the HTTP acceleration will not be able to perform HTTP prefetching, caching and compression. The reason is the encryption of IP packets via IPsec's ESP header (in either transport or tunnel mode) renders the TCP header and payload unintelligible to the PEPs. Without being able to examine the transport or application headers, the PEP may not function optimally or at all [i.1] and [i.7]. Thus a user or network administrator can choose between using PEPs or using IPsec.

However there are some steps which can be taken to allow the use of IPsec and PEPs to coexist. Through the use of security policies, an end user can select the use of IPsec for some traffic and not for other traffic, PEP processing can be applied to the traffic sent without IPsec.

Another alternative is to implement IPsec over the satellite link between the two PEPs of a distributed PEP implementation. This is not end-to-end use of the IPsec, but it will protect the traffic between the two PEPs. In some cases this may be acceptable, whereas in some other cases IPsec may be used to authenticate the origin of the data. If the IPsec endpoints are remote, this also requires remote placement of the PEPs.

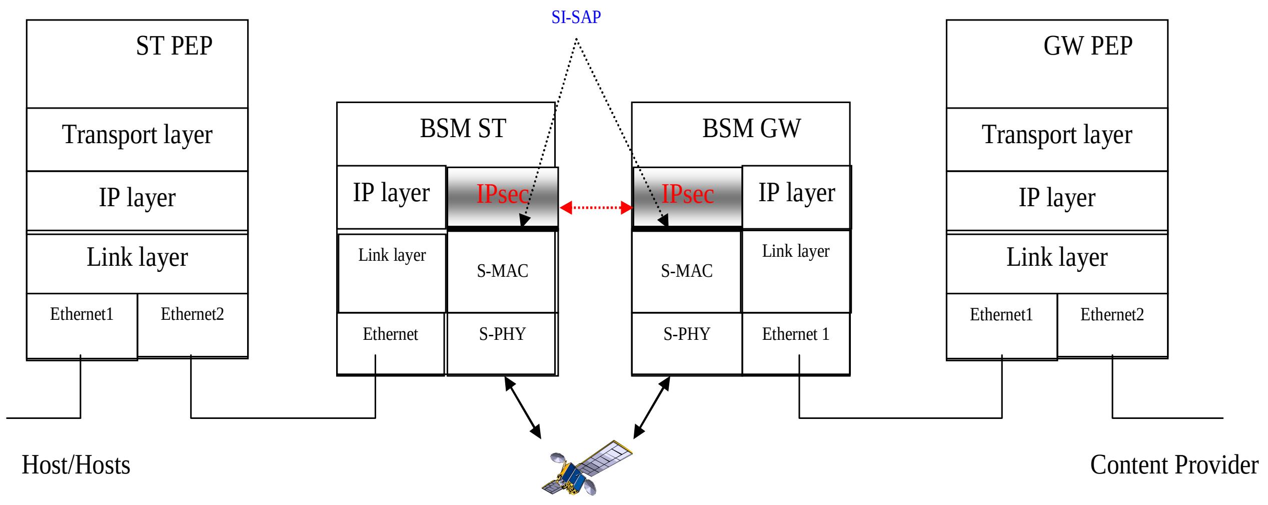

Figure 9: Distributed PEPs implementation with satellite link IPsec security

As shown in figure 9, PEPs can be used successfully with IPsec in tunnel mode between the BSM ST and Gateway. Here the encryption is performed on incoming traffic after the PEP operations and decryption is performed on outgoing traffic before the PEP operations. Another variation on this scenario is shown in figure 10, where the IPsec tunnel is configured between the ST PEP (PEP client) and the Gateway PEP (PEP server). Here the IPsec operations are under the control of the PEP entities. But the PEP cannot then perform link-layer optimizations, such as header compression or link-layer retransmission and may have limited access to cross-layer functions within the satellite terminal.

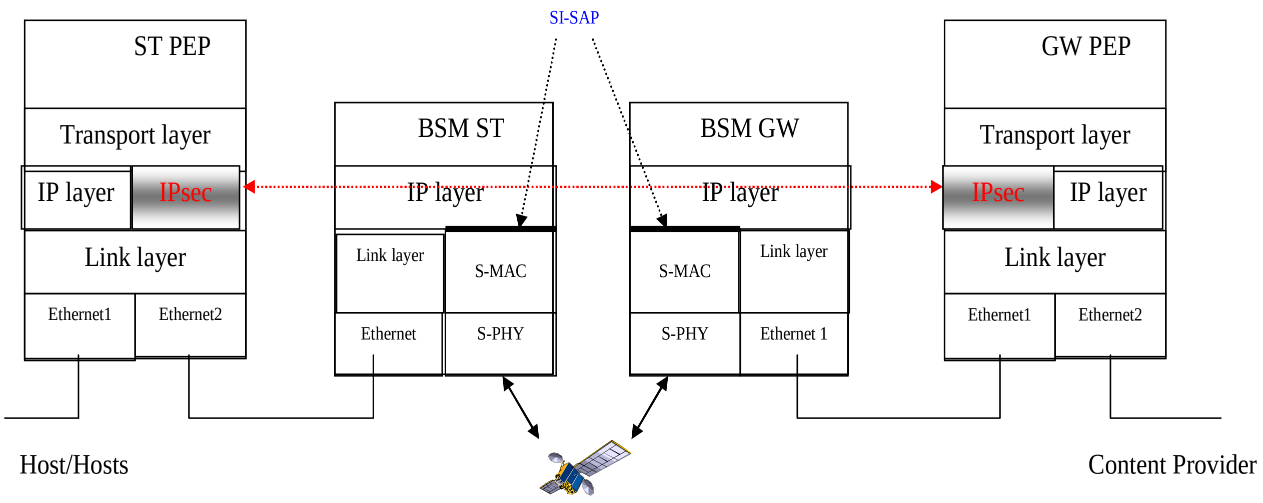

Figure 10: Distributed PEPs implementation with PEP IPsec security

In both case and in terms of overheads, IPsec tunnel mode requires an extra IP header, where basic IPv4 header is 20 bytes and IPv6 header is 40 bytes. Also IP multicasting over satellites can exploit the broadcast nature of satellites. However, secure multicasting with IPsec (in tunnel mode) has two more added implications: First, IP multicast becomes effectively point-to-point connections between the IPsec tunnel ends; second manual keying only is used. Therefore, the recently published RFC 5374 [i.32] (multicast extension to IPsec) provides an optional extension to IPsec to resolve these issues. However, the multicast extensions to IPsec might not be available on all BSM ST, Gateway or router equipment.

5.2.3 Interworking between PEPs and link layer security systems

As shown in figure 11, link layer security mechanism such as DVB-RCS [i.9] security or Unidirectional Lightweight Encapsulation (ULE) security [i.14] can be used. Here TCP and application layer PEPs will work seamlessly over the secure satellite link. The reason is TCP header and user data are handled in clear text (no encryption) between the ST and Gateway PEPs. Then, the satellite link layer security is only applied between the BSM ST and GW (satellite terminals).

Although link layer security does not provide the desired end-to-end security, it is more efficient than using IPsec (in tunnel mode). It also can provide extra security functions that are not possible IPsec or upper layer security such user identity hiding (e.g. IP and MAC addresses). This allows providing strong privacy service over the satellite broadcasting link. Further details on BSM link layer security is presented in the next clause.

