Grant Agreement Number: 101014517
Project Acronym: AB4Rail
Project title: Alternative Bearers for Rail

DELIVERABLE D [3.1]

[Review of ACS, of existing transport protocols, application protocols, railway applications]

Project acronym:	AB4Rail
Starting date:	01-01-2021
Duration (in months):	24
Call (part) identifier:	S2R-OC-IP2-02-2020
Grant agreement no:	Number 101014517 – IP/ITD/CCA - IP2
Grant Amendments:	N/A
Due date of deliverable:	31-01-2021
Actual submission date:	25-05-2021
Coordinator:	Franco Mazzenga (Radiolabs)
Lead Beneficiary:	Alessandro Vizzarri (Radiolabs)
Version:	0.1
Type:	Report
Sensitivity or Dissemination level[1]:	PU
Contribution to S2R TDs or WAs[2]	TD2.1
Taxonomy/keywords:	Adaptable Communication System, ACS; IP-based networks for railway; transport protocols; application protocols.

This project has received funding from the Shift2Rail Joint Undertaking (JU) under grant agreement No. 101014517. The JU receives support from the European Union's Horizon 2020 research and innovation programme and the Shift2Rail JU members other than the Union.

[1] PU: Public; CO: Confidential, only for members of the consortium (including Commission Services)

[2] https://projects.shift2rail.org/s2r_matrixtd.aspx

The document history table provides a summary of all the changes in reverse chronological order (latest version first).

Document history

Date	Name	Affiliation	Position/Project Role	Action/ Short Description
27 Febr. 2021	Alessandro Vizzarri	Radiolabs (RDL)	Technical Manager/WP leader	Description of the selected ABs for rail.
25 May 2021	Alessandro Vizzarri	Radiolabs (RDL)	Technical Manager/WP leader	The updated document includes all the revisions provided by PO.

Disclaimer

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author's view – the Shift2Rail Joint Undertaking is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.

Table of Contents

Executive Summary

The adaptable communication system (ACS) has been designed to operate in accordance with the bearer independence principle, which is achieved using communication technologies that can be accessed using IP-based transport and application protocols. For this reason, this deliverable 3.1 (D3.1) starts with a short review of the concepts and of the organization of typical IP-based communication networks and the corresponding transport and application layer protocols. The most important terminology is also reviewed. The document continues with an extensive review of the operation principles of ACS over the IP networks.

A functional model of ACS operations over IP-based bearers (or networks) is also provided and discussed. The model evidences the roles of the transport and application protocols in ACS. The ACS is actively supported by the Shift2Rail community as a medium/long term flexible communication solution that will complement, integrate and substitute the FRMCS. A discussion on ACS and FRMCS is presented in Appendix B (Section 10). The review on ACS is focused on its features and operational characteristics in terms of its (IP-based) envisaged network architecture and the functional analysis of its main element i.e., the ACS gateway (ACS-GW). The main rail applications and the application classes identified in ACS are also reviewed and will be considered in the subsequent railway activities.

The main features of the transport protocols that will be considered in the development of subsequent AB4Rail WP3 activities are summarized in Section 5. The analysis includes a general discussion on the main services offered by a transport protocol to the application layer. The transport protocols that are described in the document are: (i) Transmission Control Protocol (TCP) and its multipath version (i.e., MP-TCP), (ii) User Datagram Protocol (UDP), (iii) Stream Control Transmission Protocol (SCTP), and (iv) Quick UDP Internet Connections (QUIC). The cubic and the Bandwidth and Round-trip propagation time (BBR) congestion control strategies for TCP are detailed. The discussion on transport protocol concludes with some notes on the Rail Safe Transport Application (RaSTA) protocol, which is specifically designed for rail signaling but is currently not a published norm (only a pre-version exists) and its specifications are not available for free. A comparison among these transport protocols has been performed in terms of provided features to higher layers.

The review continues with the analysis of the main application protocols that are layered on top of the transport protocols. In principle, any application could craft its own network protocol. Similarly, to the transport protocols, a general discussion on the desirable characteristics of a generic application protocol is first presented. However, many applications are standardized, and they work with well-known and widely accepted application protocols. Among them, the most important ones are reviewed in this deliverable i.e., (i) Hyper-Text Transfer Protocol (HTTP), (ii) File Transfer Protocol (FTP), and (iii) Remote Desktop Protocol (RDP). Their secure counterparts are also discussed including HTTPS and FTPS (in place of SFTP). Both HTTPS and FTPS are based on the Transport Layer Security (TLS) layer, which is also discussed in detail in Appendix (Section 9). In Appendix, we also discuss the main features of the SIP protocol and its most important issues, such as the NAT traversal, when it is adopted over IP networks including middleboxes. The SIP protocol is of great importance for ACS signaling. In Appendix B (Section 10) we briefly review other projects related to rail sector, concerning the usage of wireless sensor network technologies for train monitoring, control, safety, integrity etc. (i.e., CONNECTA, DEWI/SCOTT).

List of abbreviations, acronyms, and definitions

Acronym	Definition
3GPP	Third Generation Partnership Project
ACS	Adaptable Communication System
AIMD	Additive Increase Multiplicative Decrease
API	Application Programming Interface
ASCI	Advanced Speech Call Items
ATP	Automatic Train Protection
ato	Automatic Train Operation
BIC	Bearer Independent Communication
BBR	Bottleneck Bandwidth and Round-trip propagation time
CA	Certification Authority
CBTC	Communication-Based Train Control
CCS	Command, Control and Signaling
CWND	Congestion Window
DNS	Domain Name System
DOS	Denial Of Service
DPI	Deep Packet Inspection
DTLS	Datagram Transport Layer Security
ERTMS	European Rail Traffic Management System
ETCS	European Train Control System
E2E	End-to-End
FIFO	First In – First Out
FRMCS	Future Rail Mobile Communication System
FTP	File Transfer Protocol
FTPS	FTP Secure
GCC	Generic Conference Control
GSM-R	Global System for Mobile Communications – Railway
HAPS	High Altitude Platform System
HTTP	Hypertext Transfer Protocol
ICE	Interactive Connectivity Establishment
IDS	Intrusion detection system
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMS	IP Multimedia Sub-system
IP	Internet Protocol
IP2	Innovation Program 2
ITU	International Telecommunication Union
MCPTT	Mission Critical Push-to-Talk
MCS MUX	Multipoint Communication Service multiplexer
MIME	Multipurpose Internet Mail Extensions
MITM	Man In The Middle

MNO	Mobile Network Operator
MPTCP	Multipath TCP
MTU	Maximum Transfer Unit
NA	Network Application
NAT	Network Address Translation
OA	On-board Application
OTT	Over-the-Top
PLMN	Public Land Mobile Network
PMR	Private Mobile Radio
PMTUD	Path MTU Discovery
PSTN	Public Switched Telecommunication Network
PPDR	Public Protection and Disaster Relief
QoS/QoE	Quality of Service/Quality of Experience
QUIC	Quick UDP Internet Connections
RAT	Radio Access Technology
RPC	Remote Procedure Call
RTCP	Real-Time Control Protocol
RTSP	Real-Time Streaming Protocol
RTP	Real-Time Protocol
RTT	Round-Trip Time
SBC	Session Border Controller
SCTP	Stream Control Transmission Protocol
SIL	Safety Integrity Level
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SSH	Secure Shell
SSL	Secure Sockets Layer
STUN	Simple Traversal of UDP through NATs
S2R JU	European Shift2Rail Joint Undertaking
TCMS	Train Control and Monitoring System
TCP	Transmission Control Protocol
TD	Technology Demonstrator
TETRA	Terrestrial Trunked Radio
TMS	Traffic Management System
TLS	Transport Layer Security
TSAP	Transport Service Access Point
TURN	Traversal Using Relay NAT
UDP	User Datagram Protocol
UIC	Union internationale des chemins de fer
URI	Uniform Resource Identifier
URL	Uniform Resource Location
URN	Uniform Resource Name
VoIP	Voice over IP
WWW	World-Wide Web

List of Figures

List of Tables

1. Introduction

This document constitutes the Deliverable D3.1 “Review of ACS, of existing transport protocols, application protocols, railway applications” according to Shift2Rail Joint Undertaking programme of the project titled “Alternative Bearer for Rail” (Project Acronym: AB4Rail, Grant Agreement No 101014517 — IP/ITD/CCA — IP2). On 22nd July 2020, the European Commission awarded a grant to the AB4Rail consortium of the Shift2Rail / Horizon 2020 call (S2R-OC-IP2-02-2020). AB4Rail is a project connected to the development of a new Communication System planned within the Technical Demonstrator TD2.1 of the 2nd Innovation Programme (IP2) of Shift2Rail JU: Advanced Traffic Management & Control Systems.
The IP2 “Advanced Traffic Management & Control Systems” is one of the five asset-specific Innovation Programmes (IPs), covering all the different structural (technical) and functional (process) sub-systems related to control, command, and communication of railway systems.

1.1 **Purpose and scope of the document**

The aim of this document is to provide a detailed overview of the main features of Adaptable Communication System (ACS), together with the main characteristics of existing transport protocols and application protocols viable for the railway applications.

1.2 **Document organization**

Section 2 introduces the technological context. The Section 3 provides a first review of Adaptable Communication System (ACS), while Section 4 a reviews of railway applications. The section 5 describes the existing transport protocols, the Section 6 the application protocols.
The main conclusions are included in Section 7. Sections 9 and 10 contain Appendix A and Appendix B. the first one is focused on Transport Layer Security (TLS) protocol and Session Initiation Protocol (SIP9), while the second one is focused on the main research projects on transport and applications in railway environment.

1.3 **Reference Documents**

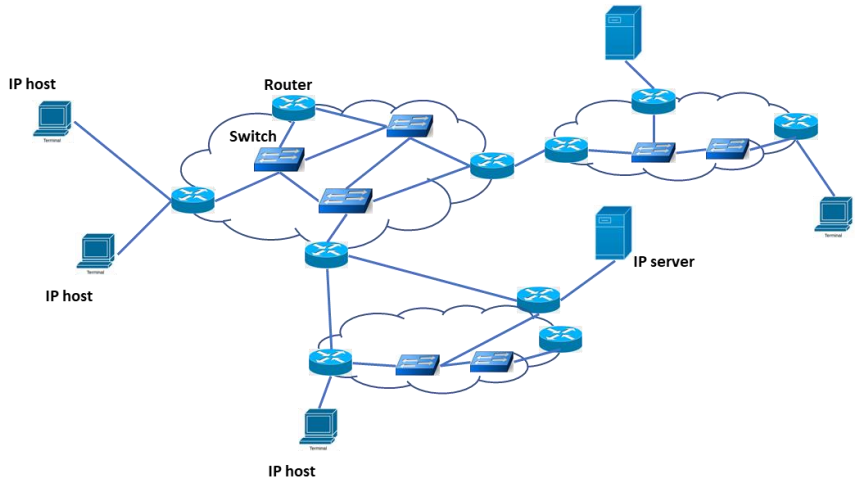
Table 1: Reference Documents.

Document Number	Document Description
RD-1	AB4Rail Grant Agreement Number 101014517 - IP/ITD/CCA - IP2
RD-2	AB4Rail Consortium Agreement

2. The technological context

IP-based networks are typically built using an architectural principle of a simple network and agile/smart edges. The basic approach is to assume the network is a simple collection of buffered switches (or routers) and links. Packets traversing the network are passed from router to router. Each router selects the next link to use to forward the packet closer to its intended destination. If the link is busy, the packet will be placed in a queue and processed later, when the link returns to be available. If the queue is full, the packet is discarded. This network behavior is not entirely useful if what you want is a reliable data stream to transit through the network. The approach adopted by the IP architecture is to pass the responsibility for managing the data flow, including detecting and repairing packet loss, as well as managing the data flow rate, to an end-to-end transport protocol, as well as to application protocols (Figure 1).

Figure 1. Principle architecture of the Internet with its main elements.



In order to better understand the topics that will be discussed in this document, it could be useful to remind some important definitions and concepts typical of an IP-based network. In general, protocols used over IP networks at different levels of the protocol stack can be classified into two broad categories i.e., (i) stateless and (ii) stateful protocols.

