

# Satellite Network Transport Architecture (SaNTA)

Erling Kristiansen

*European Space Agency, Noordwijk, The Netherlands*

Afonso Nunes

*Skysoft Portugal, Lisbon, Portugal*

José Brázio

*Instituto de Telecomunicações, Lisbon, Portugal*

André Zúquete

*Instituto de Telecomunicações / IEETA, Aveiro, Portugal*

**We presented to ICSSC 2004 a paper entitled “Re-thinking the End-to-end Paradigm - The missing protocol layer”. This paper described a novel architecture in which the TCP end-to-end transport in a heterogeneous network environment is split into a succession of separate transport instances, each operating over a homogeneous sub-network. This split architecture is augmented with a new end-to-end transport layer, inserted between the application layer and the split-transport layer. This new layer re-establishes the end-to-end context that was lost by the split. Since the presentation of that paper, it was realized that the introduction of changes to the protocol stack of all end systems, necessary in order to implement the new architecture, could be a major impediment to deployment of the system. An enhancement to the original architecture, now named SaNTA, is presented in this paper. In SaNTA, the new end-to-end layer is introduced between “satellite enhancer” (SE) entities located in association with the users’ Internet access routers rather than truly end-to-end. Support for network security has also been introduced into the SaNTA architecture. The SaNTA architecture has been implemented in a prototype that was tested both under controlled laboratory conditions and over a satellite link accessed through a WAN.**

## I. Introduction

In our ICSSC 2004 paper <sup>1</sup>“Re-thinking the End-to-end Paradigm - The missing protocol layer” we presented a novel transport architecture for TCP/IP networks comprising a satellite link. In the year that has passed since the presentation of that paper, work has progressed, further enhancing the architecture. Network security has been included as an intrinsic part of the architecture. After a brief recap of the problem we are addressing, and of the previously presented architecture, this paper will present the enhanced architecture model, which we now refer to as SaNTA (Satellite Network Transport Architecture). The current state of the SaNTA development, as well as test results and some ideas for future work will be presented.

## II. The end-to-end principle and associated problems

A concept central to the Internet, as originally conceived, is the end-to-end principle. Internet protocols are essentially operating between end systems, and the role of the network is to route IP packets from the originating end system to the destination end system. The network is not supposed to take into account, or even to know about the higher layer context to which the packets belong. Conversely, end systems are not supposed to know, or make use of any information about the network path or its current state of loading, except what they can infer from watching the behaviour of the end-to-end protocols in which they participate. (This rather simplistic view of the Internet deliberately ignores all the internal on-goings, like routing protocols and DNS because these are not visible to higher layer protocols).

The end-to-end principle works very well on a wide range of well-behaved networks that are more or less in agreement with the, mainly implicit, assumptions on which the Internet protocols were designed.

But in some recent types of links, these assumptions start to break down, and performance suffers from the Internet protocols not being tuned for these environments. Among these environments, this paper will address only networks that include satellite links. But rather similar problems arise in other environments like terrestrial mobile networks, and we think our proposed architecture may be applicable to such environments as well, though this has not been investigated in detail.

The focus of this paper is on TCP, though compatibility with other transport protocols will be discussed briefly as well.

Long end-to-end latency influences TCP performance essentially in two ways: Slow start, the exponential increase of sending rate at the beginning of a connection, takes a long time due to the long round trip delay. And TCP's congestion control algorithm reacts slowly, sometimes leading to many packets being lost before congestion control has reacted to an overload condition.

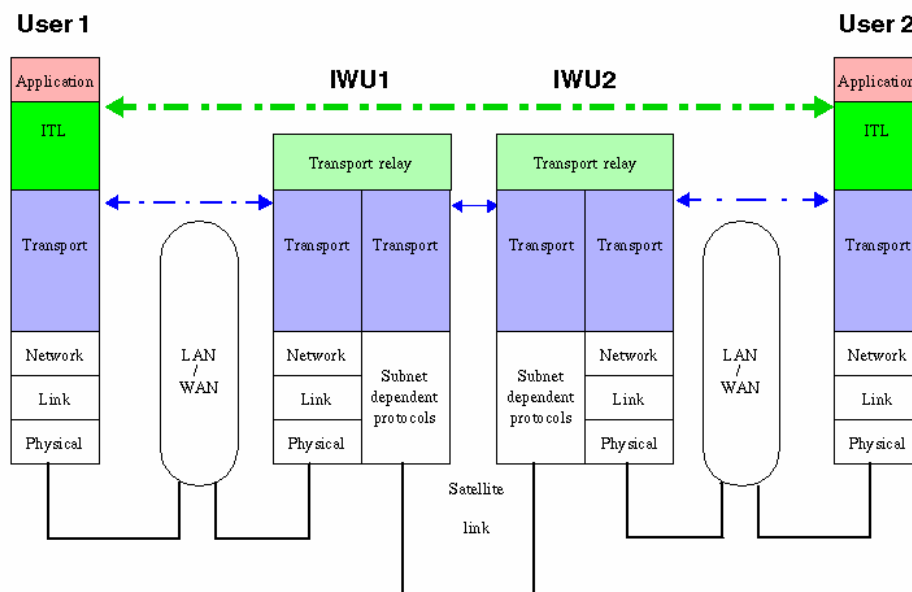
TCP assumes that all packet losses are due to congestion, and slows down when packet loss is detected. In some satellite links, packet error rate may be considerable; a packet lost due to link errors makes TCP slow down unnecessarily and is sometimes counterproductive.