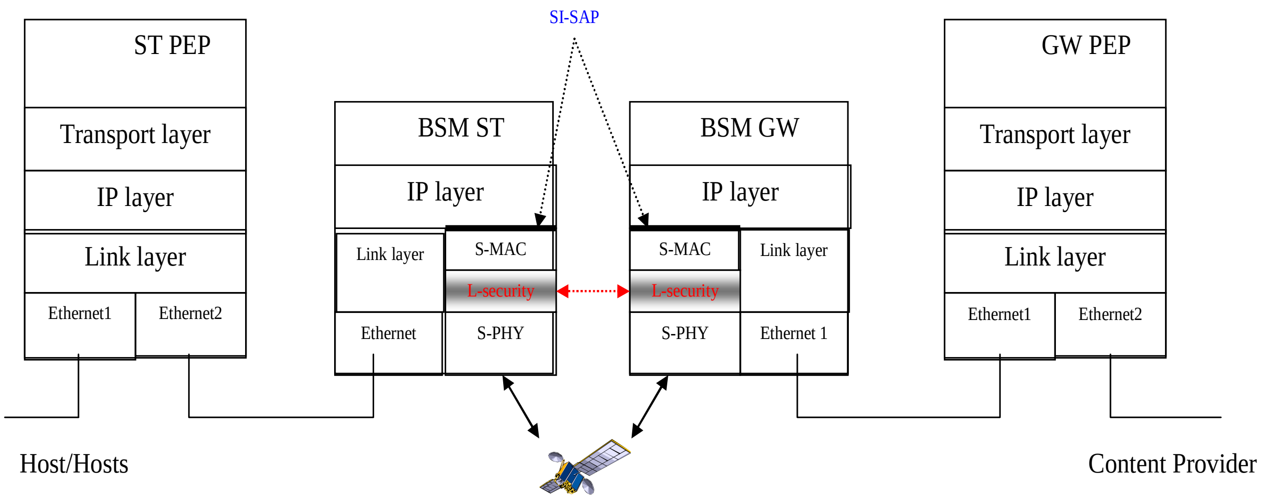

Figure 11: Distributed PEPs implementation with satellite link layer security

5.3 BSM link layer security architecture suitable for PEPs

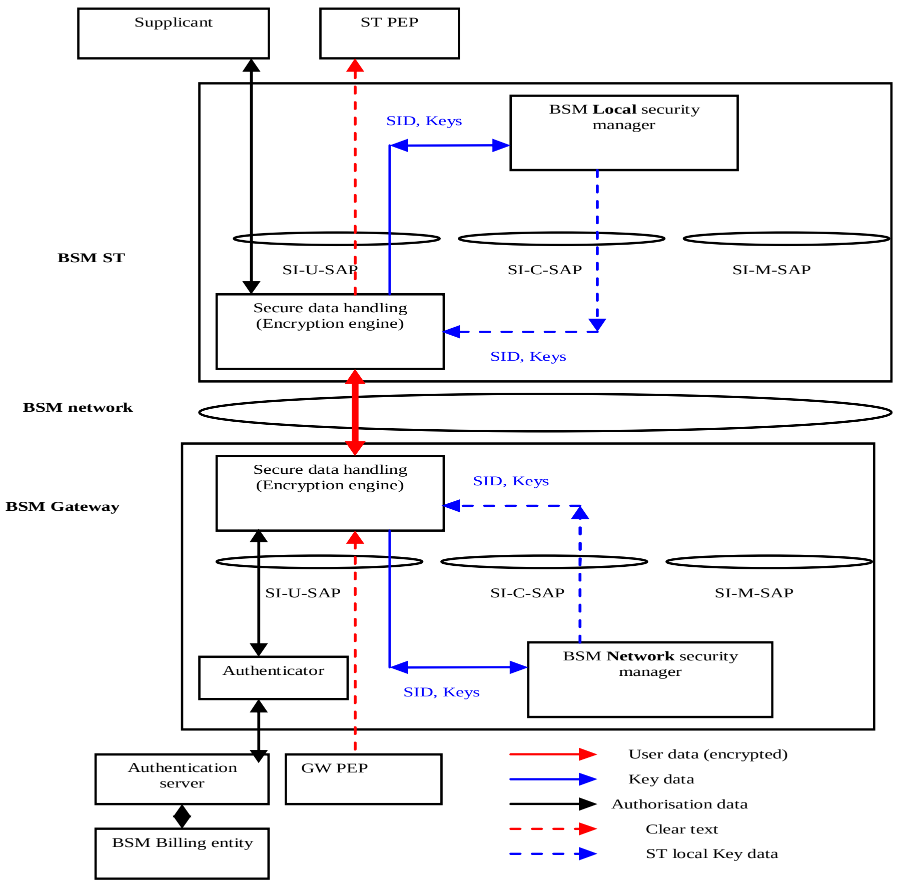

Figure 12: Mixed link layer BSM security entities

As shown in figure12, [i.1], the ST and Gateway PEPs can operate seamlessly with link layer security (below SI-SAP) such as DVB-RCS with Multi Protocol Encapsulation (MPE) or Unidirectional Lightweight Encapsulation (ULE) RFC 4326 [i.8]. In addition, figure 12 provided detailed key management architecture and security interactions across the BSM SI-SAP interface. The data encryption (and data integrity check) is performed below the SI-SAP, while the key management is performed above the SI-SAP within entities co-located with the BSM ST and Gateway (satellite terminals).

Also figure 12 shows the user authentication process (supplicant, authenticator and Authentication server entities), where secure link layer is used to carry authentication information (such as user name and password) between supplicant and authentication server. This authentication is independent of the PEPs operations.

The SI-U-SAP (User) interface is used to communicate secure user information (this includes TCP headers and user data). Also the user authentication messages use the SI-U-SAP interface. However, the key management information is passed through the SI-C-SAP (Control) interface, because this function is normally performed during connection establishment phase.

Both authentication server and the BSM Network manager communicate with the BSM Network Control Centre (NCC) regarding security and authorization. These interactions are not shown here in order to simplify the diagram. The Security association identity (SID) can be used in all security management message exchanges.

Thus link layer security can work seamless with TCP and application layer PEPs and provide strong access control to the satellite network resources.

6 Analysis of PEP deployment impact on current and future BSM architecture

In clauses 4 and 5 the BSM PEP architecture, transport and application layer techniques, cross layer and security issues presented. Thus, PEP interworking with BSM architecture might require updating the following BSM technical specification:

- TS 102 463 [i.5] and TS 102 464 [i.6] for Interworking with BSM IntServ and DiffServ QoS.
- TS 102 465 [i.1] for interworking with BSM Security Functional Architecture.

In addition, there are some possible future work and advanced topics related to PEPs that can be added to the BSM future roadmap such as:

- Definition of converged PEP architecture for satellite (both FSS and MSS) and alignment with terrestrial PEP architectures (e.g. 3GPP). This may require the specification of a new distributed PEP protocol with adjustable parameters for satellite and terrestrial fixed/mobile networks.
- QoS aware PEPs, where current PEP products only use proprietary signalling between the PEP and the Satellite terminal. Therefore standards based protocols and mechanisms can be defined for:
  - Capacity and QoS management interface between PEP and ST/GW.
  - Simple and workable cross layer techniques to support A-PEPs and T-PEPs.
- Multicast PEPs to provide A-PEP and/or T-PEP enhancement via satellite multicast connection.
- Technical Reports on the following issues:
  - End-to-end problems with PEPs e.g. IPsec and certain applications with dynamic port number.
  - Interworking between PEPs and Next Generation Satellite Network elements, using the Session Initiation Protocol (SIP) as a signalling protocol.
  - Compression techniques for transport header and payload with focus on UDP PEPs.

Annex A: End-to-end options and improvement techniques

In this clause, we survey the most relevant available options in standard TCP and HTTP for minimizing satellite link under-utilization. The focus is on mechanisms that can be configured on operating systems and web browsers/servers that have already been deployed [i.3].

A.1 TCP end-to-end techniques

Maximum Segment Size (MSS)

The MSS value can fit into the IP MTU in order to avoid fragmentation at the IP layer. Therefore, the choice of an appropriate MSS depends on the MTU. However, the MSS may interact with some TCP mechanisms. For example, since the congestion window is counted in units of segments, high MSS values allow TCP congestion windows to increase faster.

Maximum transmission window

The maximum transmission window of a TCP sender is defined by the Receive WINdow (RWIN) advertised by the receiver in each TCP segment. The maximum transmission window can be at least equal to the bandwidth delay product (BDP) of a path to fully utilize its capacity. In GEO systems the BDP is usually very high because of long RTTs. As a result, it is absolutely necessary to adopt larger RWIN on clients. A window size of at least four segments is needed for applying Fast Retransmit and Fast Recovery mechanisms.

Selective acknowledgements (SACKs)

This option is widely implemented and is useful when multiple losses occur in a single window, allowing the recovery of the lost segments in a single RTT. It may also help to improve TCP throughput since retransmissions may be performed earlier than the Retransmission Time Out (RTO) expiration.

Timestamps Option

This option enables a more accurate RTO calculation than by default. This should help to avoid spurious retransmissions and the activation of congestion avoidance mechanisms. However, timestamps add a 12-byte overhead to TCP headers. Also it is required for high sending rates, to provide protection from wrapped sequence numbers.

References [i.25] and [i.26] describe the defined set of TCP extensions and slow start.

A.2 Application layer end-to-end techniques

Web browsing performance over satellites is highly dependent on the HTTP version and the options used, which determine how TCP services are used by HTTP. In this clause, we focus on the main tuneable features and parameters of HTTP: versionsHTTP1.0 and HTTP 1.1, the keep-alive option, use of parallel persistent connections, and pipelining. We also consider the impact of object sizes on performance and content compression [i.22].

HTTP 1.0

HTTP 1.0 is the first version of the HTTP protocol. In this case, the client opens a TCP connection for each object on the page. Once a connection is open, the client sends a GET message requesting the object to be downloaded. Therefore, a three-way handshake procedure for establishing connection, followed by a Slow Start phase, is performed for each object. Moreover, every GET request adds roughly one RTT to the total web page download time. This RTT wasting pattern results in a significant decrease in data transmission efficiency.

HTTP 1.1

Current browsers and servers mainly implement HTTP 1.1 by default. This version addresses several issues that were somewhat neglected in HTTP 1.0 and contemplates a number of options that may reduce HTTP- and TCP-induced transmission stalls in a long RTT link:

1) Persistent connections: In HTTP 1.1, one or more objects may be downloaded using the same TCP connection, therefore reducing the number of connection establishment and SlowStart phases. Some HTTP 1.0 clients support persistent connections by using the Keep-Alive option.

2) Parallel connections: HTTP 1.1 makes it possible to use a number of parallel connections in a web page download. This option may result in significant benefits over satellites since data may be transmitted through one connection, thus taking advantage of periods of inactivity in other concurrent connections.