A stateless protocol is a communications protocol in which no session information is retained by the receiver (typically a server). Relevant session data is sent to the receiver by the client in such a way that every packet of information transferred can be understood in isolation, without context information from previous packets in the session. This property of stateless protocols makes them ideal in high volume applications, increasing performance by removing server load caused by retention of session information. A stateless protocol does not require the server to retain session information or status about each communicating partner for the duration of multiple requests. A protocol that requires keeping of the internal state on the server is known as a stateful protocol. Focusing on transport protocols, a TCP connection-oriented session is a stateful connection because both systems maintain information about the session itself during its life. On the contrary, the Internet Protocol (IP) and the Hypertext Transfer Protocol (HTTP) are stateless. The stateless design simplifies the server design because there is no need to dynamically allocate storage to deal with conversations in progress. If a client session dies in mid-transaction, no part of the system needs to be responsible for cleaning up the present state of the server. A disadvantage of statelessness is that it may be necessary to include additional information in every request, and this extra information will need to be interpreted by the server. Transport and application protocols are end-to-end

protocols since they reside on two (or) more communicating hosts. Their behavior is not easy to characterize even taking into account for the presence of middleboxes over the communication path and many times the emulation/simulation is the only viable method to assess their end-to-end performance. In general, Internet is not an easy environment for data transfer. In the modern Internet network, the presence of middleboxes (i.e., Firewall, NAT etc.) leads to the choice of transport protocols that are finally able to cross them. A middlebox is defined as any intermediary device performing specific functions, differently from normal standard functions of an IP router on the datagram path between a source host and destination host. In some discussions, especially those concentrating on HTTP traffic, the word "intermediary" is used. More precisely, a middlebox is a computer networking device that transforms, inspects, filters, and manipulates traffic for purposes other than packet forwarding. These "extraneous" functions can interfere with application performance and have been criticized for violating "important architectural principles" such as the end-to-end principle. Examples of (important and necessary) middleboxes include firewalls, network address translators (NATs), load balancers, and deep packet inspection (DPI) boxes. Furthermore, middleboxes are widely deployed across both private and public networks. On the other side, dedicated middlebox hardware is widely deployed in enterprise networks to improve network security. In addition, home network routers often have integrated firewall, NAT, or other middlebox functionalities.

The main examples of commonly deployed middleboxes are:

- **Firewalls**: they filter traffic based on a set of predefined security rules defined by a network administrator. Firewalls can be defined at any level of the protocol stack. As an example, IP firewalls reject packets "based purely on fields in the IP and transport headers". Other types of firewalls may inspect traffic at the session or even at application layer;
- **Intrusion detection systems (IDSs)**: they are in charge of monitoring traffic and collect data for offline analysis for security anomalies. IDSs do not filter packets in real time and are capable of more complex inspection to decide whether to accept or reject packets;
- **Network address translators (NATs)**: they replace the source and/or destination IP addresses of packets that cross them. Typically, NATs are deployed to allow multiple end-hosts to share a single IP address. Hosts "behind" the NAT are assigned a private IP address and their packets destined to the public Internet cross a NAT, which replaces their internal private address with a shared public address;
- **WAN optimizers**: they improve bandwidth consumption and perceived latency between endpoints. This is achieved with the addition of Quality of Experience (QoE) platforms including edge cloud devices and content delivery servers;
- **Load balancers**: they provide one point of entry to a service, but forward traffic flows to one or more hosts that actually provide the service.

The ACS approach should create an overlay network over existing IP networks by means of extensive usage of tunnels. The successful setup of one tunnel among two endpoints depends on the ability of the tunnel protocols and of the underlying tunnel transport protocols (e.g., UDP is typical for the tunnel) to properly operate in the presence and across the middleboxes. If we assume the protocols used by ACS to create tunnels allow to overcome middleboxes over the underlay IP networks, in principle the tunnel-based ACS communications should be free by middleboxes effects. This important consideration will be used in the development of the analysis activities of the protocols object of the activities in the subsequent tasks of the WP3 of the AB4Rail project. However, in our activities we are not interested in describing tunnel protocols as they are transparent to applications running on communicating hosts. What matters is only the end-to-end data transfer in the AB4Rail project. Thus, this deliverable summarizes the main features of the transport and application protocols that will be considered in successive AB4Rail activities. The document is

organized as follows. Section **Errore. L'origine riferimento non è stata trovata.** reviews the ACS system. Section 4 describes the railway applications. In Section 5, Existing transport protocols are described and their features are reported. In Section 6, main application protocols are described. It is also reported the security version of HTTP and FTP. In Appendix A, the work operation of the TLS and SIP protocols is reported. Moreover, in Appendix B we briefly review other projects related to rail sector, concerning the usage of wireless sensor network technologies for train monitoring, control, safety, integrity etc. (i.e., CONNECTA, DEWI/SCOTT).

3. Review of ACS

The main motivations for novel railway communication systems can be summarized as follows. First of all, the introduction of GSM-R as (a mandatory) radio communication system for operational voice and European Train Control System (ETCS) data communication between infrastructure and on-board systems has been a stable pillar for the European railway area. The GSM-R adopts a single radio technology and transmits in a single radio spectrum band. Due to the recent advancements/evolution of railways services mostly based on data transmissions, it has become clear that basing railway communications on a single radio technology -that is shortly becoming obsolescent- is not a viable approach to going forward. Thus, a more flexible approach is needed for the future that should be based on multi communication technologies.

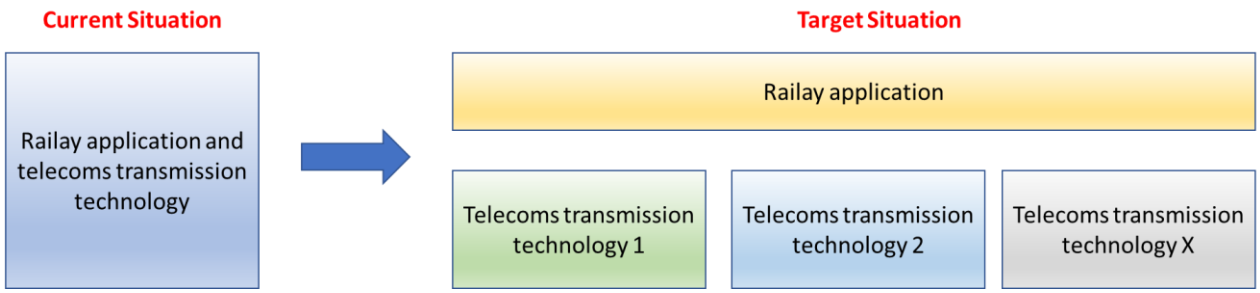
3.1 Actual railway communication systems

The communication subsystem for rail applications is a fundamental component of the Command, Control and Signaling system (CCS). This subsystem supports the communication between onboard, trackside and adjacent or central system. Today, European Rail Traffic Management System (ERTMS, https://ec.europa.eu/transport/modes/rail/ertms_en) relies on communication system, which is based on GSM technology, called GSM-R, enhanced with Advanced Speech Call Items (ASCI) and railway specific functionality, defined in MORANE and EIRENE. GSM-R technology is predominately deployed on high-speed and mainline tracks across Europe to enable communication interoperability. At the same time, in some countries infrastructure managers are still relying on alternative communication systems based on analogue radio, such as Private Mobile Radio (PMR) or public mobile operator services, for tracks in rural or remote areas.
In areas with high user density or significant traffic demand, some operators also use alternative radio system like PMR to overcome radio resource shortage imposed by GSM-R. The urban rail domain is usually not concerned with interoperability regulations and therefore it has implemented networks based on TETRA (TErrestrial Trunked RAdio) for voice requirements and the Wi-Fi variant IEEE802.11p (now IEEE 802.11-2016) for data applications, including the support of the urban rail train control system CBTC (Communication-Based Train Control). Railways are currently using dedicated systems in dedicated spectrum, in order to manage the network and the subscribers in a fully independent way.
Railways require a strict control over the subscriber base, the entitlements of each user and over the level of coverage and service availability in all the geographical extension of the railway lines supported by the system. It is obvious that the railway communication system has become a complex patchwork of networks and radio technologies with limited interworking and inter-system handover capabilities.
With GSM-R, railway-specific functionality is directly integrated into the technical specification of the transmission technology. In the following we enlist the main EIRENE specifications:

- Coverage probability of 95% based on a coverage level of -98 dBm for voice and non-safety critical data;
- Coverage probability of 95% based on a coverage level of -95 dBm on lines with ETCS levels 2/3 for speeds lower than or equal to 220 km/h;
- Coverage probability of 95% based on a coverage level between -95 dBm and –92 dBm on lines with ETCS levels 2/3 for speeds above 220km/h and lower than or equal to 280km/h;
- Coverage probability of 95% based on a coverage level of -92 dBm on lines with ETCS levels 2/3 for speeds above 280km/h.

It should be observed that features such as railway emergency calls and location dependent addressing (used for functional numbering) are an integral part of the GSM-R specification. Many of these application layer capabilities depend on the capabilities of the underlying GSM-R network that would not necessarily be present in a general-purpose commercial wireless network, neither in a new radio network technology.
These network capabilities are specified together with the applications in the same GSM-R specifications. This means that there is no independence of rail applications from the transmission technology, and this is a problem for introducing novel applications in the railway ecosystem.
As technologies and requirements evolve, there is a widely recognized need to allow the use of alternative transmission technologies for track-to-train operational communications. There is the necessity to move beyond 2G technology (which at some point will no longer be maintainable), and to support general purpose satellite or Wi-Fi technology. Operational rail applications will need to be able to run over transmission substrates that do not necessarily provide the same capabilities and features that GSM-R networks provide today.
Therefore, separating the bearer (the technical transmission technology aspects) from the application (railway functionality elements) becomes a necessity. This separation is the essence of the concept of bearer independence. The schematic representation of the Bearer Independent Communications concept is depicted in Figure 2.

Figure 2: Bearer Independent Communications concept [1]



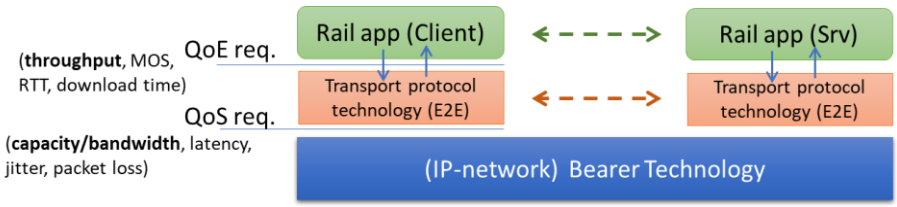
The working definition of the bearer independent principle is reported below.

- Bearer Independent Communications (BIC) occur between two or more users and/or applications over a single access network or multiple heterogeneous access networks with no dependence on the availability of any bearer-specific features.

In the current network practice, to achieve BIC, communications should be based on the IP protocol. In principle, communications between applications could be supported over any IP-based bearer,

whether commercial (public) or private (i.e., the railway proprietary communication infrastructure), without assuming that the bearer provides any capabilities beyond data transmission using the Internet Protocol (IP). Furthermore, service continuity is required in both stationary and (high) mobility scenarios with the transparent use of one or more bearers.
Due to BIC adoption, the specifications of communication requirements for railway applications are moved upper in the protocol stack i.e., towards the application layer and above the transport layer. This is a paradigm shift and it is illustrated in the following of this Section.
Due to bearer-application separation, the specifications of requirements for railway communications are now expressed in terms of the Quality of Service/Quality of Experience (QoS/QoE) requirements that must be perceived by the railway app, running on top of the transmission (bearer) technology.

Figure 3: bearer-application separation.



As shown in
Figure 3, the railway application may interact with the (IP-based) bearer through the end-to-end (E2E) transport (protocol) technology and to properly operate, the railway application could require a specified throughput (to be not confused with the transmission capacity/bandwidth defined at the IP layer as shown in the same Figure 3).
To be selected for transporting the data from the railway application, the communication bearer must guarantee the achievement of QoE requirements, which can be related to QoS requirements issued on the key performance indicators such as delay, packet loss, packet error, jitter, which characterize the behavior of the bearer as seen by the protocol transport layer.
The new railway communication system should be able to support multi-bearers, as well as multi-link communications. To this purpose, SYSTRA in 2016 [1] has identified two competing architectural models of supporting multi-bearers listed in the following points:

- The conservative view: this is based on 3GPP specifications, and on services similar to those offered by Mobile Network Operators (MNOs). The Future Rail Mobile Communication System (FRMCS) is the most important example;
- The flexible view: this is based on Over-the-Top (OTT) service view using IETF and IEEE standards. This view potentially expands the set of bearers to include bearers such as Wi-Fi and satellite, HAPS (High Altitude Platform System) and many other technologies.

It has been observed that the flexible view should be preferred in the medium / long term. It offers the broadest possibility to evolve operational rail communications over time. However, in the near term, there could however be benefit in implementing 3GPP-based solutions (the conservative view) as a stepping stone on the way to the realization of the flexible view. In fact, in accordance with the flexible view, 3GPP-based solutions ultimately become one solution out of many.
The shift to communications based on the bearer independence implies the need for re-thinking of the ecosystem of rail operation transmission systems and rail operational applications.

3.2 More on the conservative and flexible views of communication systems

In this Subsection we summarize some important findings from SYSTRA concerning the applicability of the BIC principle in railway [1]. The full and comprehensive bearer independence is unlikely to be achieved until GSM-R has been decommissioned. This also means passing from circuit switched-based services to packet switched-based services. The conditions for the success of the BIC are related mainly to:

- Independence of the application layer and the transport;
- Guaranteed connectivity;
- Management of mobility in the different railway scenarios;
- Ecosystem development;
- The deployment plan;
- Operational management.