Long, fat links may not be fully utilized due to exhaustion of the TCP window, unless window scaling is enabled. Long, thin links are likely to carry just a few simultaneous connections, so bandwidth available to a given TCP connection may fluctuate significantly as connections come and go. TCP has difficulty achieving good link utilization under such conditions.

All these factors considered, it is generally agreed that TCP performance over networks including satellite links is often less than that could be desired, and that there is room for improvement.

### III. The basic architecture

Figure 1 shows the architecture we proposed in <sup>1</sup>.



**Figure 1: Basic transport architecture**

The architecture has a lot in common with the split transport architecture that has become popular as the basis of a variety of “performance enhancing proxies” (PEPs), in that it contains a concatenation of separate transport protocol instances, together forming a transport chain between the end systems. While the end-to-end path is heterogeneous in nature, each of the individual transport instances typically operates across a homogeneous sub-network: A LAN, the Internet, the satellite network. By splitting the path, it becomes possible to use for each sub-net a transport protocol that is well suited for the characteristics of that sub-network. Typically, TCP would be used over LANs, intranets and the Internet while the satellite network would use a proprietary, optimized transport protocol or a flavour of TCP tuned specifically for the environment.

What is new in our proposed architecture is the extra end-to-end transport layer inserted between the per-link split transport layer and the application. We call it the Inter Transport Layer (ITL). This “thin” protocol layer re-establishes the end-to-end context that was lost in the split-transport. The end-to-end transport layer can be made to behave, as seen from the application, exactly like a native end-to-end TCP, while benefiting from the performance enhancement of the split-transport layer just below it. As will be explained below, the new layer can also serve as an attractive vehicle for implementing end-to-end security, as well as possibly for QoS signalling and maybe other functions. The latter ideas have not been fully analyzed yet, but will also be addressed very briefly.

#### IV. The SaNTA architecture

The basic architecture described above has an important drawback: All participating end systems need to install the new protocol stack. This may be feasible in some restricted environments, but is not realistic in most of today’s open and dynamic network environments where end systems run a plethora of versions of several operating systems and are not controlled or managed by any central entity. So we started searching for a way to ease this constraint.

Our search was helped by the following observation:

Virtually all corporate users, and, with the proliferation of broadband to the home like ADSL and TV cable, also many private home users, do not connect directly to the Internet. Rather, they are connected to a LAN that, in turn, connects to the rest of the world via a single access point, the access router. Access routers mostly do more than just IP routing: They typically contain Network Address Translation (NAT) and firewall functions and maybe more. The result is that the user LAN, whether it only contains a single home PC or thousands of corporate end systems, can be seen as a separate management and, at least to some extent, security domain that enjoys a certain amount of isolation from the Internet, mediated by the access router. From a performance point of view, user LANs are rather benign environments, exhibiting low delay and packet loss, and mostly have ample bandwidth for the traffic they carry. So they have little influence on the performance of the overall end-to-end path.

This led to the idea that the architecture described above could be operated between access routers, or devices located, in network terms, close to access routers, rather than directly between end systems.

The SaNTA architecture was born.

Figure 2 shows this concept: Two user LANs and, inside the dotted box, an instance of figure 1.

Figure 3 expands this picture to show the full architecture.