3) Pipelining: Pipelining is an option defined with the goal of avoiding the HTTP stop-and-wait effect derived from the object request-reply sequence within a TCP connection. When pipelining is used, a client may perform a GET request before the previous object is completely received. Ideally, pipelining should lead to a higher number of outstanding GET requests than object downloads at any time during the connection. Therefore, in such cases, HTTP-induced waiting times would be minimized. Note that pipelining may be used regardless of the number of parallel TCP connections. Unfortunately, many web servers and proxies do not respect the pipelining feature. For this reason, many browsers disable this option by default or do not support it.

HTTP 1.1 and server pre-emptive FIN

Some web servers use a technique based on sending a pre-emptive TCP FIN segment that imposes a premature end of a TCP connection in order to reduce server load, which improves server performance. Therefore even if HTTP 1.1 is used, in such cases, the web download behaviour may be close to that of an HTTP 1.0 communication. Hence, HTTP 1.0 performance is still relevant for worst case analysis.

Web page object sizes

Web pages are not built to keep the pipe full of data. The web browser client typically generates GET messages as Uniform Resource Locator (URLs) objects appear on the downloaded HTML text. The client does not know in advance the size of each object. This is a problem, since consecutive GET messages may require a series of very small objects. This pattern may lead to idle periods that could be avoided if small and large objects were interleaved with the web page download, minimizing transmission stalls. Thus, shorter download times could be achieved if the URL objects in the HTML text were suitably located.

Content compression

One way of improving performance over links with relatively low data rates is by performing content compression in order to reduce the amount of application data to be transmitted, and the corresponding lower layer header overhead. There are two main kinds of compression techniques depending on the nature of the content:

i) lossless compression, which applies to text- or binary-based content (such as HTML and Javascript); and

ii) lossy compression, which can be used for image-based content (JPEG, GIF, bitmap, etc.).

In the latter case, image quality is sacrificed in order to obtain smaller sized image objects. HTTP 1.0 web servers support lossless compression coding techniques by using default formats such as gzip and compress. Additional lossless encoding formats such as deflate are supported in HTTP 1.1. Note that negligible gain may be achieved when lossless compression is applied to content already compressed (e.g. image formats). Lossy compression techniques can be used by PEPs, where PEPs can do transcoding and reduce the quality of images, etc. However, this needs to be under the control of the operator/user to ensure this matches the expectations of the user.

Annex B: PEPs options and improvement techniques

B.1 PEPs types and classifications

There are many types of Performance Enhancing Proxies. Different types of PEPs are used in different environments to overcome different link characteristics which affect protocol performance [i.3].

Layering

In principle, a PEP implementation may function at any protocol layer but typically it functions at one or two layers only. In the present document, we focus on PEP implementations that function at the transport layer or at the application layer as such PEPs are most commonly used to enhance performance over links with problematic characteristics.

Transport Layer PEPs

Transport layer PEPs operate at the transport level. They may be aware of the type of application being carried by the transport layer but, at most, only use this information to influence their behaviour with respect to the transport protocol; they do not modify the application protocol in any way, but let the application protocol operate end-to-end.

Most transport layer PEP implementations interact with TCP. Such an implementation is called a TCP Performance Enhancing Proxy (T-PEP). The term TCP spoofing is sometimes used synonymously for TCP PEP functionality. However, the term TCP spoofing more accurately describes the characteristic of intercepting a TCP connection in the middle and terminating the connection as if the interceptor is the intended destination. Most TCP PEP implementations use TCP spoofing but some do not.

Application Layer PEPs

Some application protocols employ extraneous round trips, overly verbose headers and/or inefficient header encoding which may have a significant impact on performance, in particular, with long delay and slow satellite links. This unnecessary overhead can be reduced, in general or for a particular type of link, by using an application layer PEP in an intermediate node.

Application layer PEPs operate above the transport layer. An example of application layer proxy is a Web cache. Application layer PEPs, can be implemented to improve application protocol as well as transport layer performance with respect to a particular application being used with a particular type of link. An application layer PEP may have the same functionality as the corresponding regular proxy for the same application but extended with link-specific optimizations of the application protocol operation.

Distribution

A PEP implementation may be integrated, i.e. it comprises a single PEP component implemented within a single node, or distributed, i.e. it comprises two or more PEP components, typically implemented in multiple nodes. An integrated PEP implementation represents a single point at which performance enhancement is applied. A distributed PEP implementation is generally used to surround a particular link for which performance enhancement is desired. For example, a PEP implementation for a satellite connection may be distributed between two PEPs located at each end of the satellite link. A typical example of a distributed PEP is the Satlabs I-PEP [i.4] and an example of integrated PEP is the PEPsal solution (annex D).

Implementation Symmetry

A PEP implementation may be symmetric or asymmetric. Symmetric PEPs use identical behaviour in both directions, i.e. the actions taken by the PEP occur independent from which interface a packet is received. Asymmetric PEPs operate differently in each direction. The direction can be defined in terms of the link (e.g. from a central site to a remote site) or in terms of protocol traffic (e.g. the direction of TCP data flow, often called the TCP data channel, or the direction of TCP ACK flow, often called the TCP ACK channel). An asymmetric PEP implementation is generally used at a point where the characteristics of the links on each side of the PEP differ or with asymmetric protocol traffic.

Whether a PEP implementation is symmetric or asymmetric is independent of whether the PEP implementation is integrated or distributed. In other words, a distributed PEP implementation might operate symmetrically at each end of a link (i.e. the two PEPs function identically). On the other hand, a distributed PEP implementation might operate asymmetrically, with a different PEP implementation at each end of the link.

Split Connections

A split connection TCP implementation terminates the TCP connection received from an end system and establishes a corresponding TCP connection to the other end system. In a distributed PEP implementation, this is typically done to allow the use of a third connection between two PEPs optimized for the link (for example the I-PEP protocol [i.4] that is recommended by Satlabs). This might be a TCP connection optimized for the link or it might be another protocol, for example, a proprietary protocol running on top of UDP. Also, the distributed implementation might use a separate connection between the proxies for each TCP connection or it might multiplex the data from multiple TCP connections across a single connection between the PEPs. In an integrated PEP split connection TCP implementation, the PEP again terminates the connection from one end system and originates a separate connection to the other end system.

Application layer proxies for TCP-based applications are split connection TCP implementations with end systems using PEPs as a service related to a particular application. Therefore, all transport (TCP) layer enhancements that are available with split connection TCP implementations can also be employed with application layer PEPs in conjunction with application layer enhancements.

Transparency

PEPs may operate totally transparently to the end systems, transport endpoints, and/or applications involved (in a connection), requiring no modifications to the end systems, transport endpoints, or applications.

On the other hand, a PEP implementation may require modifications to both ends in order to be used. In between, a PEP implementation may require modifications to only one of the ends involved. Either of these kind of PEP implementations is non-transparent, at least to the layer requiring modification.

B.2 TCP PEP (T-PEP) techniques

An obvious key characteristic of a PEP implementation is the mechanism(s) it uses to improve performance. Some examples of PEP mechanisms are described in the following subsections. A PEP implementation might implement more than one of these mechanisms.

TCP ACK Handling

Many TCP PEP implementations are based on TCP ACK manipulation. The handling of TCP acknowledgments can differ significantly between different TCP PEP implementations. Many implementations combine some of these mechanisms and possibly employ some additional mechanisms as well.

- TCP ACK Spacing: In environments where ACKs tend to bunch together, ACK spacing is used to smooth out the flow of TCP acknowledgments traversing a link. This improves performance by eliminating bursts of TCP data segments that the TCP sender would send due to back-to-back arriving TCP acknowledgments.

- Local TCP Acknowledgements: In some PEP implementations, TCP data segments received by the PEP are locally acknowledged by the PEP. This is very useful over network paths with a large bandwidth*delay product (e.g. satellites) as it speeds up TCP slow start and allows the sending TCP to quickly open up its congestion window.

- Local TCP Retransmissions: A TCP PEP may locally retransmit data segments lost on the path between the TCP PEP and the receiving end system, thus aiming at faster recovery from lost data. In order to achieve this, the TCP PEP may use acknowledgments arriving from the end system that receives the TCP data segments, along with appropriate timeouts, to determine when to locally retransmit lost data. TCP PEPs sending local acknowledgments to the sending end system are required to employ local retransmissions towards the receiving end system.

- TCP ACK Filtering and Reconstruction: On paths with highly asymmetric bandwidth the TCP ACKs flowing in the low-speed direction may get congested if the asymmetry ratio is high enough. The ACK filtering and reconstruction mechanism addresses this by filtering the ACKs on one side of the link and reconstructing the deleted ACKs on the other side of the link. While TCP ACKs are cumulative, reconstruction of the deleted ACKs is necessary to maintain the TCP ACK clock of the sender.

Tunneling

A Performance Enhancing Proxy may encapsulate messages in a tunnel to carry the messages across a particular link or to force messages to traverse a particular path. A PEP at the other end of the encapsulation tunnel removes the tunnel wrappers before final delivery to the receiving end system. A tunnel might be used by a distributed split connection TCP implementation as the means for carrying the connection between the distributed PEPs. A tunnel might also be used to support forcing TCP connections which use asymmetric routing to go through the end points of a distributed PEP implementation.