In the conservative view (e.g., FRMCS, see Appendix B) operational rail communications could potentially take advantage of the Mission Critical Push-to-Talk (MCPTT) functionality that is an optional function in 3GPP Release 13 standards. It is unlikely, however, that MNOs will deploy MCPTT unless they perceive sufficient demand for it from Public Protection and Disaster Relief (PPDR) and/or operational rail. If MNOs do not demand these features, manufacturers will not implement them. Wi-Fi calling depends on 3GPP features. It is thus consistent with what has been referred to as the conservative view, but not with full bearer independence (i.e., the flexible view).
In the flexible view with bearer independence, we are seeking to deliver both voice and data operational rail communication services independently of underlying bearer-specific features. Today, it is routine for commercial data services to be fully bearer independent, and for voice services to run on top of the bearer independent data services (as Voice over IP, VoIP). This is the most natural approach for bearer independent operational rail communications to take.
In fact, current standards for many IP-based candidate bearers are potentially adequate for operational rail communications, once bearer-specific dependencies have been eliminated. Two areas that would nonetheless require intensive attention before incorporating any bearer into operational rail standards are (i) reliability and robustness requirements and (ii) QoS requirements, including end-to-end latencies.
One issue considered in the SYSTRA document focused on the network ownership and operational models, and their implications [1]. We summarize some of the findings:

- Spectrum options relevant for BIC provide a patchwork of models from legacy GSM-R model to a fully operated network model. This patchwork could be a major brake of putting into service standardized interference resilient products. Harmonized spectrum for railway applications could provide a framework to standardization of interference resilient equipment, thus answering reliability figures of the FRMCS;
- The dedicated network is the preferred overall network model for GSM-R, but the alternative arrangements that are already in place demonstrate that other models may have value. The migration to fully bearer independent communications has the potential to facilitate the further evolution of other network operational models;
- The introduction of business and operational models other than dedicated GSM-R networks offers opportunities for rail operations, but it potentially also leads to considerably more technical and contractual complexity.

With the move away from a single, dedicated GSM-R network, there is a need in many scenarios to establish clear Service Level Agreements (SLAs) in order to ensure that the necessary QoS is provided. Where multiple bearers operate in parallel, these SLAs might be tricky to implement.

Furthermore, the shift to bearer independence implies the need for re-thinking of the ecosystem of rail operation transmission systems and rail operational applications. If third parties are to be able to supply operational rail applications (especially in on-board systems), attention must be paid to ensure open operating system platforms with well documented interfaces, open well documented communications Application Programming Interfaces (APIs), competitive constraints are not allowed to impede entry of new players, the certification and approval process needs to carefully consider how to enable this kind of innovation without sacrificing safety, and legal and regulatory requirements need to be fully thought through.
In a bearer independent context, the network neutrality is also an important issue. In particular, if a commercial MNO were to support operational rail communications, network neutrality rules (Regulation 2015/2120) are inapplicable because operational rail communications do not provide access to the Internet, and do not serve the general public. Even if they were applicable, operational rail communications would clearly represent a specialized service. Network neutrality does not appear to pose an impediment to the use of public mobile networks for operational rail. Network neutrality (Regulation 2015/2120) does not appear to be applicable to rail passenger communications, as long as the services are offered only to passengers. To the extent that they are offered to the public at large (in train stations, for example), then the network neutrality rules might well be applicable.
At this point, the question is if there is some idea or some way to realize a BIC-based rail communication system. The answer is: the Adaptable Communication System (ACS) is conceived and designed to provide an effective answer to the necessity of a railway communication system based on the BIC Principle.

3.3 The Adaptable Communication System (ACS)

The main goals of the Innovation Program 2 (IP2) of the European Shift2Rail Joint Undertaking (S2R JU), a Technology Demonstrator (TD) “adaptable communications for all railways” activity is:

- To bring together key stakeholders to investigate the future communication needs;
- To define and specify key functions;
- Develop prototypes to prove the feasibility and capabilities of an ACS for railways.

So far, the FRMCS project is mainly concerned with requirements from the high-speed and mainline domains, which are typically served by GSM-R radio systems. From that perspective, FRMCS is primarily analyzing a successor system for GSM-R. In contrast, the scope of WP3 in S2R is wider and in addition covers regional, freight, urban and metro line(s). While the FRMCS project solely relies on 3GPP specifications (even for interoperability) and building blocks to provide bearer flexibility and QoS management for applications, the ACS intends to provide these functions also independently of the 3GPP system.
These approaches are not mutually exclusive. Nevertheless, Shif2Rail WP3 specification and prototyping activities focus in particular on applicability of FRMCS (USR) use-cases with the ACS.

3.4 A short story of ACS and references

ACS is defined in X2Rail-1 Del. D3.3, “*Specification of the Communication System and Guideline for Choice of Technology*”, 29-01-2019 [2]. The specification considers a model-based architecture to achieve maximum reusability and independence of technologies.
ACS **is a cost effective and scalable communication system** for handling multiple access networks and it is designed to enable **bearer independency**. The ACS allows:

- QoS control;

- Enhanced throughput by carrier aggregation;
- Improved resilience by data redundancy allowing access for and interoperability of railway applications.

The previous objectives can be achieved by providing:

- Transparency of networks in use for applications such as voice or signaling
- Concurrent use of access networks
- Handover management
- Capability to select and change network technology based on application profile and communications resource availability
- Priority and pre-emption capabilities depending on application profiles
- Support of end-to-end application security
- Set of common application interfaces for authentication, profile and notification management.

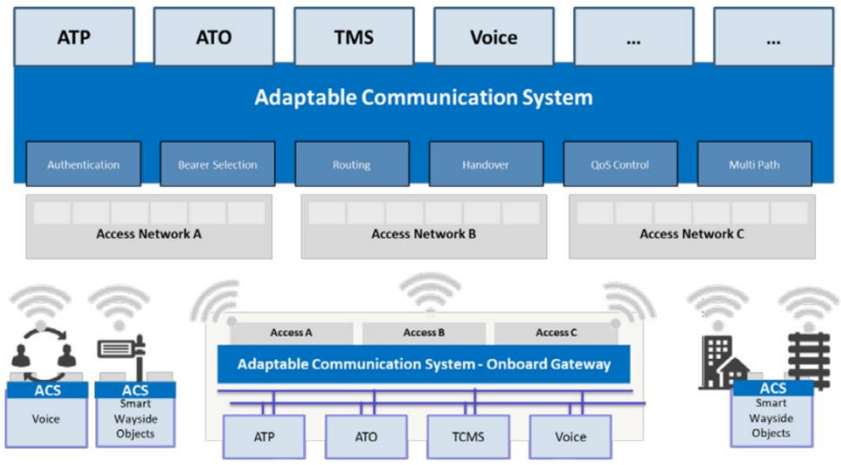
The ACS system (or "vision") can be realized by defining, designing and developing its basic building block, which is the ACS Gateway (ACS-GW). It is the building block for achieving the ACS objectives.

As indicated in [2] the basic architecture of the ACS includes the following three domains:

- The Application domain
- The ACS domain
- The Network domain

The ACS interconnects application domains through the network domain. It acts as an intelligent, flexible, etc. intermediary between two domains. Instead, applications interface with ACS devices and networks only interface with the ACS. ACS is designed in accordance with the BIP principle. The network domain can include different communication bearers (wired/wireless) and the ACS device can interface with different core networks. **To effectively implement the BIP concept, interfacing of ACS with the network domain should occur at IP level.**
The ACS operations is better detailed in Figure 4 (taken from [2]).

Figure 4: Main blocks in ACS



The ACS shall provide some (basic) **services** to railway applications such as Automatic Train Protection (ATP), Automatic Train Operation (ATO), Traffic Management System (TMS), Voice

services, etc. Main ACS services specifically oriented to support data transfer from the railway applications are:

- **Registration:** it includes identification, authentication and authorization;
- **Session management:** it includes session setup, session negotiation and session termination;
- **Information and event services:** it includes location, communication characteristics, coverage.

This is achieved through a **common application interface** to allow interaction between the ACS and railway applications. The interface **is the same** for the onboard gateway, central (trackside) applications and wayside objects from a functional point of view.
Other network services to be implemented in the ACS system to support data transfer are:

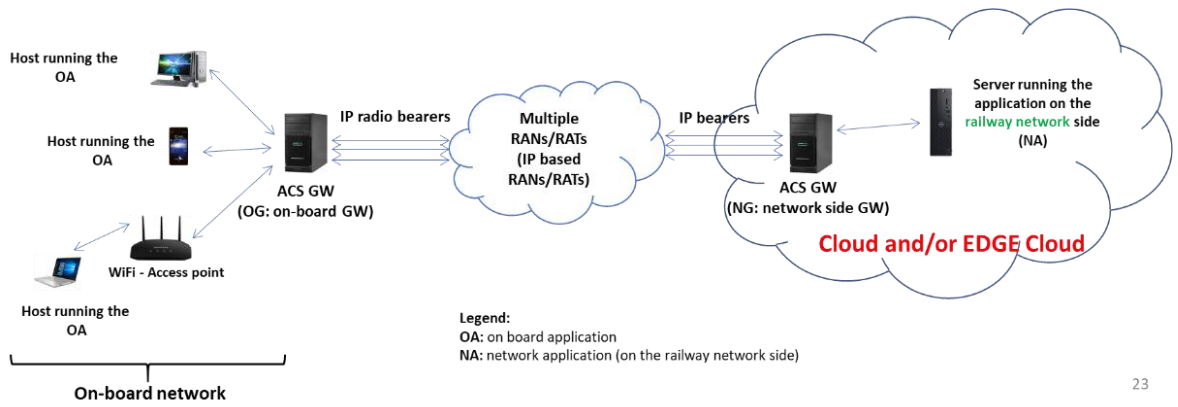
- (Automatic/intelligent) Bearer Selection and management;
- Routing among different bearers;
- Handover: inter and intra bearer;
- QoS control over the bearers;
- Possibility to enable Multipath links for the applications.

Finally, for what concerns the addressing scheme in ACS-based networks, the IPv6 is adopted but, for compatibility, IPv4 is supported too. In this case, interworking issues between IPv4 and IPv6 could arise and ACS should solve these problems too.
The main ACS requirements are detailed in Table 1 in [2]. The requirements better summarizing the ACS characteristics are:

- ACS should allow parallel use of the available networks, and should implement network "handover" (vertical/horizontal) capability to guarantee the availability of communication services based on multiple active bearers;
- The type of network used to convey data from the application should be transparent to the application itself;
- The selection of the networks as well as of the communication bearer should be based on QoS mechanisms and not on technology.

The ACS GW is the **keystone** device allowing to setup an ACS-based communication system. The typical scheme of the ACS-based communication is depicted in Figure 5.

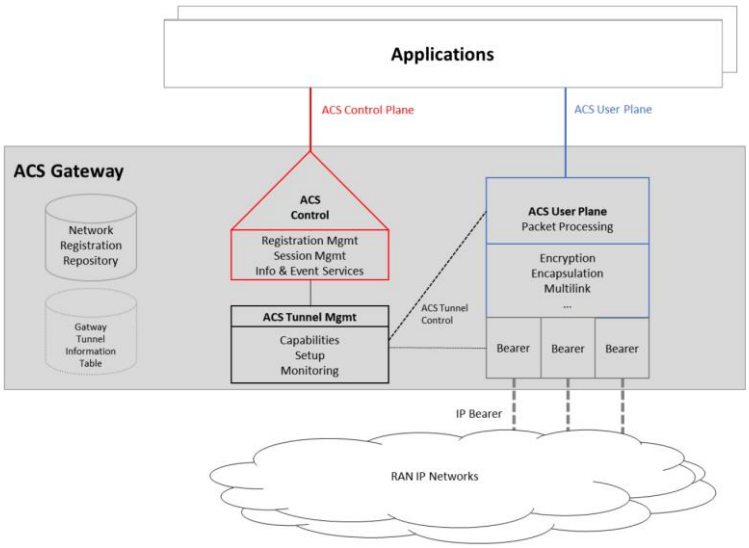
Figure 5: typical scheme of the ACS-based communication



As shown in Figure 5, the devices in the on-board network are connected to the ACS-GW, which in turn interfaces with multiple IP-based radio access technologies (RAT) and/or networks.

Depending on the specific railway scenario and applications requirements, on the network side the ACS-GW can be positioned in the Cloud (i.e., the railway data-center) or in the edge cloud (i.e., using the computational facilities provided for example by the Multi Access Edge Cloud devices, as recently standardized by ETSI, that will be integrated in the actual 4G and future 5G radio access networks). The edge cloud solution can be envisaged for those applications with very tight delay requirements.
The principle functional architecture of the ACS-GW has been presented in [2] and is depicted in the scheme in Figure 6 (from [2]).

Figure 6: The principle architecture of the ACS-GW.