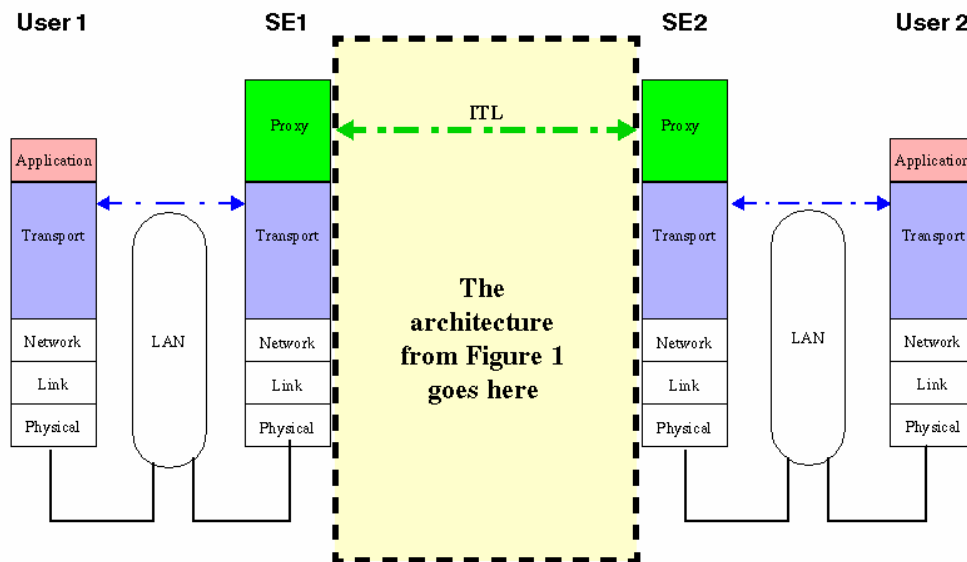
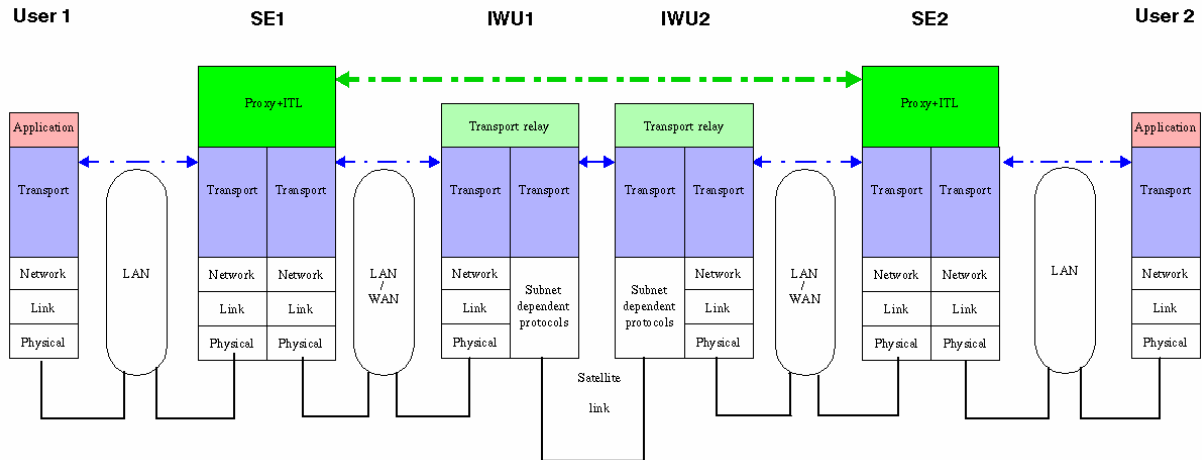


Figure 2: The SaNTA concept



**Figure 3: The full SaNTA architecture**

We call the end points of the new protocol stack Satellite Enhancers (SEs). An SE is effectively a transport proxy: It terminates the TCP connection with the end user and concatenates it with the SaNTA ITL protocol operating between SEs. ITL, in turn, is mediated by a chain of concatenated transport protocol instances.

Many firewalls today contain application proxies, so they, too, split the transport connection. Therefore, the SE is well suited for implementation within or in association with a firewall. In our pilot implementation, we have implemented the SE as a separate entity, so we have not fully explored the potential for integration with other equipment. But it seems that, in concept, there is a good match.

With reference to figures 2 and 3, the following sequence takes place:

- End user 1 initiates a TCP connection towards end user 2.
- The connection is intercepted by SE 1, acting as a transport proxy.
- SE 1 opens an ITL connection to SE 2.
- SE 2, acting as a transport proxy, opens a TCP connection to end user 2.

Opening of the ITL connection consists of the following steps:

- SE 1 opens a TCP connection to Inter working unit (IWU) 1.
- IWU 1 opens a Satellite Transport Protocol connection to IWU 2.
- IWU 2 opens a TCP connection to SE 2.

Two-way end user communication can now take place.

At the end of the session, all connections are disconnected in an orderly fashion.

It is noted that while SEs are proxies, the IWUs are not; they are explicit network entities and are addressed as such.

A hybrid of the SaNTA architecture and the original concept is possible: An end user system may implement the enhanced protocol stack, and may communicate directly with a local IWU. This configuration has been implemented and tested, but will not be discussed further here. It may have some attraction for small, single-user satellite terminals or mobile devices where several logical functionalities tend to be integrated into a small number of physical entities.