Compression

Many PEP implementations include support for one or more forms of compression. In some PEP implementations, compression may even be the only mechanism used for performance improvement. Compression reduces the number of bytes which need to be sent across a link. This is useful in general and can be very important for bandwidth limited links. Benefits of using compression include improved link efficiency and higher effective link utilization, reduced latency and improved interactive response time, decreased overhead and reduced packet loss rate over lossy links.

Handling Periods of Link Disconnection with TCP

During link disconnection or link outage periods, a TCP sender does not receive the expected acknowledgments. Upon expiration of the retransmit timer, this causes TCP to close its congestion window with all of the related drawbacks. A TCP PEP may monitor the traffic coming from the TCP sender towards the TCP receiver behind the disconnected link. The TCP PEP retains the last ACK, so that it can shut down the TCP sender's window by sending the last ACK with a window set to zero. Thus, the TCP sender will go into persist mode.

To make this work in both directions with an integrated TCP PEP implementation, the TCP receiver behind the disconnected link can be aware of the current state of the connection and, in the event of a disconnection, it can be capable of freezing all timers. Another possibility is that the disconnected link is surrounded by a distributed PEP pair.

Priority-based Multiplexing

Implementing priority-based multiplexing of data over a slow and expensive link may significantly improve the performance and usability of the link for selected applications or connections. A user behind a slow link would experience the link more feasible to use in case of simultaneous data transfers, if urgent data transfers (e.g. interactive connections) could have shorter response time (better performance) than less urgent background transfers. If the interactive connections transmit enough data to keep the slow link fully utilized, it might be necessary to fully suspend the background transfers for awhile to ensure timely delivery for the interactive connections.

This kind of operation can be controlled in conjunction with a split connection TCP PEP by assigning different priorities for different connections (or applications). A split connection PEP implementation allows the PEP in an intermediate node to delay the data delivery of a lower-priority TCP flow for an unlimited period of time by simply rescheduling the order in which it forwards data of different flows to the destination host behind the slow link. This does not have a negative impact on the delayed TCP flow as normal TCP flow control takes care of suspending the flow between the TCP sender and the PEP.

This can further be assisted, if the protocol stacks on both sides of the slow link implement priority based scheduling of connections. With such a PEP implementation, along with user-controlled priorities, the user can assign higher priority for selected interactive connection(s) and have much shorter response time for the selected connection(s), even if there are simultaneous low priority bulk data transfers which in regular end-to-end operation would otherwise eat the available bandwidth of the slow link almost completely. These low priority bulk data transfers would then proceed during the idle periods of interactive connections, allowing the user to keep the satellite link fully utilized.

B.3 Application layer PEP (A-PEP) techniques

Extended caching

This solution implements a client side cache that replaces the browser's permanent cache [i.22]. The client communicates through a custom protocol with another caching entity close to the base station (i.e. PEP server). Fingerprints of objects mapped to URLs are used to determine whether or not objects have changed with respect to previous web page downloads. When a URL mapping expires on the client side cache, the client PEP asks the PEP server to check whether a more recent mapping exists. If the object has not changed, then the PEP server simply retransmits the URL to fingerprint mapping to the client PEP. Thus, unnecessary object downloading through the wireless link is avoided, reducing web page download times and bandwidth requirements.

Compression and content format adaptation

A PEP may apply compression techniques (clause A.2) in order to reduce the total amount of data sent to the satellite link. In particular, lossy compression, which is not supported by default by servers, is a good candidate technique to be used by a PEP. Note that the achievable compression degree for lossless compression may be statistically characterized, while lossy compression gain is not known a priori and may be tuned by the user (when possible) or selected by the proxy administrator.

Additionally, content format adaptation mechanisms may be supported by a PEP in order to deliver information in a suitable format, depending on the device's hardware features and also on what the user wants to display on his/her terminal. Efficient formats are needed in order to save bandwidth and device power consumption.

Delta encoding

Delta encoding is a technique based on sending differences between new and old versions of a document. This strategy avoids the need for downloading a web page every time a client wishes to access its content. Thus, it is generally successful since updated documents are often significantly similar to their predecessors.

Client prefetching

Client prefetching allows early fetches to be performed on web pages linked to pages that have already been read. The rationale behind this mechanism is as follows: Inactive link periods may be used so that when a user clicks on a link on a displayed web page, the new one appears instantaneously on the client browser.

Multilayer PEPs

Significant performance enhancement can be achieved by combining optimizations at more than one layer. A number of PEPs provide multilayer optimization. Mowgli [i.22] uses a custom transport protocol for the wireless link, together with application layer techniques such as compression and content format adaptation.

DNS- and URL-rewriting

Web page content may be distributed among a number of servers. In such cases, several DNS lookups can be performed by a client. DNS re-writing is a technique whereby a proxy may force the client to point to one single proxy server where the web content is assumed to be already downloaded, thus reducing the number of DNS lookups and TCP connection establishments. Similarly, uniform resource locators URLs on a web page can be replaced by the IP address of a proxy server. Note that more than one proxy server may be used. Using the appropriate number of proxy servers may improve performance, leading to behaviour analogous to that of using an optimal number of simultaneous TCP connections.

Annex C: Summary of techniques used by PEP manufacturers

A Questionnaire was sent to several PEP manufacturers and here is summary of the most popular techniques types and techniques.

Regarding the PEP characteristics described in clause B.1, is the survey summary:

Table C.1: Summary of PEP Characteristics

PEP Characteristics	Answer
Transport, Application or Both?	All products implement TCP PEPs in products which carry IP over satellite. Products which are also used for Internet and/or intranet access also include an HTTP PEP.
Distributed or Centralized?	For most manufacturers, both TCP PEP and HTTP PEP are distributed.
Symmetric or Asymmetric?	TCP PEP is symmetric. HTTP PEP is asymmetric.
Split Connection?	Yes.
Transparent to Transport? (as defined in RFC 3135 [i.3])	Yes.
Transparent to Application? (as defined in RFC 3135 [i.3])	Yes. In general, the TCP PEP can always be used without the HTTP PEP. In addition, the HTTP PEP is transparent to the TCP PEP and can be used without the TCP PEP, when useful.

Regarding the PEP mechanisms described in clauses B.2 and B.3, is the survey summary:

Table C.2: Summary of PEP Mechanisms

PEP Mechanisms	Answer
TCP ACK Spacing?	No.
Local TCP Acknowledgements?	Yes.
Local TCP Retransmissions?	Yes.
TCP ACK Filtering and Reconstruction?	Most products use a split connection nature for both TCP PEPs and HTTP PEPs. Thus they all have optimized ACKing strategy over the satellite link.
Tunneling?	Most products do not use IP tunnels. However, the backbone protocol used to multiplex TCP connections when traversing the satellite link could be considered a tunnel in that it encapsulates the TCP data and there is a many to one mapping of TCP connections to backbone connection.
Compression?	Most products support TCP PEP and HTTP PEP support header and payload compression. Some products support IP compression for non-TCP protocols.
Handling Link Disconnection?	Some products include this feature and some do not include it. However, in general the HTTP PEP is not aware of link disconnection, while TCP PEP will keep the TCP connections up for a duration depending on the satellite network configuration.
Priority-based Multiplexing?	Most TCP PEP products have QoS mechanisms including prioritization such as using Weighted Fair Queuing algorithm. Some products even have support for the use of multiple parallel backbone connections in order to prevent "higher" QoS TCP flows from being impacted by "lower" QoS TCP flows.
If HTTP PEP, do you use Caching, Parse/pre-fetch, or other mechanisms	All HTTP PEP products support the parse/pre-fetch approach. This includes caching web objects on the PEP client and prefetching web objects on the PEP server.

Annex D:
Examples of distributed and integrated PEPs

D.1 Application layer PEP: Hughes HPEP

Since web browsing is supported by a client/server model, an HTTP PEP implementation requires an asymmetric architecture. As illustrated in figure D.1, Hughes' HTTP PEP (HPEP) implementation uses a distributed, asymmetric architecture with a PEP client on the web browser side (the HPEP downstream proxy) and PEP server on the web server side (the HPEP upstream proxy).

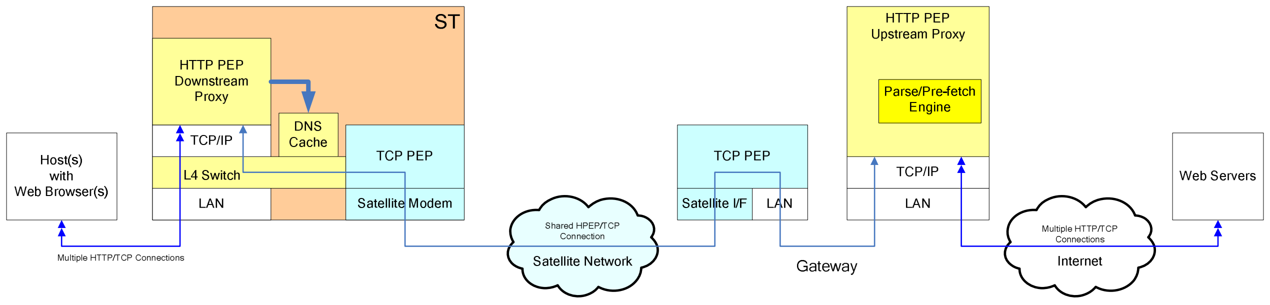

Figure D.1: HPEP architecture

Architecturally, HPEP is positioned similarly to an HTTP proxy. (In fact, transparent operation can optionally be disabled and Hughes HPEP can be pointed at as a proxy by the web browser.) For transparent operation, the PEP can be placed into the traffic path. The most convenient way to do this is to include the PEP in the default router of the hosts at the remote site, in this case, the satellite terminal.