We can distinguish between two sides of the ACS-GW: the IP-bearers side and the application side. In both sides we can always distinguish between user plane and control plane functionalities.
On IP Bearer side, we observe that ACS GW interfaces with radio communication bearers at IP level only. The main goal of ACS GW is to setup communication tunnels over the IP bearers. The ACS GW (intelligently) controls bearers and monitors them. Tunnels are created and managed by the ACS GW over the active bearers. Data are transferred from applications over tunnels.
The interfacing to the IP Bearers is achieved by a logical IP interface which allows the ACS GW to configure the bearer. Depending on the selected communication technology, the logical IP interface could be made available directly by the bearer. In other cases, it could be necessary to create software (glue logic) to realize the interface for control purposes. In some cases, the bearer interface could be not available for the bearer and the glue logic software could not be programmed.
On the application side, the single rail application should be extended/modified to (possibly) interface with ACS control side by means of SIP protocol to negotiate with the ACS GW for the assignment of communications resources as dictated by QoS requirements. Finally, each application can communicate with the ACS GW for data transmission on the user plane.
For “non-ACS enabled applications”, i.e., applications not supporting ACS control protocol and therefore not able to register, manage sessions or subscribe to events, preconfigured tunnels can be configured and provided on start-up of the system. Then, applications can be mapped to these tunnels by VLAN tag, IP-address, ports, etc. via configuration. Support for clients not connected via a gateway (e.g., wayside objects with limited capabilities) is for further study.

The proposed protocol for ACS session management is Session Initiation Protocol (SIP), described in RFC 3261. The SIP protocol supports unified procedures for registration, session and event management across different domains and networks. SIP is an IETF standardized application-layer control (signaling) protocol for:

- creating, modifying, and terminating sessions: it makes use of elements called proxy servers (e.g., the ACS GW) to help route requests to the current location and
- supporting provider routing policies which become applicable in various scenarios e.g., for inter-domain routing in roaming scenarios.

Applications initiate communication sessions end-to-end (from application to application) via the ACS control protocol. Both the on-board and the network side ACS GWs act as proxies for this protocol and are always included in the communication flow. The ACS-GWs maintain a tunnel information table to map the endpoints' identities and IP addresses associated to the designated tunnel as the mandatory information. This ensures appropriate E2E IP routing via onboard and network gateways. In the network gateway, additionally all registration information is maintained by repository. This means the network gateway has control of all actively registered applications. Applications' IP addresses (both on the network and onboard domain) must be unique within the system and routable by the gateways.

3.5 ACS modeling for subsequent studies on application and transport protocols – preliminary analysis

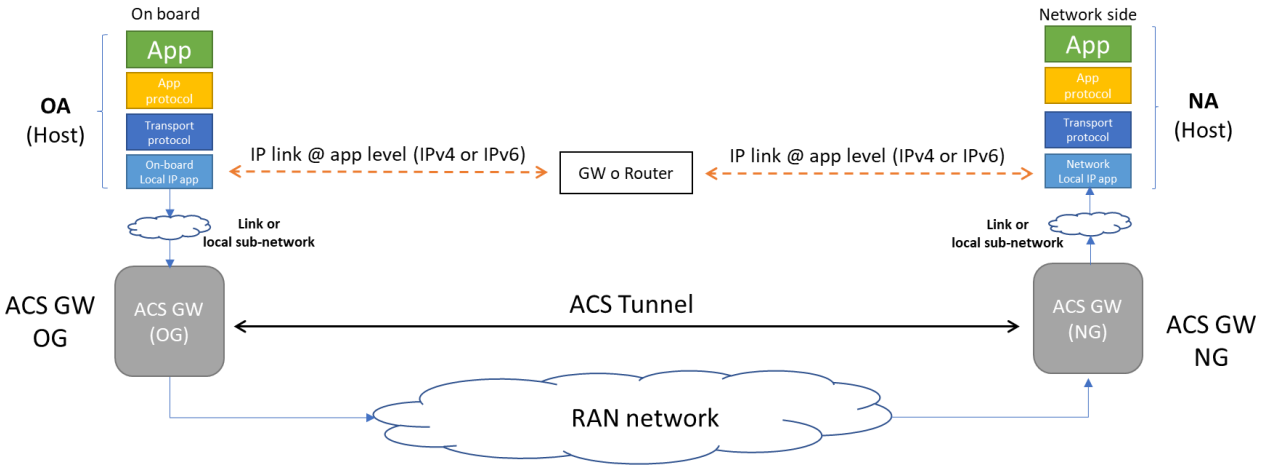
ACS is a communication platform based on the BIP principle (driven by the IP protocol). It can cope with many and somewhat contrasting requirements of railway applications. The ACS GW is the main building block of the ACS network. In general, the ACS-GW network architecture can be of mesh type (ACS-GW are the main nodes, On-board Applications (OA) and Network Applications (NA) are the clients of the main nodes) [2]. In particular, the ACS-GW can create a logical overlay (mesh type) network over the IP protocol layer. The ACS adopts standard, well known and tested IETF protocols. This means ACS "speaks the Intern language" natively. Differently from FRMCS, in ACS there is no need to define the on-board architecture for railway applications. The railway applications specifically designed for ACS should support SIP to interface with ACS for control purposes.

From the summary presented in above, the ACS provides the single/multi bearer underlying communication infrastructure to be used by railway applications to exchange voice/data-based services. Actual investigation on ACS has focused on the assessment of its functionalities mainly concerning the "bearer side". The WP3 activities deals with communications protocols on the "application side" of the ACS-GW focusing, in particular, on the "best" selection of (secure/un-secure) transport and application protocols layers. Furthermore, the ACS GWs allow to inter-connect OAs running on the host(s) in the local onboard networks with the corresponding applications running on the network side (NA) in the cloud or in the edge cloud. In general, it is assumed the ACS-GWs allows to interconnect hosts residing on sub-networks with different IP addressing schemes e.g., one sub-network using IPv4 and the other IPv6.

The main interest in AB4Rail studies focuses on the appropriate selection of the application and transport protocols for the main railway applications. The ACS system provides connectivity to the OA and NA applications to exchange data. With respect to data transmission only OA and NA interfaces the ACS through the application and/or the transport protocols but they are practically unaware of the underlying ACS infrastructure providing data transfer i.e., they are not aware of the specific communication(s) technology(ies) used to transfer data over the network(s). Obviously,

they can exchange control information with the ACS-GW to force the ACS-GW to assign communication resources on the basis of their application class [2].
Taking into account the previous considerations the modeling scheme of the ACS system to be considered for the subsequent analysis and/or transport protocols is depicted in Figure 7.

Figure 7: Principle scheme of the model for the ACS-based communication link



As shown in Figure 7, applications run on top of the application protocol or directly above the transport protocol (this depends on the specific application). In general, we assume the client application runs on the host belonging to the on-board network, which is directly connected to the ACS-GW by means of a local (IP-based) network. IP packets generated on application side are forwarded to the ACS-GW and encapsulated in one (or more) ACS tunnels. The number and the type(s) of tunnels created and assigned to transport the application data depends on the application-to-ACS GW negotiation. We assume the ACS-GW autonomously manages the setup, the maintenance and the dropping of the tunnels. Moreover, it assigns resources to the applications on the basis of the results of negotiation carried out by means of SIP protocol. This last assumption allows to model the link between the OA and the NA (see Figure 7) as an IP link, whose parameters such as latency, packet error rate and packet loss, jitter and available capacity can vary with time, thus influencing the behavior of the upper layer protocols.
It is well known that transport and application protocol layers can severely influence the end-to-end link performance as perceived by the railway application. Main transport mechanisms influencing end-to-end performance can be related to congestion and flow control procedures, which are specific of the selected transport protocol layer. Furthermore, interactions between the application and transport layers not accounting for the specific operations' modes of the two layers can lead to significant inefficiencies to end-to-end data transfer, causing unwanted overhead increase and augmented transfer latency (e.g., refer to the case of HTTP 1.0 and TCP as a very relevant example). Multi-path transport protocols (e.g., Multipath TCP, MP-TCP) have recently emerged to increase transport capacity allowing the railway application to use more communication bearers simultaneously and/or to improve transport reliability/resiliency. In Figure 7 we have also indicated the presence of the gateway (GW)/Router performing IPv4/IPv6 conversion. For the subsequent evaluation of application and transport protocols performance the IP addressing scheme used in the model in Figure 7 is not relevant (i.e., IP addresses are not used and are not visible at transport layer). This means the transport/application protocol analysis can be performed using only one addressing scheme such as IPv4 for example. However, it is well known that IPv4-IPv6 conversion

can influence the values and the variability of the performance parameters describing the behavior of the IP-link connecting the OA and the NA. The possible influence on IP link performance due to IPv4-IPv6 conversion will be studied in AB4Rail WP3 and will be the subject of a separate deliverable (Deliverable D3.2). Results from this activity will be used to account for the effects of IPv4-IPv6 conversion in the IP-link behavior and then on the corresponding parameters.

4. Railway Applications

4.1 Main Critical Rail applications to be considered in AB4Rail

In principle, the ACS has been conceived to provide connectivity to every type of railway application in different scenarios, encompassing the entire rail ecosystem and including: mainline/highspeed lines, regional, metro/urban, station, freight. Specifications of railway applications and services envisaged in ACS are inherited by FRMCS user requirements specifications [3] and extended in [2] to incorporate other users and stakeholders, also including one additional use case for the passengers. The key use cases indicated in X2Rail-1 deliverable D3.1 can be grouped as (repeated from [3]):

- Signaling
  - *Signaling application(s) are used for communication of safety-related information regarding movement of trains. This comprises data communication service between an application on-board a train and a corresponding ground-based application used to allow trains to move a distance interval with a speed based on track limitations and known braking capabilities of the specific train. This implies knowledge of the current position of the train at all times. Disruption of the communication may cause degradation or complete stopping of a moving train. Examples of modern signaling applications in use are: ETCS and CBTC. Both ETCS and CBTC provide similar primary functionality but currently use different principles for achieving communication;*
- Critical Voice
  - *Critical Voice application(s) are primarily used to exchange vocal information between train driver(s) and controller(s) regarding operation and movement of the train. The application can be either user-to-user or multi-user (group call). Group calls are used when the same vocal information is relevant to number of users within a defined geographical area. This service is also used in emergencies to inform driver(s) and controller(s) about possible hazards along the track, e.g., people on (or near) the track, obstacles or damages on infrastructure that may cause serious problems. The outcome of such communication can be the degradation or the complete halt of train operation either partially or covering a large area. In such emergencies, critical voice communication has the highest priority. Due to the nature and use of critical voice services, special care needs to be taken in order to achieve high reliability with low latency and setup time, including even at high speed of mobility;*
- Critical Video
  - *Critical Video application(s) are used where highly reliable and available real-time video streams need to be transferred either from train to ground or from ground to train, and where the unavailability of the video transmission may have serious impact on train operation or damage to person(s) or infrastructure component(s). Typical use cases include automatic obstacle detection for ATO, monitoring of doors inside trains as well as people on platforms. Depending on the use case, different video quality requirements are applicable;*
- Critical Data
  - *Critical Data application(s) can be seen as a generic categorization of data services, where reliability and availability are of utmost importance. This is either in order not to jeopardize the availability of train operation, or to prevent damage to person(s) or infrastructure component(s). It should not be seen as a safety service (no Safety Integrity Level (SIL) class) in itself, although it is usually intended for carrying safety related information supporting SIL4 applications. ATO and Train Integrity are examples of future critical data applications.*

The AB4Rail project focuses on the studies of transport and application protocols concerning the

critical rail applications. **Performance** rail applications help to improve the performance of the railway operation, such as train departure, telemetry, etc. and **Business** rail applications are mainly oriented to passengers and they depend on the specific rail scenario. If necessary, in some scenarios, performance and business applications will be considered when they can create additional traffic to be conveyed on the ACS communication infrastructure. We implicitly assume performance and business applications can run on the commonly envisaged transport and application protocol layers and that they do not need to be further optimized. Furthermore, we also restrict the set of critical railway applications to be analyzed to those concerning the on-board to trackside communications. To this purpose, the reader should refer to CONNECTA projects (https://cordis.europa.eu/project/id/730539).