## V. Flow and congestion control

One of the primary reasons for introducing a split transport architecture is the sub-optimal performance of TCP's congestion control in the target environment. The solution must, therefore, pay due attention to flow and congestion control. Otherwise, one just replaces one problem with another version of the same problem.

In SaNTA, the philosophy applied is flow control by back-pressure. Each SaNTA transport segment applies its preferred method for dealing with congestion and flow control, and signals back to the SaNTA entity (SE or IWU) in front of it to slow down or stop feeding data if the segment cannot deal with the data rate presented to it. The SaNTA entities, in turn, exercises flow control towards the segment feeding into it when its buffers get full.

This way, at the level of the chain of SaNTA transports, no data ever needs to be dropped due to congestion.

The satellite link will normally be the bottleneck link, as well as the link with the largest latency. Therefore, as described in the next chapter, a transport protocol was chosen that is internally flow controlled, rather than relying on packet losses and retransmissions to signal congestion, like TCP does.

The TCP transport segments will use TCP's congestion and flow control mechanisms, but since these networks are generally faster and have smaller latency than the satellite link, these mechanisms will normally have little or no effect on the end-to-end throughput, provided enough buffering is provided at the IWUs, so that starvation of the satellite link due to the ingress TCP is rare. Providing huge buffers, on the other hand, may lead to other undesirable effects. So an appropriate buffer design is essential.

The protocol stack also includes a scheduler that ensures fairness between flows on the satellite link. This is quite different from TCP where each flow fights for bandwidth, and fairness is rarely achieved. In order to serve real time flows going in parallel with SaNTA enhanced traffic, the scheduler has an optional facility to give priority to UDP traffic (and possible other non-SaNTA traffic – see below) for a given fraction of the available bandwidth. This way, as long as the real time traffic does not exceed the bandwidth set aside for it, it will not compete with best effort traffic for bandwidth.

## VI. The SaNTA Satellite Transport Protocol (SaSTP)

The SaNTA architecture is modular in the sense that each SaNTA segment can use whatever transport protocol is deemed suitable. It is also modular in the sense that, though the figures above show 3 segments (LAN, satellite, Internet), additional segments, and hence additional IWUs may be inserted into the path, e.g. in a multi-hop satellite scenario or a satellite + terrestrial mobile scenario.

Several options were considered for the SaSTP protocol. The choice fell on a derivative of the X.25 LAPB protocol. This protocol is quite simple, and, with a few adaptations, very well suited for a point-to-point link. We opted for selective acknowledgment ARQ in order to speed up recovery from a single or a few packets lost due to link errors.

SaSTP and the protocol stack below it is flow controlled in order not to lose packets between layers. In a fully flow controlled stack, congestion is not an issue.

## VII. Adding network security to SaNTA

Internet security measures fall into three categories: *Application layer security*, *Transport layer security* and *Network layer security* (link and physical layer security is outside the scope of this discussion).

Application layer security is application specific, like Email encryption and authentication. It is totally transparent to lower layer protocols and will not be discussed further here.

Transport layer security uses protocols like TLS/SSL and SSH. These are truly end-to-end security measures that, in spite of their names, tend to be implemented inside applications like web browsers and servers. They are not truly services offered by the network but rather libraries offered to the application developer. They have one important limitation: They are incompatible with application layer proxies and firewalls that need access to the transport headers and possibly data contents.

Network layer security is almost exclusively implemented by IPsec. Though originally intended for true end-to-end security, well in line with the Internet end-to-end principle, it turns out that IPsec is mainly being deployed between access routers in order to implement Virtual Private Networks (VPNs). Some of the main reasons for this are that deploying and configuring IPsec into all end hosts is cumbersome and prone to being ineffective if not configured properly, and that end-to-end IPsec is, at least in part, incompatible with functionalities typically incorporated into today's access routers, like NAT and firewalls.

IPsec is also incompatible with transport or higher layer proxies elsewhere in the network, such as PEPs for satellite links. PEPs need to read, and in some cases change TCP headers. IPsec does not allow this.

Comparing the SaNTA and VPN architectures, we realized two interesting points:

- SaNTA operates the enhanced protocol between SEs located in conjunction with access routers; VPNs typically operate between access routers.
- SaNTA contains an end-to-end layer (ITL) between SEs. The underlying split-transport needs no access to ITL layer data.

This leads directly to the idea of implementing security measures at the ITL layer: The scope of the security is the same as for IPsec VPNs operating between access routers; Security at ITL layer is compatible with split transport below the ITL layer.

The choice of the security protocol to be used at ITL layer is quite open, and a number of options were evaluated. For the prototype implementation, we settled for TLS – the standard version of the well-known SSL protocol. The choice was to some extent driven by implementation issues and availability of suitable libraries, but it seems suitable and logical.