As IP packets are sent by a web browser, they are examined by a Layer 4 switch in the PEP. When the L4 switch recognizes HTTP traffic, the relevant IP packets are diverted to the HTTP PEP downstream proxy. Non-HTTP packets are simply relayed towards the spacelink, where, in this case, they will be examined and processed by a TCP PEP. HTTP packets sent by the HPEP proxy are forwarded by the L4 switch to either the web browser or the HPEP upstream proxy based on their destination addresses. HTTP packets forwarded to the HPEP upstream proxy are treated like other traffic, including being processed by the TCP PEP. In the Hughes implementation, the TCP PEP implementation is independent from the HTTP PEP implementation. The HTTP PEP works with or without the presence of the TCP PEP.

At the gateway, the gateway TCP PEP processes the packets and then forwards them as appropriate. HTTP packets sent to the HPEP upstream proxy are addressed to the proxy. Other packets (e.g. non-HTTP traffic) bypass the HPEP upstream proxy, being forwarded directly to the appropriate Internet or intranet router by the TCP PEP. Because the downstream proxy already transparently intercepts traffic of interest, the upstream proxy does not need to do so and, thus, does not need to be in line with all of the traffic. Requests sent by the HPEP upstream proxy to web servers are sent with the proxy's IP address as source. Thus, again, there is no need for the proxy to be in line with all traffic. When the upstream proxy receives responses from web servers, they are sent to the relevant HPEP downstream proxy's IP address (via the TCP PEP, in this case). The downstream proxy then forwards the response to the web browser.

The HPEP downstream and upstream proxies use a semi-permanent TCP connection to communicate with each other. The connection is opened when the first web browser request is received and stays open until no web browser requests have been seen for awhile. (The timeout is on the order of several tens of minutes.) In cases where upstream proxy resources do not need to be conserved, the TCP connection can actually be made permanent. The two proxies use a proprietary protocol to multiplex all HTTP messages together. The protocol is similar to a tunnel but includes signalling between the two proxies. The signalling is mainly related to parse/pre-fetch operation (described below) but is also used to support multi-threading, i.e. the HTTP messages associated with different web browser TCP connections are forwarded independently to prevent blocking.

The HPEP "tunnel" provides some amount of performance improvement by itself. But, the primary mechanism for improving web browsing performance is provided by the operation of the parse/pre-fetch engine in the HPEP upstream proxy. When a requested web page is received from a web servers, before forwarding it to the downstream proxy, the upstream proxy parses the web page, looking for embedded objects which will need to be requested by the web browser when it parses the web page. The upstream proxy then starts pre-fetching, in parallel, these web objects and forwarding them to the downstream proxy so that they will already be waiting when the web browser requests them. Since not every embedded object can be pre-fetched, the signalling between the proxies includes the upstream proxy indicating to the downlink proxy exactly which objects are being pre-fetched. If a web browser asks for an object which is being pre-fetched but which has not yet reached the downstream proxy, the signalling lets the downstream proxy know that it is not necessary to forward that particular request through to the gateway. Thus, the parse/pre-fetch operation, in addition to significantly improving web page download time by anticipating which objects need to be fetched, also significantly reduces the amount of HTTP traffic which actually needs to be sent in the inbound direction.

As illustrated in figure D.1, the HPEP implementation also makes use of a DNS cache to improve performance. Web pages are often made up of objects retrieved from different servers. When HPEP is being used transparently, the web browser needs to do a DNS query for each new server accessed. Since the HPEP upstream proxy is actually getting all of the objects on behalf of the browser, the proxy also needs to make the same set of DNS queries. This fact is leveraged by having the upstream proxy forward the DNS responses it receives to the downstream proxy where they are stored in a local DNS cache. When a DNS query is seen by the Layer 4 switch in the downstream proxy, the L4 switch forwards the query to the DNS cache. If the DNS information is available (e.g. because another web browser already accessed this site or the HPEP upstream proxy has already accessed this site to pre-fetch an object), the DNS query is responded to locally. Otherwise, the DNS cache relays the DNS query towards the actual DNS server.

D.2 Distributed TCP PEP: Satlabs I-PEP

As shown in the I-PEP specification document [i.4], figure D.2 depicts the overall system scenario: Two satellite service providers using equipment and software from different manufacturers offer DVB-RCS services for numerous customers - who in turn use a variety of receiver equipment and software. To enable such a heterogeneous scenario, the data formats and basic rules for information exchange across the satellite communication link need to be standardized.

The service providers are running servers from two different vendors, colour-coded in yellow and orange. The customers utilize equipment from three different manufacturers, colour-coded in yellow, orange, and green). All servers and clients can communicate at the level defined in this protocol specification (green arrows). Beyond this basic interoperability for efficient satellite communication, solutions from a single vendor on the client and server side can provide additional and/or further optimized services to their customers, benefiting from the vendor-specific extensibility of the present protocol specification (blue arrows). Finally, with this protocol specification strictly confined to the air interface, vendors may provide even more product differentiation by intelligent proprietary algorithms on both the client and the server platforms: as local configuration, system integration, operation, and algorithms are left open.

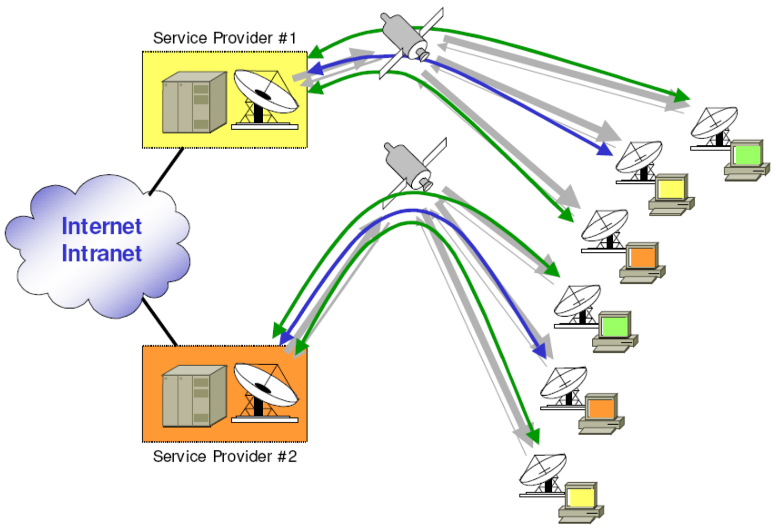

Figure D.2: General I-PEP overview

The I-PEP functional architecture assumes a split-connection approach with essentially two different roles of interest as depicted in figure D.3. The logical subdivision assumes an identifiable I-PEP server and a client capable of supporting the I-PEP protocol defined in the present document. Associated with the I-PEP client and server are an application client and server, respectively. The two I-PEP entities communicate via a DVB-RCS link, communication between application and I-PEP client on one side and application and I-PEP server on the other may be carried out via arbitrary local or wide-area networks using standard IP-based communication protocols such as TCP. In practice, application and I-PEP are likely to be co-located in the same LAN if not on the same physical device while application and I-PEP server are often likely to communicate across the wide area Internet.

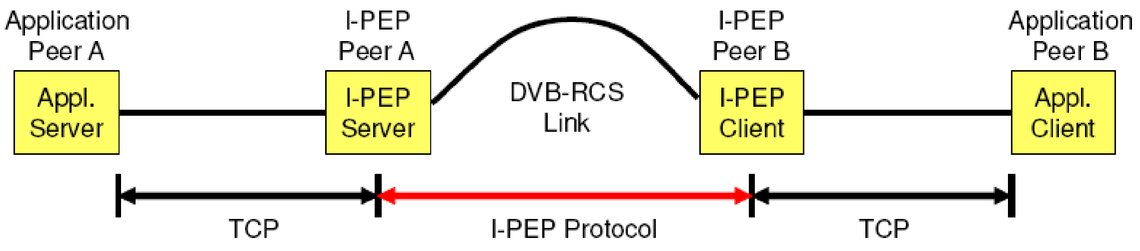

Figure D.3: Basic I-PEP Components

Note that the above distinction between client and server is artificially introduced to simplify distinguishing these entities in the description. The server parts are always considered to be located on the hub side of the DVB-RCS link, the respective clients on the DVB-RCS terminal side in the transparent star scenario. The I-PEP protocol specification itself is symmetric in that both peers may have equal capabilities, may both invoke the same actions, etc. Similarly, the two peer applications (peers A and B) may take arbitrary roles: they can act both as servers or clients or as equal peers.

The I-PEP specifications follows the above notion of client and server side for two reasons:

1) to follow the (most) common deployment of DVB-RCS; and

2) to reflect the asymmetric nature that may be built into add-on protocols (such as service announcements) as defined in the context of the session (and management) layer in support of service location and load balancing.