From the test trials of ACS detailed in [4] the main railway (critical) applications that have been considered for testing and that will be also considered in AB4Rail are:

a. **ETCS/ERMTS** that can be classified as critical application for train movement and safety; we are only interested in ETCS level 2 or level 3 systems since they adopt a radio technology to exchange data between the train (i.e., periodic position report (PR) messages) and the RBC. In fact, in ETCS Level 2 Movement authorities (MA) messages are given to the train driver in order to allow the train to move itself on the track and the most of the signals are displayed in the train borne cabin, substituting the lateral traditional signals. This allows to operate without a lateral trackside signaling. The ETCS Level 3 provides an implementation of full radio-based train spacing. Fixed track-release signaling devices, such as balises, are no longer required. In ETCS, the RBC (Radio Block Center) is the safe central trackside equipment of the ERTMS/ETCS and it is responsible for the security of all trains running in the area with which a GSM-R (and more in general a) radio communication has been established. In other words, RBC manages the exchange of data required for safe train travel and separation, but only in its area of governance responsibility.
b. **Voice:** in relation to the purposes and context of the requested voice service (VS) the single VS could be classified as critical, performance or business. This leads to different assignment of priority and traffic class ID in the ACS thus impacting on the way the voice data are scheduled for transmission, call setup time and so on. Voice based services are pervasive in all the FRMCS+ACS user requirements and in accordance with their relevance these can be classified into the critical or business categories with very different requirements in terms of latency, call setup time, etc. Independently of the specific voice service, VoIP seems to be the main (and maybe the only one) well proven technology to implement voice transmissions over IP-based packet data networks. The ACS abstracts the transmission technology to the application i.e., it provides connectivity with specified requirements to each application. Depending on their importance and classification, the different voice-based applications (using VoIP or any other standard (if any!)) correspond, from the ACS point of view, to one or more data streams/flows. Each one is characterized by different traffic class ID and then by different requirements in terms of transport performance (refer to Table 2) and priority level to be guaranteed by the ACS system. From the FRMCS user requirements [3] the main list of the critical voice communication applications is reported in [3] (Table page 73):
    i. On-train outgoing voice communication from the driver towards the controller(s) of the train;
    ii. On-train incoming voice communication from the controller towards a driver;
    iii. Multi-train voice communication for drivers including ground user(s);
    iv. Banking voice communication;
    v. Trackside maintenance voice communication;
    vi. Shunting voice communication;

vii. Public emergency call;
viii. Ground to ground voice communication;
ix. Voice Recording and access to the recorded data;
x. On-train outgoing voice communication from train staff towards a ground user;
xi. On-train incoming voice communication from a ground user towards train staff.

For each one of the previously indicated voice applications, the FRMCS provides an indication of performance requirements in terms of the following attributes: Type, Symmetry Up/Down, Distribution, Latency, Bandwidth, Reliability, Setup Speed. In accordance with the number of parties involved in the conversation, we can distinguish between unicast or multicast voice communications. The values of each attribute are specified in [3] and have been inherited and adapted to ACS for the assessment of the ACS Traffic ID classes and priorities. They are summarized in the next Section. It should be taken into account that the single voice service is typically accompanied with support services carrying information concerning, for example the status (active/inactive) of the callers, voice recording the call, etc. In general, each railway voice service based on VoIP should be always considered as enriched VoIP call.

c. **Video and data services**: similar to voice services even in this case video services could be classified as critical or business. Under the generic name of "data services", include video-based, IoT and web browsing services are included.
IP-based CCTV systems are the defining factor for modern train surveillance systems. In addition to ensuring passenger safety, these systems have expanded their scope and are now providing mission-critical information that will increase operational efficiency. IP-based cameras and NVR computing platforms are now being deployed in more and more locations throughout the rail system. To this purpose, the FRMCS has introduced video-based communication applications that, depending on the specific application where CCTV can be used, can belong to critical, performance or business categories. In general, an optical fiber backbone through a multiservice IP-network supports centralized applications and operation management tools, and network security (CCTV surveillance and alarm monitoring), and passenger information system. The possibility of transferring IP-CCTV videos over wireless networks enables to extend their usage to provide on board connectivity and services.
The real time video communication application facilitates the data communication for real time transmission of video images)[3] for critical railway operation, including the control of camera movements and zoom. Critical Real time video images are considered to be an effective mitigation measure in relation to hazards that may not be detected otherwise by the train control system. In addition, real time video images can enhance operational performance of the railway system when used to support the end user within the target environment. Control of camera movements and/or zoom can be used when supported by the camera system.
This application can be used for example for [3]:
i. ATO and ATC
ii. Automated detection of objects on or near tracks in the context of, e.g., GoA3/GoA4 operation
iii. Supervision of platform and tunnels (either by a remote human user or in an automated way)
iv. Generation of Alarms (e.g., supervision of railway track, doors, train, etc.)

[3] "Video images" may also refer to images coming from other sources, e.g., lidar and/or radar sensors.

v. Smoke detection
vi. Protection of passengers
vii. Prevention of vandalism
viii. To transfer video image in parallel with voice communication (e.g., during Railway Emergency Communication)
ix. Video-based remote control of trains (e.g., in the case of degraded mode operation in GoA4 operation)

d. **Machine type communications**: these types of services can be related to the communications of IoT devices within/participating in the on-board train network, including sensors, devices for telemetry etc. willing to communicate data to the railway data center. There are several communication applications aiming to transmit data collected by on-board sensors for different applications. In [3] some critical IoT communication applications have been considered:
   i. Working alone: the system shall be able to monitor the status (location, movements, health, etc.) of a user working alone. Once the application is active, the application can trigger voice and/or data communication applications based on the status of the worker. The status of the worker is collected thanks to different external devices or capabilities in the terminal (biometric signals, speedometer, verticality of the terminal, etc.) and it is interpreted by the application.
   ii. Train integrity monitoring data communication: the train integrity monitoring system shall have a reliable communication bearer in order to ensure safety-related data be transferred between the components monitoring train integrity. The communication infrastructure (ACS or FRMCS) shall provide the communication bearer for this data exchange. This application allows the monitoring of the train integrity status, to ensure the integrity of a train during railway operation. For example, with ETCS level 3, letting the CCS on-board sub-system and/or the ground system to apply the foreseen safety reaction when the train integrity status is lost or unknown.

Performance communication applications

The main communication applications that can be of interest for evaluating the ACS performance at protocol level because they can share the same ACS communication infrastructure of the critical services thus creating background traffic over the ACS system are listed in the following (taken from [3](Table page 73).

- Multi-train voice communication for drivers excluding ground user(s)
- On-train voice communication
- Lineside telephony
- On-train voice communication towards passengers (public address)
- Station public address
- Communication at stations and depots
- On-train telemetry communications
- Infrastructure telemetry communications
- On-train remote equipment control
- Monitoring and control of non-critical infrastructure
- Non-critical Real time video
- Wireless on-train data communication for train staff
- Wireless data communication for railway staff on platforms
- Train driver advisory - train performance
- Train departure data communications
- Messaging services

- Transfer of data
- Record and broadcast of information
- Transfer of CCTV archives
- Real time video call
- Augmented reality data communication

Business communication applications

The main business communication applications for railway have been identified in [3] and are listed in the following points:

- Information help point for public
- Emergency help point for public
- Wireless internet on-train for passengers
- Wireless internet for passengers on platforms

Classification of railway applications in accordance with the ACS view

The SIP-based ACS control interface allows application to negotiate communication parameters with the ACS-GW. The SIP protocol is used because it supports unified procedures for registration, session and event management across different domains and networks. SIP is an IETF standardized application-layer control (signaling) protocol for creating, modifying, and terminating sessions. The network architecture for SIP signaling makes use of proxy servers to route requests to the current location and it supports provider for routing policies which becomes applicable in various scenarios e.g., for inter-domain routing in roaming scenarios.

In [2] the format and the contents (fields) of two important messages exchanged between the railway application and the ACS-GW have been defined for: the ACS authorization request and the ACS session management. In the following we only refer to Session Request, used whenever an ACS tunnel is established. The **ACS Session Request** parameters have been listed in [2](page 77) and, for the convenience of the reader, they are reported (copied) here in Table 2.

Table 2: ACS Session Request parameters

Information element	Status	Type	Description
Interface version	M	*** for future use ***	
Target name	M	*** for future use ***	
Identification tag	M	sourceIP;port dest IP:port	Tag for UP data packets for routing to appropriate tunnel.
ACS KPI information	M	Msec	Time for information replies Refer to ACS_KPI

			- Push / Change - regular
Maximum outage	M	Msec	Allows ACS to decide for additional bearers for redundancy.
Priority	M	0,1,2,3,4,5	Priority levels to control preemption of ACS tunnels. 0 has highest priority. ACS may preempt lower priority services in case of shortage of bearer resources for ACS tunnels. Can be dynamic per session request e.g. regular call upgrade to emergency call. Refer to Note 1.
ACS traffic class ID	M	0,1,2,3,4,5,6,7	Defines per application category values for latency, reliability, session establishment. Note 2.
Bearer costs allowed	M	Yes No	Indication of application is allowed to use commercial bearer service. Yes – costs are allowed.
Transport protocol	M	TCP, UDP	
Throughput characteristics	M	Constant, average, maximum,	Constant: is considered to be minimum (GBR) Average: application transfers data on demand in peaks – no constant bitrate reserved. (=GBR with variable usage) Maximum: Limit the resource to a specific maximum.
Throughput uplink Note1	M	kbps	Required? Or can be addressed by traffic class…?
Throughput downlink Note1	M	kbps	Required?

Three important fields in the ACS Session Request message have been evidenced and further specified in [2]. The throughput is defined at application IP level and not considering ACS overhead. An example of the priority classes has been suggested as reported in the following points:

- 0: Emergency Voice
- 1: Voice, CBTC, ETCS
- 2: CCTV
- 3: Cashless payment
- 4: Infotainment
- 5: Internet on board

Finally, the ACS Traffic Class IDs have been defined in accordance with the FRMCS requirements and are listed in Table 4 in [2] and are repeated (copied) here in Table 3.

Table 3: list of class IDs

Traffic Class ID	Referenced FRMCS Application Category	Latency	Reliability	Setup Time
0	ACS control plane	(FFS) For further study	FFS	FFS
1	Voice	Low	Normal	Normal
2	Critical Voice Critical Video	Low	Low	Immediate
3	Video Critical Data (legacy apps) Non-Critical data	Normal	Low	Normal
4	Very Critical Data	Ultra-Low	Ultra-Low	Immediate
5	Critical data	Low	Ultra-Low	Immediate
6	Messaging	Best-Effort	Low	Normal
7	File transfer	Best-Effort	Normal	Normal

Looking at the specifications in [2], it is not difficult to assign at least one priority class to each application category and then to specific application during the Session Request phase.
To define latency and reliability, the requirements from FRMCS have been considered. They are reported in Table 5 in [2] and are repeated here in Table 4.

Table 4: FRMCS definitions inherited in ACS

Link quality	FRMCS - Functional Requirement	FRMCS – System Requirement	Service Attribute value
Latency	Low	Ultra-Low	≤10ms
		Low	≤100ms
	Normal	Normal	≤500ms
		Best Effort	>500ms
Packet Loss (%)	High	Ultra-Low	1-99.9999%
		Low	1-99.9%
	Normal	Normal	1-99%

The requirement on the setup time of a communication session is essential because of rapid communication that is required by some safety critical applications. Communication session setup encompasses the value of the elapsed time between the communication establishment request and the indication of successful communication session establishment. The setup time is defined in FRMCS as follows and it has been inherited by ACS. The FRMCS User requires two classes:

- Immediate communication session: the FRMCS-user requires immediate setup of the communication session. The duration of the immediate communication session establishment shall not exceed **1 second**.
- Normal communication session: setup time range does not harm the use of the application. The time

duration of the normal communication session establishment shall not exceed **3 seconds**.

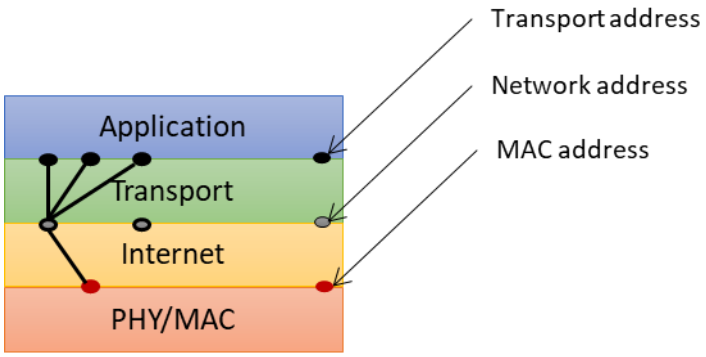
The railway services to be supported in ACS fully inherit and extend the services already specified in FRMCS and they can be grouped in the ACS traffic classes. It should be noted that the requirements listed in Table 3 are specific for each class of railway applications and do not refer to the usage of any specific underlying communication technology. It is the task of the ACS-GW to "find" and to allocate all the necessary communication resources to meet (at the best) the requirements of the railway applications. These requirements are specified in the Session Request phase and, in particular, the ACS-GW should account for the specified priority and of the traffic class ID. This opens a new and very interesting research line, which is out of the scope and well beyond of the AB4Rail objectives, concerning the intelligent/adaptive resource management in ACS.