In addition to TLS security at ITL, we chose to include IPsec AH-only security between neighbouring SaNTA entities in order to authenticate all SaNTA entities to each other. As a consequence, the baseline is to not do peer authentication at ITL since all entities in the path are authenticated. But peer authentication is available in the software, and can be switched on, should the concern arise that a rogue SaNTA entity might attempt to insert itself into the network.

The resulting security architecture is shown in figure 4.

In summary, the following security services are provided between SaNTA SEs:

- **Confidentiality** of application data is provided by TLS.
- **Authentication** between neighbouring SaNTA entities is provided by IPsec AH.
- **Authentication** of peer SEs may be optionally provided by TLS.
- **Integrity** is provided SE to SE by TLS as well as between SaNTA entities by IPsec.
- **Replay immunity** is provided SE to SE by TLS as well as between SaNTA entities by IPsec.
- **Denial of Service protection** is provided, to some extent, by IPsec.
- **Non-repudiation** is not provided, and seems out of scope.
- **Access control** to SaNTA entities is outside the scope of the work.
- **Immunity to traffic analysis** can only partly be provided: The ITL connection setup and subsequently all ITL headers have to be in clear.

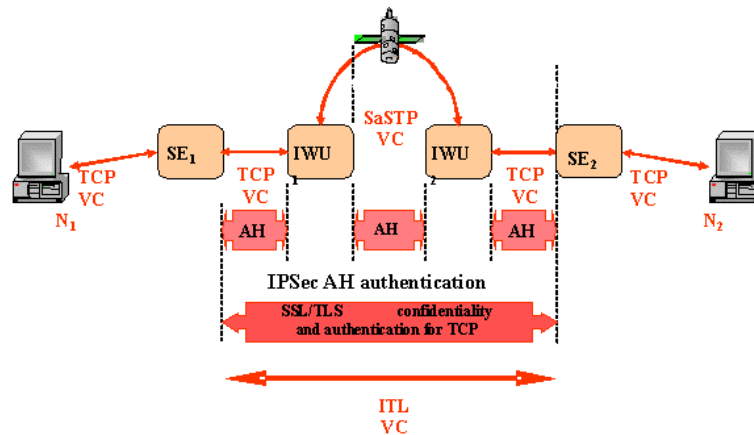


Figure 4: SaNTA TCP security architecture

### VIII. Non-SaNTA traffic

An operational setup will need to be capable of dealing with any kind of Internet traffic, not only TCP traffic. Though the focus on SaNTA is on acceleration of TCP, we have gone at least part of the way towards compatibility with a full Internet environment.

Several non-TCP types of traffic need to be addressed:

UDP traffic is encapsulated into a connectionless ITL protocol by SEs, and is encrypted. Since TLS is a stream encryption protocol, it is not very suitable for connectionless traffic. Therefore, we chose to use IPsec ESP encryption of the ITL payload, including the UDP and original IP headers.

Any other packets encountered will either be treated like the UDP packets, i.e. encapsulated and encrypted, or will be routed at IP layer. We are still debating which packet types fall into which category. Within the current work scope, this is not a high priority issue. The focus at the moment is that such traffic will be served, but will not enjoy any acceleration, and will enjoy cryptographic protection whenever possible within in the current architecture.

The satellite network may contain terminals that are not SaNTA enabled (have no SE). Traffic between a SaNTA enabled terminal and a non-SaNTA enabled one will fall back to normal TCP and will be routed at IP layer. It will not enjoy any acceleration or security by the SaNTA architecture, but will be passed on transparently.

This architecture is shown in figure 5.

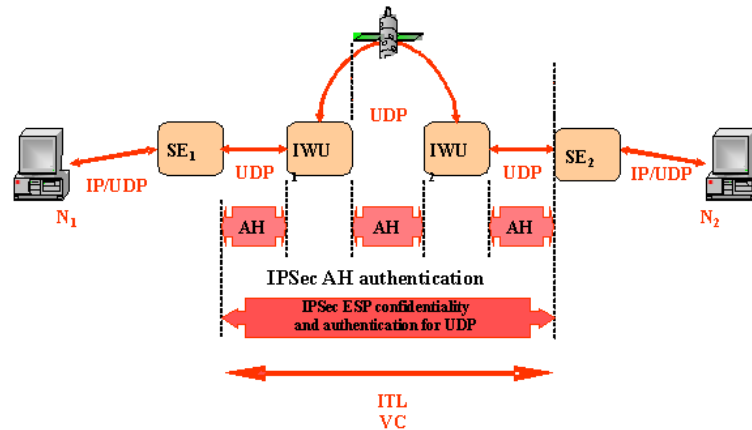


Figure 5: SaNTA UDP security architecture

## IX. Prototype design and implementation

The design of all SaNTA protocols and entities was carried out in SDL (System Definition Language), using the TAU tool suite from Telelogic.