The I-PEP transport protocol is used to efficiently carry data between two I-PEP peers. To avoid defining a completely new protocol from scratch and thereby to allow for achieving interoperability even with non-PEP peers, the standard reliable transport protocol in the Internet is taken as a basis: TCP (RFC 793 [i.33], RFC 1122 [i.34], RFC 2581 [i.26], among others). As it has been repeatedly proven, TCP is not well suited for communication across network paths that:

a) exhibit a high bandwidth × delay product; or may

b) show a significant fraction of non-congestion related packet losses.

Satellite links by definition fall into category a) and, depending on set-up and environmental conditions, may also show characteristics of b).

These - well known and documented - deficiencies of TCP have been addressed in the past in the IETF (in general) as well as by other (standardization) bodies. While the IETF has produced numerous general purpose TCP extensions - e.g. to support large windows and selective acknowledgements, among many others - other groups have focused on dedicated environments, such as satellite communications investigated by the Consultative Committee for Space Data Systems (CCSDS) that has produced an extension set to and profile of TCP termed SCPS-TP in 1999. SCPS-TP addresses parts of the issues with TCP for satellites and maintains backward compatibility with vanilla TCP.

The I-PEP specification builds on top of TCP, SCPS-TP and TCP extensions, the extensions were current at the time I-PEP was worked on, and created a protocol suitable for satellite communication via DVB-RCS while maintaining backward compatibility with TCP and SCPS-TP. Taking SCPS-TP as a starting point, the I-PEP eliminates options that were not considered essential for reliable communications via DVB-RCS (including those that can be easily accomplished independently, e.g. by just sending UDP datagrams), creating a specific I-PEP profile from the available SCPS-TP capabilities. For example, the best effort transport service from SCPS-TP is not needed, congestion control rules need to be adapted, and several parameters and options deserve revisiting. Functions that are deemed relevant for satellite communications but not yet included in SCPS-TP are added in the present document. The extensions to the transport protocol are kept to a minimum and other (e.g. session related) functionality is pushed into a separate protocol orthogonal to the transport.

D.2.1 TCP Noordwijk

TCP Noordwijk (TCPN), [i.23] and [i.24], is a transport intended to be TCP-friendly transport and specifically designed to operate with I-PEP over a DVB-RCS link and tailored to efficiently transmit short, but frequent, amount of data (such as web traffic), without compromising reliability and scalability provided by standard TCP [i.25]. Compatibility with transport protocols of different vendors I-PEP relies on TCPN sender-only modifications to SCPS-TP reference implementation [i.4] and[i.27]. On the other hand, TCPN is able to adapt transmission rate to the available bandwidth in the bottleneck link with the twofold effect of avoiding network congestion collapse and not limiting throughput of competing standard TCP connections.

Mainly, TCPN changes the traditional "window-based" transmission paradigm with a "burst-based" paradigm. Packet transmission is then regulated by two variables: the burst size (BURST), which is the number of packets to send at once, and the burst transmission interval (TX_TIMER), which is the time interval between two consecutive burst transmissions. Both BURST and TX_TIMER values are constantly updated during the protocol operations according to available capacity assessments made through ACK-based measurements.

At the beginning of connection, TCPN transmits by using default settings, $BURST_0$ and TX_TIMER_0, which are tuned to optimize transfer of HTTP objects: $BURST_0$ is chosen proportional to the dimension of most of the typical web objects and at the same time not too big to affect performance of concurrent flows, while TX_TIMER_0 is chosen such that the initial transmission rate ($BURST_0/TX\_TIMER_0$) satisfies a pre-determined rate requirements. This initial "blind" phase lasts at least 1 Round-Trip Time (RTT), that is the time that TCPN sender can wait to receive first ACK trains. Upon the reception of the ACK trains, TX_TIMER is updated with the aim to optimize system capacity utilization with a reference maximum burst value of $BURST_0$. On the other hand, actual BURST is tuned according to the assessed congestion but always kept smaller or equal than $BURST_0$. In this frame, scalability is guaranteed by two factors:

- I-PEP to I-PEP communication environment is assumed to be controlled; buffers are tailored according to the expected traffic patterns and then able to accommodate spikes in the incoming traffic (e.g. due to a large number of TCPN connections starting simultaneously).
- TCPN implements a burst rate control scheme aiming to privilege short transfers; in other words, to compensate the "blind" transmission of new connections, longer connections reduce their rate behaving as "low-priority" connections.

The overall protocol architecture is based on three functional components: an ACK-based Capacity and Congestion Estimation (ACCE), Burst Rate Control and Flow/Error Control. Figure D.4 summarizes the basic sender behaviour as well as relationships among TCPN functional components.

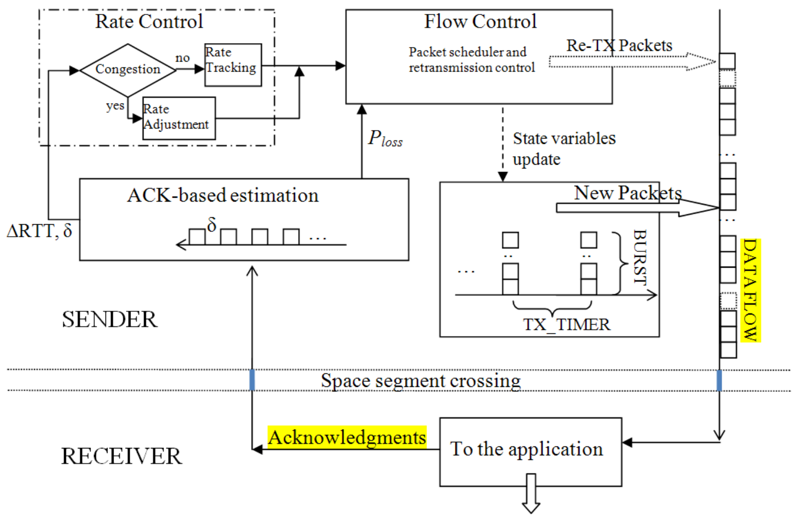

Figure D.4: TCPN sender behaviour

First, the ACK-based capacity and congestion (ACCE) component monitors ACK flow and computes a set of related statistics/information. In particular, the following values are assessed and notified to the Burst Rate Control component for each received ACK train:

- Packet pair dispersion (δ) - a burst of TCP packets transmitted back-to-back leads to a train of ACKs in line of principle received regularly spaced. Packet pair dispersion is a meter of the available bandwidth of the bottleneck link [i.28]. TCPN computes samples of packet pair dispersion by averaging them over each packet burst. In fact, a longer ACK train smoothes effects of compression/dilations concerning the single ACK pairs over a larger interval time. In addition, since the minimum and maximum allowed bandwidth are assumed to be known, two thresholds can be set to limit the range of the acceptable dispersion samples matching the system capacity accordingly. However, there is erratic behaviour when use with diffserv and active queue management, therefore some caution may be needed for its use outside PEP environment.

- RTT variations (ΔRTT) - when transmitting packets in bursts on an unloaded network with capacity allocated statically, the ACK related to the first packet will experience the minimum network RTT, while the following ones will arrive with a larger delay, each one spaced of δ. The RTT of the first packet is defined as the RTT of the burst. When either congestion occurs or access delay is greater than zero, the burst RTT is subject to variations, which are signalled as $\Delta\mathrm{RTT}_i$ to the Rate Control component.

In addition, ACCE detects possible lost packets and gets updated advertised window from ACKs. Such information is then notified to the Flow Control component.

Depending on the input parameters, Burst Rate Control component computes new BURST and TX_TIMER values. Two different algorithms regulate variations on the burst rate: Rate Tracking and Rate Adjustment. They run alternatively on the basis of the comparison of the current ΔRTT provided by ACCE, with a threshold β, which represents the maximum RTT variation that can be attributable to DAMA control loop [i.29], [i.30] and [i.31]. If $\Delta\mathrm{RTT} < \beta$, link is assumed uncongested and the Rate Tracking algorithm runs. In this case, the goal is to achieve the maximum burst rate by gradually increasing BURST up to its maximum value BURST_0. If $\Delta\mathrm{RTT} > \beta$, Rate Adjustment runs and BURST size is reduced by keeping TX_TIMER unchanged. In any case, BURST and TX_TIMER and transmits them to the Flow Control components. Flow Control component updates the effective burst size used for transmission and schedules both the transmission of new and possible retransmissions.

D.3 Integrated PEP: PEPsal

As shown in the PEPsal published papers [i.13], when the satellite link represents the last hop for Internet access, the integrated approach presents the advantage of not requiring at the end user any non standard (i.e. provider dependent), additional hardware or software modification to the receiver box. Integrated PEPs are based on TCP splitting and offer higher performance because they are able to confine the satellite impairments (long RTTs and random losses) on the second component of the split connection, where they can be counteracted by specific optimized TCPs.