5. Existing transport protocols

In this section we report the most popular transport protocols used in current networks. After their description, we try to classify them by comparing their features.
We consider the following features and for each of them we provide a definition to help the comparison [8]:

- Congestion control. The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Congestion control is the set of mechanisms used for monitoring the amount of data injected in the network and for regulating it, as to keep traffic levels at an acceptable value.
- Flow control. It is the group of mechanisms to keep a fast sender from swamping a slow receiver with data. Differently by congestion control that may involve more than one or two host, flow control involves only the sender and a receiver.
- Error control. It is the group of mechanisms to detect if an error occurs in the transmission (e.g., packet lost, corrupted packets). Some actions can be taken in order to recover the lost data.
- Connection establishment. Some transport protocols need to establish a connection in order to manage lose, delay, corrupt, and duplicate packets due to the transmission in the network.
- Addressing. The transport protocol with this functionality is able to give the possibility to define transport addresses to which processes can listen for connection requests. Usually, these addresses are called Transport Service Access Point (TSAP) or ports. Processes running on both application clients and servers can attach themselves to a local TSAP to establish a connection to a remote TSAP.
- Checksum (for misdelivery). Protocols may add some information (a checksum) in the header in order to detect possible corrupted bits, caused from the transmission in the network. Some protocols calculate the checksum by considering source and destination ports and source and destination network addresses as input. This allows to avoid misdelivery when the checksum is not corrected, thus discarding the arrived packet.
- Multiplexing. This is the possibility to transmit several application processes through only one network address (see Figure 8).

Figure 8: multiplexing and inverse multiplexing.



- Connection-oriented. It tells if the transport protocol is connection-oriented or connectionless. In the first case, the transport protocol has to establish a connection and exchange some parameters on it.
- Casting. It refers to the type of link connecting the source and one or more destinations. Possible transmissions are: unicast (point-to-point), multi-cast (point-to-multipoint), broad-cast (from one point to all destinations), any-cast (from the source to any (only one) destination).

- Stream/message-oriented. It defines the amount of data that the transport protocol receives from the application and send it to the network. In the stream case, the transport protocol receives a stream of byte and it transmits according to the network limitation. In the message case, application passes data as a message, thus transport protocol cannot separate them by inserting them into two different segments.
- Reliable. This functionality provides an error control mechanism able to recover each lost packet, even by retransmitting it.
- Data bundling. In case of bundling, data are grouped together and are sent only when the specific size has been received from the application. In this case, delays may occur.

5.1 Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) has been defined by IETF and it is firstly described in RFC 793 [6] defined in September 1981. The RFC introduces it as follows:

> "The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet switched computer communication networks, and in interconnected systems of such networks."

This protocol has been widely adopted worldwide and commonly used for most of the applications, since it has been designed to dynamically adapt to properties of the Internet (e.g., topologies, bandwidths, delays, packet sizes, and other parameters) and to be robust in the face of many kinds of failures. To point out the importance of TCP, many extensions have been defined which are grouped in an RFC guide, RFC7414 [7].

5.1.1 Protocol description

The TPC service is provided when a TPC session is active, i.e., when both the sender and the receiver create two end points or ***socket***. Each socket is identified by a number (or address) consisting of the IP address of the host and the TCP port (or Transport Service Access Point, TSAP). Thus, the TCP session is defined by a four-tuple composed of:

- Source IP address
- Destination IP address
- Source TCP port
- Destination TCP port[4].

A TCP connection is established by the three-way handshake procedure. In Table 5 defined socket primitives are reported [8]. They are used to establish a connection, to send and receive data and finally to release the connection.

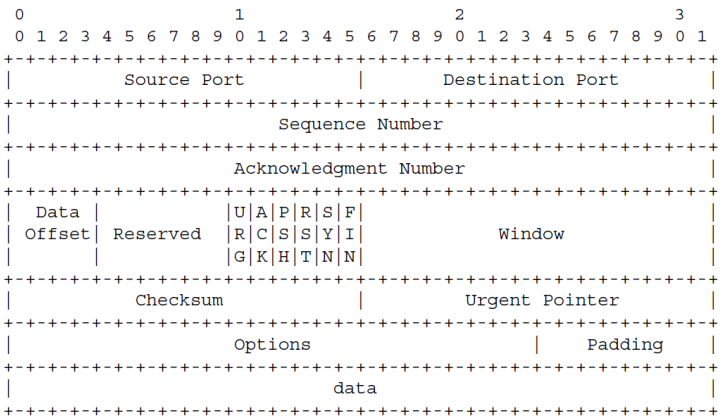
Table 5: socket primitives of TCP

Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

[4] Destination TCP port is often used to indicate the requested service

Then, a TCP session is when two separated TCP connections are established (thus, the connection are full-duplex) in a point-to-point way (i.e., only unicast connections are supported, not multicast or broadcast).
As transport protocol, TCP multiplexes multiple sockets on one network address of the host through the ports. A TCP connection is a byte stream, not a message stream. It means that the data passed by the application are partitioned in bytes and inserted in one or more segments. TCP may send it immediately or buffer it in order to collect a larger amount to send at once, at its discretion. Applications are not aware of it, even if they can flag the PUSH bit in the TCP header (or use TCP_NODELAY option in Windows and Linux) to expedite the transmission. By default, TCP segment partitioning uses Nagle's algorithm [8] for data buffering.
Transmitted data (including 20-byte TCP header) must fit into the minimum between the IP packet size (i.e., 65,512 byte) and the Maximum Transfer Unit (MTU)[5], usually discovered by the Path MTU Discovery (PMTUD), so that it can be sent and received in a single, unfragmented packet.
In Figure 9 it is reported the format of the TCP header.

Figure 9: TCP header format



In the following it is reported the meaning of each field:

- Source Port (16 bits): identifies a local end point of the connection (i.e., the logical address of the source process);
- Destination Port (16 bits): identifies the logical end point of the destination process;
- Sequence number (32 bits): it contains the sequence number of the first byte contained in the TCP payload respect to the sequence number of the connection beginning (numbering in byte not in segment);
- Acknowledgement number (32 bits): it specifies the next in-order byte expected, not the last byte correctly received. This field is valid if the ACK field is set to 1, thus using the piggybacking technique;
- TCP header length or Data Offset (4 bits): it indicates how many 32-bit words are in the TCP header, which is variable (options field);
- Reserved (4 bits): reserved for future use. They must be zero. In RFC 793, they have a 6-bit length, while in RFC 3168 the last two bits have been assigned to control bits;
- Control bits (8 bits): eight 1-bit flags

[5] The MTU is generally 1500 bytes (the Ethernet payload size) and thus defines the upper bound on segment size.

  - ECE: set to 1 by the receiver to signal the sender to slow down
  - CWR: set to 1 by the sender to signal the receiver that the congestion window is reduced then to stop sending ECE (i.e., the ECN-Echo)
  - URG: set to 1 when the Urgent Pointer field has a valid value
  - ACK: set to 1 when the Acknowledgement number is a valid value
  - PSH: set to 1 when the receiver is kindly requested to deliver the data to the application upon arrival instead waiting for the buffer filling
  - RST: set to 1 to abruptly reset a connection, to reject an invalid segment or refuse an attempt to open a connection
  - SYN: set to 1 to establish a connection in the 1° segment during the "three-way handshaking": (a) with ACK=0 (it is a CONNECTION REQUEST); (b) in the replay segment, SYN=1 and ACK=1 (it is a CONNECTION ACCEPTED)
  - FIN: set to 1 by the sender to release a connection (no more data to transmit)
- Window size (16 bits): it indicates how many bytes can be sent starting at the byte indicated in the ACK number field without receiving ACKs
- Checksum (16 bits): it contains info to verify the correctness of the received segment (it is computed by the pseudo-header)
- Urgent Pointer (16 bits): it indicates the offset in bytes (respect to the sequence number) at which urgent data are to be found/delivered to the receiver application
- Options (variable length): it is a way to add extra functionalities not contained in the standard TCP header (e.g.: maximum segment size, Window scale, selective retransmission)
- Padding (variable length): it pads to 32-bit with zeros;
- Data (optional): the TCP segment can be used only for control data or ACKs (data empty) or for both user data and control data (use of control bits)

In order to provide an ordered and reliable data delivery, every byte on a TCP connection has its own 32-bit sequence number. It is used: *i.* to order segments on receipt, *ii.* to identify segments in acknowledgments, and *iii.* to detect unacknowledged segments for retransmission. To detect any possible misdelivery or any data corruption, the TCP segment is equipped by a 16-bit checksum, which provides the integrity check [9, RFC1071].
TCP protocol is based on a sliding window with a dynamic size. When the sender transmits a segment, a timer for each byte starts. When the segment arrives, the receiver TCP entity sends back another segment bearing:

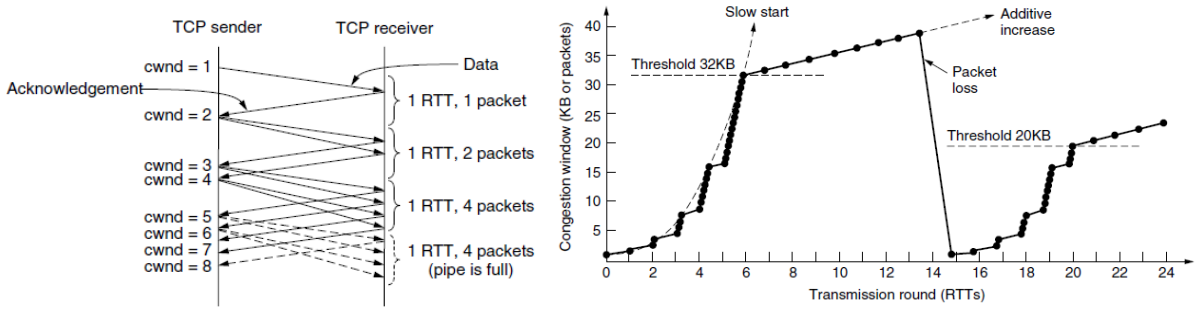
- The acknowledgement number, which is equal to next byte the entity is expecting to receive;
- The amount of data that the receiving entity is able to receive without buffer overflow (namely, receiving window or advertising window). This allows to implement a flow control. The window scale option allows a receiver to use windows greater than 64 kB [10].
- Other data, if any.

If the timer expires before the acknowledgment arrives, the sending TCP entity transmit the segment again.
TCP also provides a congestion control [11, RFC5681], by defining a separate congestion window based always on the sequence number of segments. The operating protocol requires that the congestion window increases if a congestion is absent or while decreases when a congestion Is detected. Usually, a segment loss is considered as due to a congestion. Of course, the sending window at a given point in time is the minimum of the receiver window and the congestion window. Several versions of TCP have been defined along the years, the most based on the concept of Additive Increase Multiplicative Decrease (AIMD). The most common are reported in the

following.

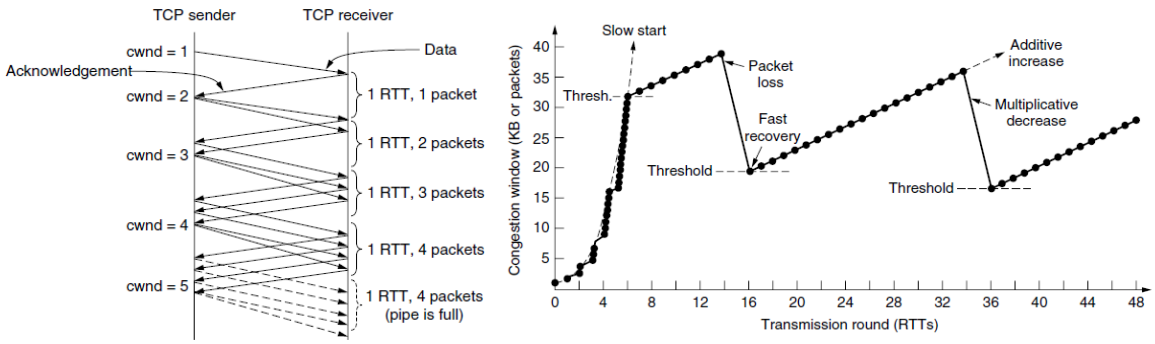
In Figure 10 it is reported the behavior of the cwnd with the **TCP Tahoe** version of the TCP congestion control. For each segment that is acknowledged before the retransmission timer goes off (i.e., after a Round-Trip Time, RTT), the sender adds one segment to the congestion window (slow start). This provides an exponential increase of the congestion window size. After crossing a threshold (the slow start threshold, or ssthresh), the increase rate becomes additive, i.e., the congestion window is increased by one segment every round-trip time. Thus, TCP passes from slow start to additive increase or congestion avoidance. Whenever a packet loss is detected, (i.e., by a timeout), the slow start threshold is set to be half of the congestion window and the entire process is restarted.

Figure 10: TCP Tahoe flow control behaviour.



In Figure 11 it is reported an update version of the previous one, namely **TCP Reno**. Differently from TCP Tahoe, TCP Reno causes again an additive increase after a packet loss. When a packet is lost, the sender receives duplicated acknowledgements. After three loss of packets, TCP sends again the lost segment (fast retransmit). Moreover, if other duplicated acknowledgements continue to arrive to sender, it means that other packets arrive to the destination, reducing the number of packets in the network. Fast recovery is implemented by linearly increasing the congestion window, providing the classical saw-tooth pattern of additive increase (by one segment every RTT) and multiplicative decrease (by half in one RTT).