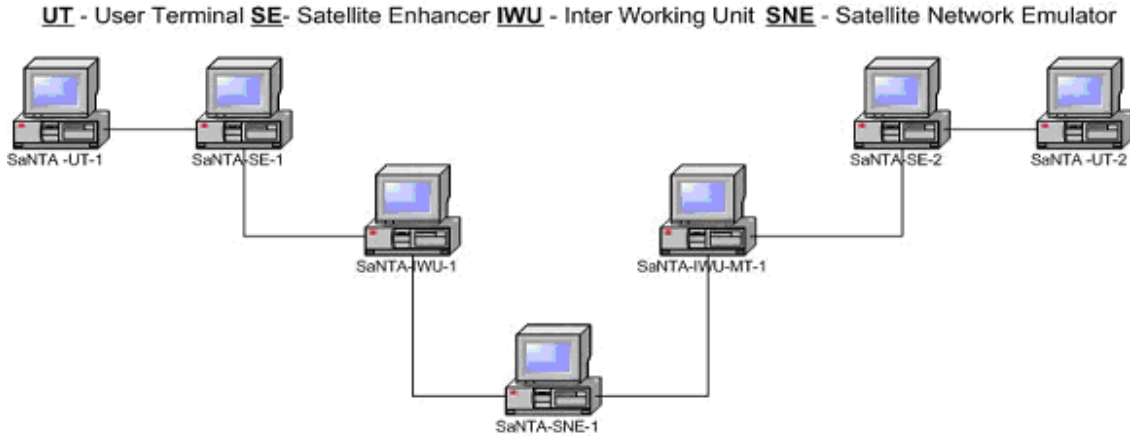
Subsequently, TAU was used in simulation mode to perform a thorough validation of the design. Message sequence charts of typical scenarios, including a variety of faults and mishaps, were produced and evaluated by inspection. TAU also performs a variety of automatic consistency checks.

Once the validation was satisfactorily completed, the TAU code generator was used to generate C code that implements the various entities. A few functionalities that are not easily modelled in SDL were hand-coded in C and integrated with the generated code to form the complete system.

The various SaNTA components were installed on Linux PCs, and make extensive use of Linux libraries and kernel functionalities that were found to be very useful tools, in particular in implementing the security features. OpenSSL was used to implement the ITL layer TLS security while OpenS/WAN and Linux native kernel functionalities were used for IPsec. The Linux features *iptables* and *libipq* proved very versatile and powerful in intercepting IP packets, manipulating them, and re-injecting them into the kernel. *iptables*, in particular, gives a lot of flexibility in selecting which packets get what kind of treatment.

It was the team's first experience with SDL and TAU, so a substantial amount of learning the methodology was required. But, once the approach was mastered, it proved very efficient. In particular, when changes to the design were needed later, it was very easy to update the SDL, repeat the validation if required, and generate new code. The custom-built C-code implements specific low-level functions, so changes to these are really orthogonal to changes to the SDL code, as long as the interface between the two remains.

Figure 6 shows the lab setup, consisting of 7 PCs, representing 2 end user PCs, 2 SEs, 2 IWUs and an emulation of the satellite link, including delay, bandwidth limitation and, optionally, random packet losses.



**Figure 6: Lab test setup.**

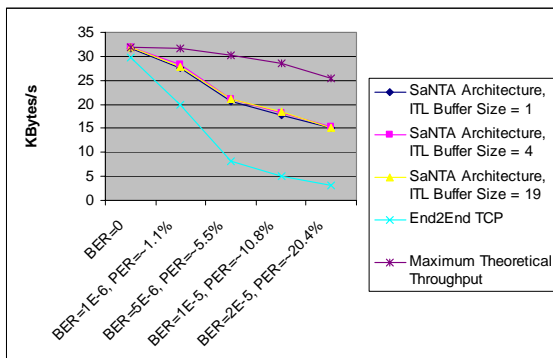
### X. Tests and test results

The majority of tests were conducted in a lab environment. We used the *dummysnet*<sup>2,3</sup> network simulator, that is currently part of FreeBSD, to simulate the satellite link. *dummysnet* allows to simulate a network path with specified delay, bandwidth and packet loss statistics, and was found to be easy to use and quite appropriate for the task at hand. Using a lab setup rather than a live satellite link allows for a well controlled environment and a high degree of reproducibility, but does not fully capture the complexity of a real-life environment. Therefore, tests were also performed over a satellite link kindly made available by Portugal Telecom.