The TCP integrated splitting architecture that derives from the application of the PEPsal software on the satellite gateway is shown in figure D.5. PEPsal is a TCP PEP, however it operates at three different layers (IP, TCP, and Application); hence it can be well described by analyzing the working scheme of each layer. At the network layer, PEPsal uses "netfilter" to intercept the connections that involve the satellite link (it "steals" the TCP SYN packet in the three-way handshake phase of a TCP connection). Then, it works at the transport layer, pretending to be the opposite side of the TCP connection for each of the two endpoints involved. It acts as the TCP receiver with the source, by acknowledging the incoming packets, while at the same time it sets up a new TCP connection towards the real endpoint receiver. On this second connection an enhanced TCP variant can be used. Finally, to exchange data between the two connections, it is necessary to make use of an application that directly copies data between the two sockets.

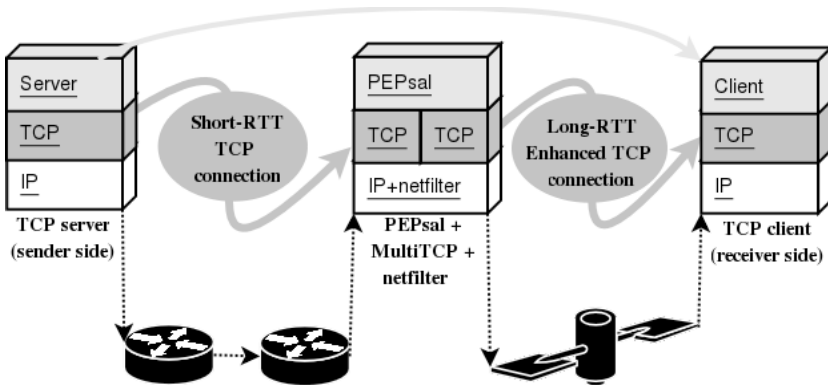

Figure D.5: PEPsal architecture (based on integrated TCP splitting approach)

PEPsal classification, takes into account the following characteristics: layering, distribution, symmetry and transparency. As far as layering is concerned, PEPsal can be considered as a TCP PEP, because it implements TCP splitting mechanism, it also uses the IP and Application layers. Considering distribution, PEPsal can be classified as an integrated PEP, since it runs only on a single box on the forward link satellite gateway. PEPsal can be either asymmetric or symmetric, depending on its network layer configuration: i.e. it can act in the forward direction (usual configuration), but also in the return one (in this case, with proper modifications on the receiver side). Finally, PEPsal is transparent in the customary asymmetric configuration. Since modifications are not required in both the connection endpoints, TCP users are unaware of the connection splitting performed at the satellite gateway.

TCP Hybla, which is the TCP variant adopted by the PEPsal architecture, was conceived by the same authors of PEPsal with the primary aim of counteracting the performance deterioration caused by the long RTTs typical of satellite connections. It consists of a set of procedures, which includes an enhancement of the standard congestion control algorithms, the mandatory adoption of the SACK policy, the use of timestamps, the adoption of Hoe's channel bandwidth estimate and the implementation of packet spacing techniques.

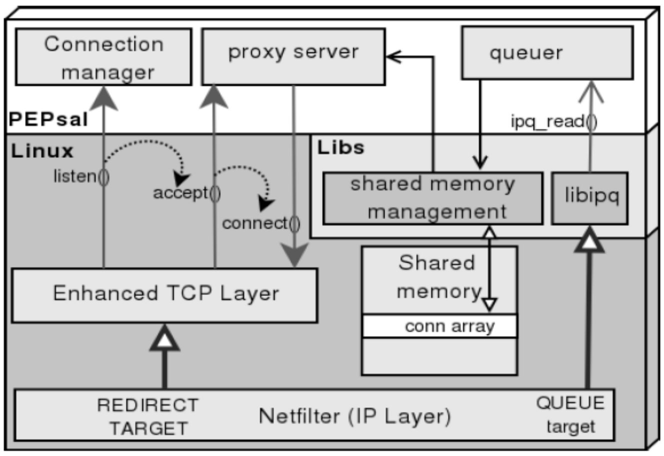

Figure D.6: The PEPsal software modules

All the software components used to implement the PEPsal architecture are reported in figure D.6, which also outlines its functioning. The bottom area ("Linux") represents the Linux kernel, which is accessed to set up the netfilter targets and to make use of a modified protocol for all TCP connections. Incoming TCP segments are mangled, so the TCP SYN segment is passed to the PEPsal user application via the ipqueue library, while the rest of the packets containing segments for that connection are redirected to local TCP port 5000. The small area on the right hand side ("Libs") shows which system libraries have been used to implement PEPsal. Shared memory is a fast and powerful IPC method, used by the PEPsal processes to share information about incoming and outgoing connections, their state and their original endpoints. A bitmap array index is used by the application to give faster access to this memory zone. Ipqueue library is commonly used to pass a whole IP packet, including network and transport headers, to user space applications. PEPsal uses it to read information from incoming TCP SYN packets, which contain no useful data except for the headers themselves, and would be normally forwarded by the gateway to their destinations to initiate new connections.

The top area ("PEPsal") represents the user space application itself, which is written in C using the two libraries described above. One process, the "queuer", constantly waits for data coming from netfilter, by blocking on the ipqueue read routine. When Linux netfilter reads incoming TCP SYN segments and copies them into a queue, the queuer annotates the information (IP addresses and TCP ports) on the two endpoints in a known zone of the shared memory. Then, the SYN packet is released and continues its path through the netfilter chain. Just after that, the SYN packet, as well as all every subsequent packet containing a segment of that connection, is redirected by netfilter to TCP port 5000, where a TCP daemon, the "connection manager" is listening for it. Another process, the "proxy server", accepts the connection and searches in the shared memory for the instance matching the source address and the TCP port of the host that has started the connection. Once the destination IP address and port have been found in the connection array, a new TCP connection is attempted towards the real destination. After establishing the two connections, the proxy starts reading from one TCP socket and writing all the data in the other one. When one of the two connections ends, its twin socket is closed and its memory zone is released.

Annex E: Example cross layer improvement for FTP data flows

Cross-layer interactions between layer 2 resource allocation and transport layer could be used to improve TCP performance in broadband satellite networks. Let us refer to Satellite Terminals (STs) using a Demand-Assignment Multiple Access (DAMA) scheme in a GEO satellite network where the Network Control Centre (NCC) is responsible to allocate transmission resources to STs depending on their capacity requests. Moreover, let us refer to a configuration with the NCC co-located with the satellite network gateway towards the Internet [i.20]. Referring to the satellite network shown in figure E.1, we consider that STs transmit File Transfer Protocol (FTP) data flows (i.e. STs are TCP senders) through the gateway towards the Internet. Let us assume that TCP traffic flow is mapped on a suitable layer 3 IP buffer than in turn corresponds to a QID at SI-SAP level with related layer 2 queue. Capacity should be distributed among competing TCP flows with a DAMA scheme centrally-coordinated by the NCC that takes into account aggregated goodput and fairness among flows. The special interest on layer 2 queues is related to the fact that in DAMA schemes the allocation of transmission resources is depending on the layer 2 queue occupancy (or the layer 2 queue arrival rate) so that the TCP goodput performance depends on how fast the resource allocation is able to track the TCP injection of data in this layer 2 buffer.

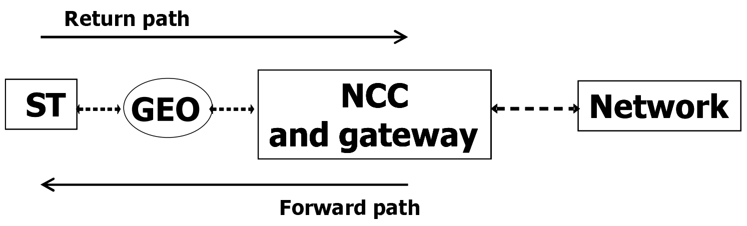

Figure E.1: Envisaged network architecture with NCC co-located with the gateway (NCC/gateway)

Suitable transport-layer PEP functionalities are here considered at the NCC/gateway to support the TCP flows, according to an integrated PEP architecture. In particular, we envisage that the NCC at layer 2 could use an upward cross-layer signalling to notify its transport layer when the capacity available in the satellite network is close to be saturated. The transport layer of the NCC could thus signal to its peer on the ST side that there is congestion (in-band downward signalling exploits modified TCP acknowledgments, ACK*) so that the increase in traffic injection by TCP could be temporarily stopped. This cross-layer approach ('Sighted PEP') could prevent the occurrence of buffer overflows and consequent TCP timeouts, thus permitting to increase the bandwidth utilization. Hence, the NCC/gateway with Sighted PEP has to filter the ACKs (*spoofer*) on the return path in case of congestion [i.21]. The detailed description of the Sighted-PEP approach with cross-layer interactions is provided below [i.20], referring to the DVB-RCS case (Multi Frequency - Time Division Multiple Access, MF-TDMA air interface):

1) In the ST, the TCP internal state information (i.e. cwnd and TCP phase, that for the classical TCP could be Slow Start or Congestion Avoidance) is propagated from layer 4 to MAC by means of either suitably-enriched headers (in-band signalling) or a periodic primitive (out-of-band signalling using X-SAP) reporting about the state of the transport layer protocol (such primitive, not concerned with the BSM work, should be synchronized with the MF-TDMA super-frame structure and the related DAMA resource request made at layer 2. In both cases, a modification to the DVB-RCS standard may be needed; moreover, in the case of X-SAP signalling also BSM SI-SAP should be modified to support this message exchange. At layer 2, a DAMA capacity request is sent for next super-frame to the NCC on the basis of the current layer 2 buffer occupancy and the prospected TCP injection of data in the next super-frame. Since the allocation of resources arrives at the ST (at least) after a round trip propagation delay from the request, the request to be effective can be performed considering a forecast on the packets arrived in the meantime; hence, the capacity request has to be based on the expected TCP cwnd increment in the allocation time. The DAMA request to the NCC needs also to convey the cwnd value of TCP and a buffer congestion indication for the ST (a threshold scheme could be considered here, where the threshold value should be suitably optimized depending on system characteristics). This procedure is described in figure E.2.