Figure 11: TCP Reno flow control behaviour.



A slight change occurs in **TCP NewReno** version, which improve the behavior of fast retransmit/fast recovery in presence of partial acknowledgements trying to repair the packet loss [11]. Another change in the control law is the **CUBIC TCP** largely adopted in Linux [12]. In this case, the window size is increased according to a cubic function:

$$W_{cubic}(t) = C \cdot (t - K)^3 + W_{max} \quad \text{(Eq. 1)}$$

where:

- W_max is the window size just before the window is reduced in the last congestion event,
- C is a constant measuring the aggressiveness of window increase in high band-delay product (BDP) networks
- *t* is the elapsed time between two packet losses (i.e., from the beginning of the current congestion avoidance)
- K is the time period that the above function takes to increase the current window size to W_{max} if there are no further congestion events.

K is calculated using the following equation:

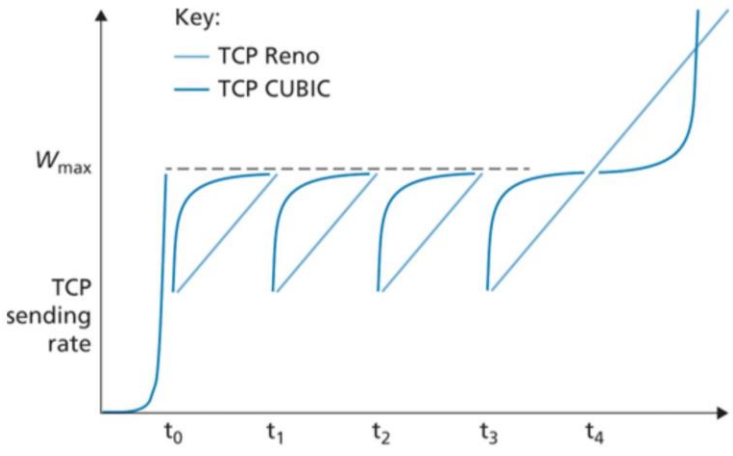
$$K = \sqrt[3]{\frac{W_{max} \cdot \left(1 - {}^{W_{cubic}(0)}/_{W_{max}}\right)}{C}} \quad \text{(Eq. 2)}$$

$\beta_{cubic} = {}^{W_{cubic}(0)}/_{W_{max}}$ is the CUBIC multiplication decrease factor, indicating the reduction of the window size when a congestion event is detected.

TCP CUBIC can work in three regions:

- TCP-friendly region, where the congestion window ensures performance similarly to the standard TCP, above all in small BDP networks;
- Concave region, where the congestion window is smaller than W_{max} (i.e., the size before a congestion event occurs). In this region the protocol tries to have the window size equal to W_{max} as fast as possible;
- Convex region, where the congestion window is greater than W_{max} and the protocol tries to increase slowly the window size, since it is searching a new W_{max} ("maximum probing phase").

Figure 12: Comparison of TCP Reno and TCP CUBIC sending rates in congestion avoidance.



Before concluding the TCP description, it is worth to note TCP extensions (sender side-only, receiver side-only or explicitly negotiated in the connection setup) [7], that can be adopted to optimize transmission for a given scenario and based on the TCP instances participating to the communication. The most important are:

- The TCP Selective Acknowledgment (SACK) [13, RFC2018]: it extends ACK mechanism by providing an earlier identification of which segments are missing;
- The Explicit Congestion Notification (ECN) [14, RFC3168]: if supported by both endpoints, it allows a network node to signal congestion without inducing loss. Alternatively to ECN, the delay-based congestion control scheme that reacts to RTT changes can be used in TCP as an early indication of congestion.

5.1.2 TCP Vegas

Other versions of TCP varying the congestion window control (cwnd) have been proposed during years, trying to better use the network channel. One of these is the TCP Vega. While TCP Reno or its older version TCP Tahoe continues to increase the cwnd until a packet is lost, TCP Vegas controls its window size by observing the packet RTTs. If observed RTTs become large, TCP Vegas recognizes that the network begins to be congested, and reduces the cwnd window size. If RTTs become small, the TCP Vegas in the sender determines that the congestion in the network has being solved, thus it increases the window size. In congestion avoidance, the window size is updated as [15]:

$$cwnd(t+t_0) = \begin{cases} cwnd(t)+1 & if\ D < \alpha \cdot RTT\_base^{-1} \\ cwnd(t) & if\ \alpha \cdot RTT_{base}{}^{-1} < D < if\ \Delta < \beta \cdot RTT\_base^{-1} \\ cwnd(t)-1 & if\ D > \beta \cdot RTT\_base^{-1} \end{cases}$$

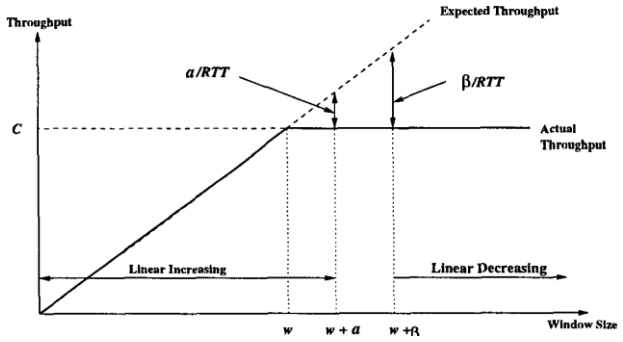
(Eq.3)

where:

- RTT is an observed round trip time
- RTT_base is the smallest value of observed RTTs
- □ and □ are some constant values and D is the difference between the expected and the actual RTT equal to:

$$D = \frac{cwnd(t)}{RTT\_base} - \frac{cwnd(t)}{RTT} \qquad \text{(Eq.4)}$$

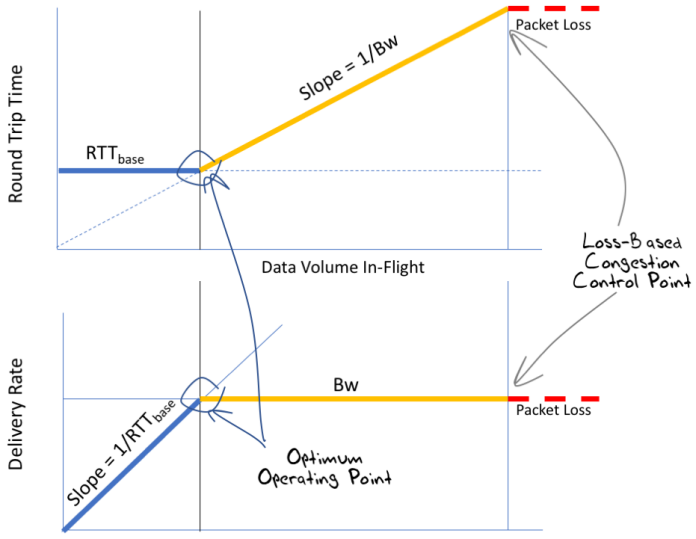
Figure 13: Behaviour of TCP Vegas



5.1.3 Bottleneck Bandwidth and Round-trip propagation time (BBR)

The Bottleneck Bandwidth and Round-trip propagation time (BBR) is a TCP congestion control algorithm developed at Google in 2016. As TCP Vegas, it uses latency or RTT, instead of lost packets as a primary factor to determine the cwnd size and, thus, the sending rate [16].
At the beginning of the congestion avoidance, TCP Reno as well as TCP cubic transmit slower than the current channel capacity available in that moment and increment the cwnd even higher than this value, until a packet is lost. When the allowable channel capacity in the network is overcome (i.e., the bottleneck capacity), buffers of the router queues start to be filled. On the contrary, BBR tries to estimate the sending rate through the RTT and the path bandwidth. In this estimate, BBR does not consider RTTs, which belong to packets sent at a lower rate because the application has limited data amount.
BBR periodically spends one RTT interval deliberately sending at a rate that is a multiple of the bandwidth delay product of the network path. This multiple is 1.25, so the higher rate is not aggressively so, but enough over an RTT interval to push a fully occupied link into a queueing state. Note that this approach is quite different by the estimate performed by TCP Vegas, which works as Eq. 3.
After the BBR has increased the sending rate, the sending entity experiences an increased RTT, due to queueing in the routers (if the available channel bandwidth has not been changed), while the delivery rate is not varied. In [16], this case is at the beginning of the yellow lines. In this case, then the sender subsequently sends at a compensating reduced sending rate for an RTT interval, allowing the bottleneck queue to drain. In case of the current available channel capacity has increased, the sending entity is able to estimate the increase through this probe. Then, the sender can operate according to this new bottleneck bandwidth estimate.

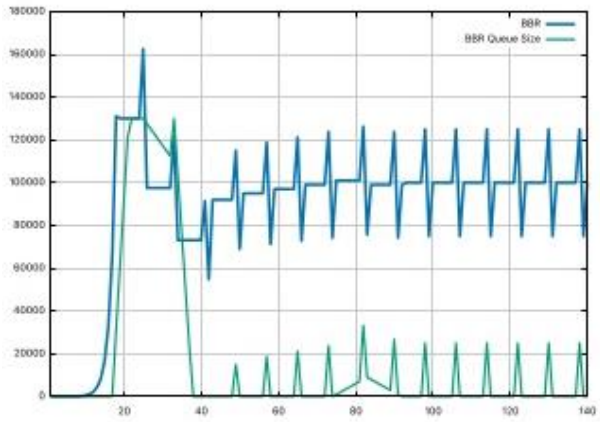
Figure 14: Simple channel model for delivery rate and RTT [16].



By this regular probing operations (i.e., queuing load and drain), BBR is able to know any change in the channel characteristics. The probing allows to increase the sending rate of 25% every 8 RTTs respect to the BDP capacity, causing an exponential increase instead of a linear increase as in TCP Vegas. When concurring with flows using other congestion window evaluation strategies, BBR is able to fast react when one of them experienced a packet loss (thus, reducing its sending rate) and

occupy resources left by them, by sending a steady rate of packets equal to the new bottleneck bandwidth estimate. The ideal behavior of BBR shown in Figure 15.

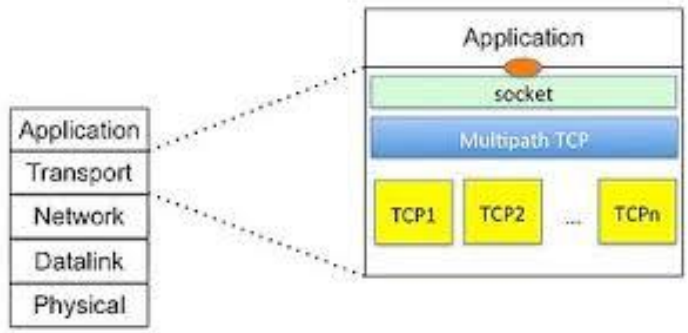
Figure 15: Ideal behaviour of BBR probing operations.



5.2 Multipath TCP (MPTCP)

Multipath TCP (MPTCP) is defined in IETF RFC6824 [17] and it extends the functionalities of TCP by supporting multihoming, mobility and load balancing. As TCP, MPTCP establishes a regular flow between two endpoints (source and destination), in which application data are transmitted as a stream.
MPTCP demultiplexes one byte-based stream into separate paths or subflows. This allows to exploit multihoming (IPv4 and IPv6, for the same TCP session), to provide load balancing, to achieve a higher throughput (in some situations) and to be resilient to network failure and/or handover, since paths can become unusable. Parameter negotiations (such as data sequence number and advertising window) are performed as for regular TCP options.
The management of congestion control in coupled way among subflows may limit the throughput increase. Moreover, scheduler may add further delays, since in-order data delivery to the applications should be guaranteed.

Figure 16: Architecture of the MPTCP.



The MPTCP operating principle is shown in the Figure 16. Note that due to its design MPTCP is indistinguishable from middleboxes similar to TCP

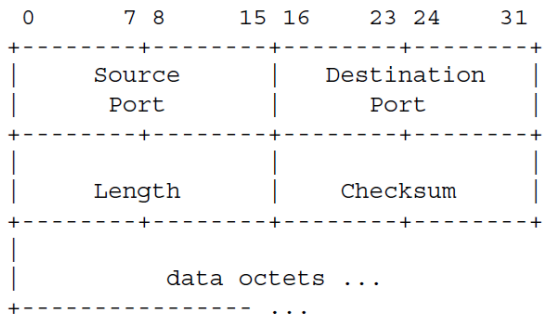
5.3 User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) has been defined by IETF in RFC 768 [18]: it consists of a header of 8 bytes, followed by the payload. Its main scope is to allow applications to send data in IP datagrams without having to establish a connection. The main aim of UDP is to deliver received IP datagrams to the correct application, i.e., to add the source and destination ports (or endpoints), without which the transport layer would not know for each incoming packet. When a UDP segment is received, its payload is handed by the process attached to the destination port, whose attachment is similar to the TCP binding. It is widely adopted in many applications such as Domain Name System (DNS) and Real-Time Protocol (RTP).