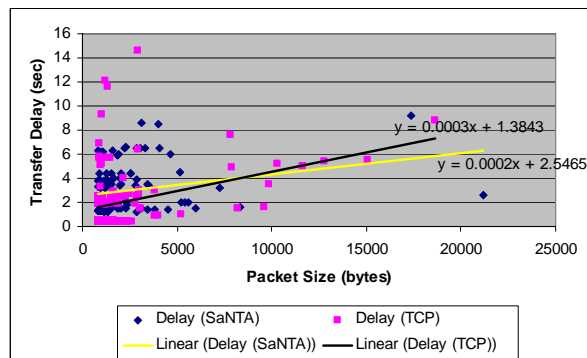
Since the Internet is part of the communication path in most of the envisaged deployment scenarios, it is important to faithfully reproduce its behaviour. It is well known that a good model of Internet behaviour is difficult to come by, so we chose to include a real Internet path in both some of the lab tests and the live tests. We do realize that one specific path is not fully representative of the whole Internet, but it was the best we could afford within the available resources.

The first test objective is to demonstrate that SaNTA achieves its primary objective of enhancing performance as compared to end-to-end TCP. Figure 7 shows the results of one such test: Transferring a large file over a path containing a satellite link with significant BER, TCP performance has dropped significantly at BER  $10^{-6}$ , and becomes unusable soon beyond this point, while SaNTA maintains acceptable performance well beyond BER  $10^{-5}$ . We also see that, in this particular test, the size of the buffer in front of the satellite link has negligible importance.

The exact performance figures obviously depend on the test conditions and parameter values. But, qualitatively, it is demonstrated that SaNTA achieves the envisaged performance enhancement.



**Figure 7: Throughput as function of the BER**



**Figure 8: Transfer Delay, BER = 2e-5**

For short data transfers, the overhead of the SaNTA ITL connection setup becomes significant. As can be seen from figure 8, this additional overhead amounts to about 1 second over a typical GEO path, actually making SaNTA slower than TCP for transfers below some 8-10 Kbytes in size. We will briefly address some ideas for improving this situation in the next chapter.

SaNTA includes a scheduler that attempt to provide fairness between data flows sharing the bottleneck link. It can be seen that this has been achieved: Figure 9 shows the error free case, figure 10 the case for BER  $10^{-6}$ , both for long data transfers. Figure 11 shows that fairness is also achieved in a situation with many partly overlapping short transfers.