2) The NCC receives the incoming DAMA requests, assigns the available resources in the super-frame and notifies the allocation through the Terminal Burst Time Plan (TBTP) broadcast message. The NCC may reduce the amount of resources assigned to an ST in a super-frame, if the resources the ST requested are not available; then, the NCC defines at MAC layer a corresponding limit, cwnd*, to the congestion window value (cwnd* < cwnd) for the ST. If the ST buffer is congested (threshold method), the cwnd* value is provided to the transport layer of the ST with external cross-layer signalling, as detailed at the following point #3. The ST cwnd value is thus blocked to the cwnd* one. This procedure is described in figure E.2.

3) The NCC at transport layer is a gateway towards the network and operates as a PEP in the forward path where the ACKs of the TCP flow under consideration are intercepted according to a spoofing action. If a cwnd* value is available at transport layer of a given ST, such value is included in a suitable field of a transport-layer modified ACK*. Moreover, a flag in the ACK* notifies the ST if the cwnd* option is active. A suitably modified TCP version running on the ST (sender) manages the modified ACK* with cwnd* so that TCP can set cwnd to the cwnd* value to block the normal cwnd increase (non-transparent PEP approach).

4) If the condition of resource shortage is solved, the cwnd value of the ST can be unlocked with the reception of standard ACKs that are not modified by the NCC/PEP.

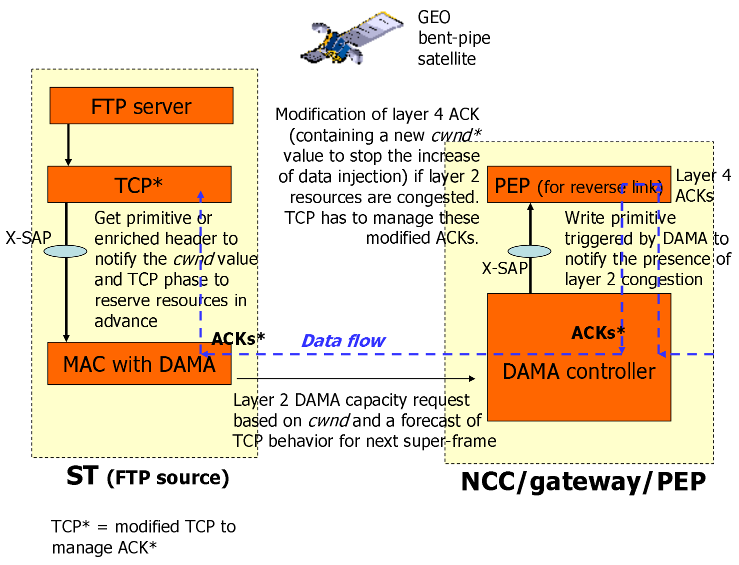
GEO
bent-pipe
satellite
FTP server
TCP*
X-SAP
Get primitive or enriched header to notify the *cwnd* value and TCP phase to reserve resources in advance
MAC with DAMA
ACKs*
Data flow
Modification of layer 4 ACK (containing a new *cwnd** value to stop the increase of data injection) if layer 2 resources are congested. TCP has to manage these modified ACKs.
PEP (for reverse link)
Layer 4 ACKs
Write primitive triggered by DAMA to notify the presence of layer 2 congestion
X-SAP
ACKs*
DAMA controller
Layer 2 DAMA capacity request based on *cwnd* and a forecast of TCP behavior for next super-frame
ST (FTP source)
NCC/gateway/PEP
TCP* = modified TCP to manage ACK*

Figure E.2: MAC-centric cross-layer scheme with PEP-spoofer (Sighted PEP) at the NCC/gateway controlling the congestion in the satellite network

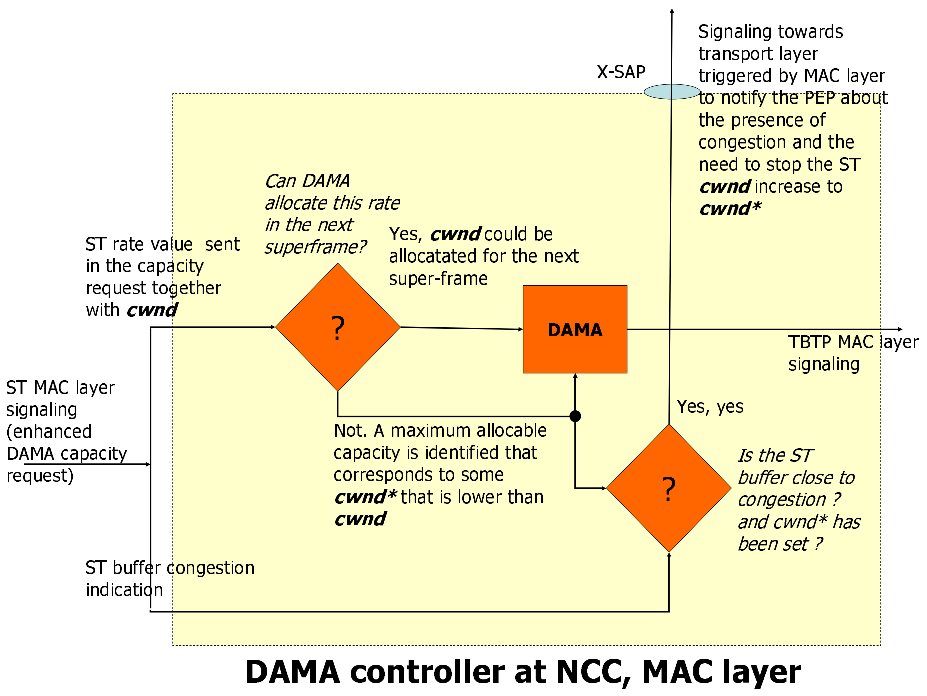
X-SAP
Signaling towards transport layer triggered by MAC layer to notify the PEP about the presence of congestion and the need to stop the ST cwnd increase to cwnd*
Can DAMA allocate this rate in the next superframe?
ST rate value sent in the capacity request together with cwnd
Yes, cwnd could be allocatated for the next super-frame
?
DAMA
TBTP MAC layer signaling
ST MAC layer signaling (enhanced DAMA capacity request)
Yes, yes
Not. A maximum allocable capacity is identified that corresponds to some cwnd* that is lower than cwnd
?
Is the ST buffer close to congestion ? and cwnd* has been set ?
ST buffer congestion indication
DAMA controller at NCC, MAC layer

Figure E.3: Details of the envisaged DAMA controller at the NCC/gateway with cross-layer exchange of signalling

The performance evaluation of this technique is described in [i.21] and the performance enhancement is due to the combined effect of the TCP traffic prediction for the DAMA request and the Sighted PEP to control the congestion in the satellite network. In particular, the Sighted PEP may avoid massive transmission buffer overflow events that would entail a TCP timeout with consequent significant reduction of the transmission bit-rate for a long time. These advantages cannot be achieved with Blind PEPs.

On the basis of the cross-layer potentialities described above and the characteristics of the Sighted PEP, the following recommendations and impacts are considered for the BSM protocol structure and SI-SAP interface definition:

- QIDSPEC at SI-SAP could be enriched with cross-layer information (i.e. TCP state coming from upper layer, and PHY modulation -downward direction- and coding conditions coming from physical layer -upward direction).
- A mechanism should be investigated to support X-SAP for the direct exchange of signalling among non-adjacent layers based on the ICMP approach, referring to the cross-layer protocol architecture in figure 7a (BSM SI-SAP should be modified to manage these primitives). As an alternative, BSM could investigate new X-SAP interfaces to support cross-layer signalling managed by an external coordinator (figures 7a and 7b) that could represent a more efficient but more complex solution.
- Layer 2 congestion notification should be an information to be propagated at higher layers in the gateway to implement a Sighted PEP with the envisaged cross-layer mechanism.

Annex F: Bibliography

- IETF RFC 3449: "TCP Performance Implications of Network Path Asymmetry".
- ETSI TR 101 984: "Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Services and architectures".
- ETSI TR 101 985: "Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia; IP over Satellite".
- ETSI TR 102 157: "Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia; IP Interworking over satellite; Performance, Availability and Quality of Service".
- Interoperable PEP (I-PEP) Transport Extensions and Session Framework for Satellite Communications: "Air Interface Specification", October 2005.

History

Document history		
V1.1.1	November 2009	Publication