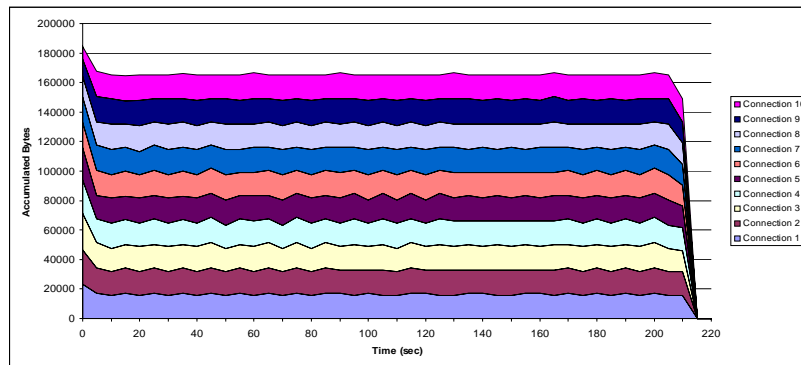


Figure 9: Short-Term End2End Timelog, Buffer Size = 1, BER = 0

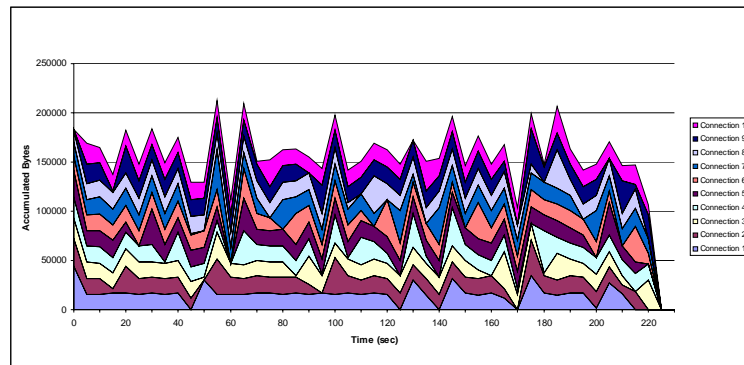


Figure 10: Short-Term End2End Timelog, Buffer Size = 1, BER = 1e-6

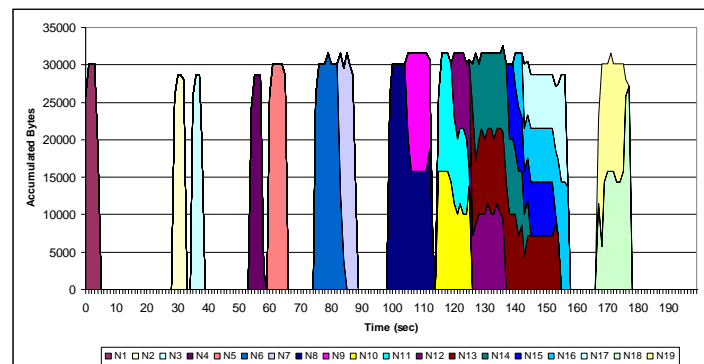


Figure 11: Short-Term End2End Throughput, BER = 0

At the time of writing, the tests relating to the security features are just about to start, and we expect to be able to present some results in the verbal presentation of this paper. The main focus of the envisaged tests is on performance: How does the inclusion of security features impact on the performance of the SaNTA architecture. In addition, several tests are foreseen to verify the correct implementation and functioning of the security features.

## XI. Ideas for future work

During the course of the project, several ideas have come up that we would like to explore in the future, but cannot afford under the current project.

In the current implementation, one TCP connection opens one ITL connection and one TLS security association. This is a performance bottleneck of SaNTA, in particular for short data transfers. In applications like HTTP, it is quite common to open several transport connections between the same end points, either concurrently or sequentially. This suggests that efficiency can be improved on two fronts: By multiplexing several connections within one ITL and/or TLS connection, and by serially re-using those connections rather than breaking and re-establishing them. This is entirely possible within the SaNTA concept.

When designing the SaNTA security architecture, we had to make a choice between using, to the highest extent possible available libraries and kernel functionalities, and doing a proprietary development more or less from scratch. Good security software is notoriously difficult to develop and, in particular, to validate, which is the main reason we chose the former approach. We are, however, aware, that this choice introduces some inefficiencies that could possibly be avoided in a custom development. For example, it might be possible to merge the ITL connection setup and the TLS association setup rather than doing them strictly in series.

Most tests were done in the lab. We would very much like to do more extensive testing in a real environment, and are currently exploring an opportunity to do this.

Last, but not least, we think that the SaNTA architecture could play a role in Quality of Service (QoS) management. The ITL layer, though conceived as an end-to-end layer, actually touches upon all IWUs. SEs and IWUs are located where the characteristics of the network change. But this is also likely to be the interconnection points between QoS management domains. So ITL seems ideally suited as a vehicle to carry QoS signalling. The initiator of an ITL connection could request a certain QoS, each IWU in the path will take note, and will either accept or reject the request. At the end of the ITL connection setup, the initiator will know whether the requested QoS has been accepted, and all IWUs will know what they have granted. The satellite network is likely to be the bottleneck, also from a QoS point of view, so, in practice, the QoS management might often degrade into end points negotiating satellite link QoS. We have not explored this in detail, but it looks quite attractive.

## XII. Conclusion

A novel transport protocol architecture, including a split transport layer and a new end-to-end layer, has been proposed. The architecture maps well onto a typical scenario where user communities access open networks through an access router/firewall. In such a scenario, the new network elements can be associated with the access routers, so that there is no need to change end user systems. The architecture includes a security framework with the unique property of providing true end-to-end security in combination with a split transport architecture for TCP traffic, a feat that cannot be accomplished with network layer security measures. In addition, the architecture provides a secure tunnel for UDP and other non-TCP traffic, though no acceleration is provided for non-TCP protocols.

A working prototype has been built, and tests confirm the feasibility of the architecture, as well as demonstrate the expected performance enhancements as compared to standard end-to-end TCP.

The architecture seems to have the potential to be a vehicle for some attractive future applications. A few of these have been identified and briefly discussed. Also, some ideas have been put forward for further optimizations.

## References

<sup>1</sup>Kristiansen, E., Neves, J., Brazio, J., Pagano, G., Palazzo, S., "Re-thinking the End-to-end Paradigm - The missing protocol layer", ICSSC 2004, Monterey, Ca.

<sup>2</sup>[http://info.iet.unipi.it/~luigi/ip\\_dummyenet/](http://info.iet.unipi.it/~luigi/ip_dummyenet/)

<sup>3</sup><http://www.freesbie.org/>

<sup>4</sup>Kristiansen, E., Donadio, R., "Multimedia over Satellite - The European Space Agency Perspective", Proceedings of the IEEE International Communications Conference, New York, 2002.

<sup>5</sup>Kristiansen, E., "The Missing Protocol Layer", IEEE Communications Society SSC Newsletter Vol. 13, No.2, Dec 2003, <http://www.comsoc.org/socstr/org/operation/techcom/ssc/newsletter/sscnlv13n2.pdf>