5.3.1 Protocol description

UDP is a connectionless and unidirectional protocol. It is able to send messages (usually called UDP datagrams) in unicast, anycast, multicast (IPv4 and IPv6) and broadcast (IPv4). Nor connection setup is required, neither negotiation features.
It preserves the message boundaries. In fact, when an application delivers a message to it, UDP transmits the message into a single IP packet or several IP packet fragments, allowing to send datagrams larger than the effective path MTU. Fragments are reassembled before delivery to the UDP receiver, making this transparent to the user of the transport service. Jumbograms are supported to avoid fragmentation. In anyway, UDP does not provide support for segmentation or any PMTUD procedure.
In Figure 17 it is reported the format of the UDP header.

Figure 17: UDP header format



In the following it is reported the meaning of each field:

- Source port (16 bit) and destination port (16 bit): source and destination port addresses;
- UDP length (16 bit): length of the segment included the header
- UDP checksum (optional): control field of the segment. It is computed from UDP header with UDP checksum set to “0”, from UDP payload and from the pseudoheader including the IP addresses.

The UDP 16-bit checksum field provides the detection of payload errors and misdelivery of packets to an unintended endpoint. When one of the two cases occurs, the received segment is discarded without sending back any indication to the user application. UDP datagrams may optionally haven’t the checksum field when IPv4 is used. This is not allowed when IPv6 is selected. Applications that require an integrity service should do this by themselves.
UDP does not provide:

- Reliability or error correction. It means that no retransmission has been provided if a message is lost. Moreover, applications should manage possible reordered, duplicated and of course lost messages;
- Congestion control or any ECN message. It may cause possible message lost when an overloaded path is used (e.g., shared with TCP connections);
- Flow control. Then, a fast-transmitting application can flood a receiving entity with packets, resulting in packet loss.

Applications that require these features need to provide them on their own or use a protocol over UDP that provides them (see RFC8085, [19]).

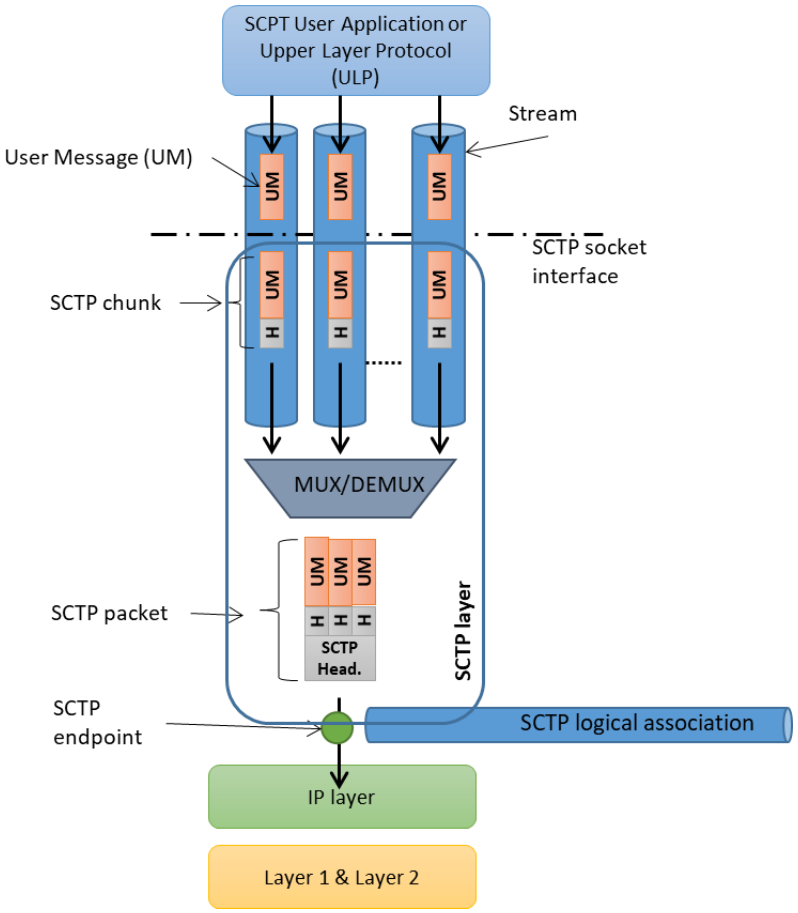
5.4 Stream Control Transmission Protocol (SCTP)

The Stream Control Transmission Protocol (SCTP) is a message-oriented transport protocol specified in RFC4960 [20]. Originally, the SCTP was defined as a transport protocol for telephony signaling (i.e., SS7 messages) to be transmitted over IP networks, also used for WebRTC services. It supports multihoming (i.e., one host with multiple IP addresses and one port number) and path failover to provide resilience to path failures.

5.4.1 Protocol description

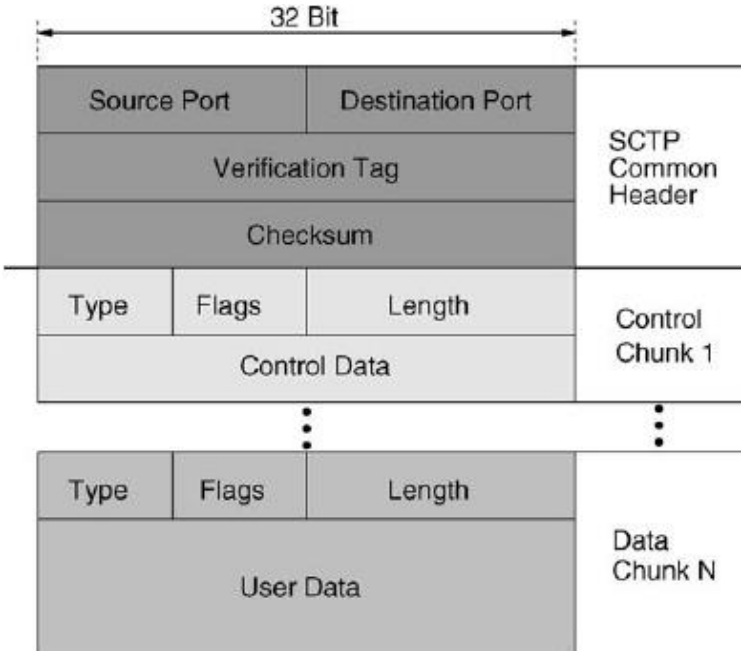
SCTP is a connection-oriented protocol using a four-way handshake to establish an SCTP association and a three-way message exchange to gracefully shut it down. The connection provides endpoints (i.e., ports) to applications, similarly to TCP and UDP, but SCTP only supports unicast.
After a connection (or association) has been established, SCTP allows multiple simultaneous data streams within the single connection (up to 65536 streams), guaranteeing an in-order delivery within each stream. This allows it to minimize head-of-line blocking (i.e., one of the packets is out of order or missing, the stream is blocked pending resolution to the order). The other streams continue to flow, while only the stream that is affected by the error is blocked.
When upper layer requests, user messages can be sent unordered (in this case they are delivered as they are completely received). Moreover, “out of order” or urgent packets with high priority and frequent heartbeat messages as connection test can be sent.

Figure 18: Principle scheme of SCTP [21].



SCTP provides the bundling of the user/application messages into a single SCTP packet. Boundaries of the single messages (called chunks) are preserved, thus transmitted in a single stream. Then, the SCTP packet has a common header and several chunks, each one consisting of chunk header and chunk user data. The common header occupies 12 bytes. Chunks can be of Control and of Data. Its header format is reported in Figure 19. In the SCTP header there is a 32-bit checksum for protecting SCTP packets against bit errors and misdelivery of packets. The chunk header says to the destination how to manage that chunk. Further chunk header can be defined so to extend the SCTP functionalities that can be negotiated in the connection establishment phase.
Each chunk is formatted with a Chunk Type field, a chunk-specific Flag field, a Chunk Length field, and a Value field.

Figure 19: SCTP header format



In the following the meaning of each field of the Chunk format is reported.

In particular:

- Chunk Type (8 bit): it identifies the type of information contained in the Chunk Value field.
- Chunk Flags (8 bit): The usage of these bits depends on the Chunk type as given by the Chunk Type field.
- Chunk Length (16 bits): (unsigned integer) This value represents the size of the overall chunk in bytes (included all fields).

The values of Chunk Types are defined as follows:

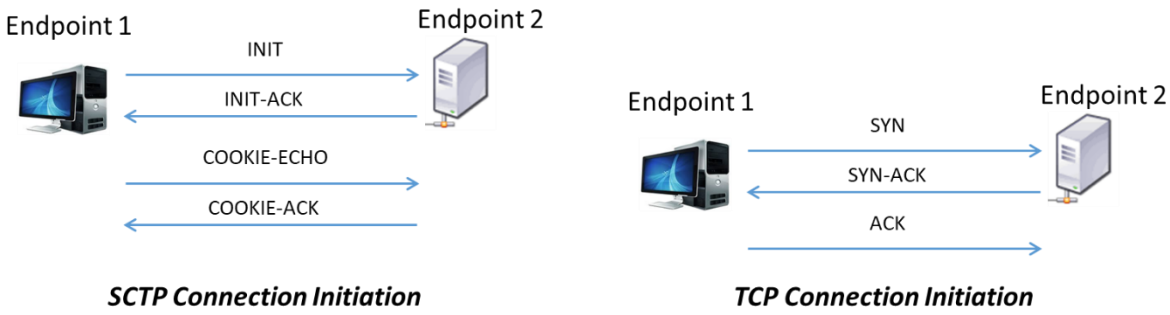
Table 6: values of Chunk Types.

ID	Chunk Type	Description
0	DATA	Payload Data
1	INIT	Initiation
2	INIT ACK	Initiation Acknowledgement
3	SACK	Selective Acknowledgement
4	HEARTBEAT	Heartbeat Request
5	HEARTBEAT ACK	Heartbeat Acknowledgement
6	ABORT	Abort
7	SHUTDOWN	Shutdown
8	SHUTDOWN ACK	Shutdown Acknowledgement
9	ERROR	Operation Error
10	COOKIE ECHO	State Cookie
11	COOKIE ACK	Cookie Acknowledgement

12	ECNE	Reserved for Explicit Congestion Notification Echo
13	CWR	Reserved for Congestion Window Reduced
14	SHUTDOWN COMPLETE	Shutdown Complete
15 to 62		Available
63		Reserved for IETF-defined Chunk Extensions
64 to 126		Available
127		Reserved for IETF-defined Chunk Extensions
128 to 190		Available
191		Reserved for IETF-defined Chunk Extensions
192 to 254		Available
255		Reserved for IETF-defined Chunk Extensions

Differently by TCP, SCTP uses a four-way handshake, whose working is reported in Figure 20a (TCP connection initiation is reported in the same figure for comparison purposes). The SCTP client initiates communications with an INIT packet. The server acknowledges with the INIT-ACK packet and a cookie (unique context that identifies the connection). The client then sends the server's cookie back to the server[6] and the server acknowledges the COOKIE-ECHO with a COOKIE-ACK.

Figure 20: Comparison of connection initiation in SCTP (a) and TCP (b) [21].



(a) (b)

Messages larger than MTU are fragmented at the sender and reassembled at the receiver before their delivering to higher layers. MTU discovery or probe packets can be used for PMTUD.
Addresses at IP layer (it can be a mix of IPv4 and IPv6 addresses) are negotiated during the handshake phase as well as the number of streams. Addresses can be reconfigured during the lifetime of the association, as specified in RFC5061 [22] and in RFC4895 [23], allowing the mobility at transport layer. The streams can be dynamically reconfigured in terms of reset or extension as specified in RFC6525 [24]. Furthermore, the selection of sending stream, the scheduling of algorithms and possible priorities are reported in [25], also including the reliability for user messages in RFC3758.
SCTP applies a TCP-friendly congestion control (see RFC4960), a sliding window flow control and

[6] The client can also send additional information after the COOKIE-ECHO.

immediate requests for acknowledging a message (see RFC7053).
Concerning security, TLS is usually used over SCTP (see RFC3436). Nevertheless, not all functionalities are supported (e.g., unordered delivery and chunk reliability), thus DTLS over SCTP is also specified in RFC6083. Other solutions are also possible concerning the use of NAT and UDP encapsulation of SCTP messages.

5.5 Datagram Congestion Control Protocol (DCCP)

The Datagram Congestion Control Protocol (DCCP) is a version of UDP equipped with congestion control, thus providing unicast connections without reliability as defined in RFC4340 [26]. It also implements the following functionalities: feature negotiation, TRR evaluation, PMTUD and path state management (which can be selected by the DCCP application). DCCP is suitable for interactive applications, streaming, on-line games and applications that transfer large amount of data. Upper-layer protocols on top of DCCP include DTLS [RFC5238], RTP [RFC5762], and Interactive Connectivity Establishment / Session Description Protocol (ICE/SDP) [RFC6773].

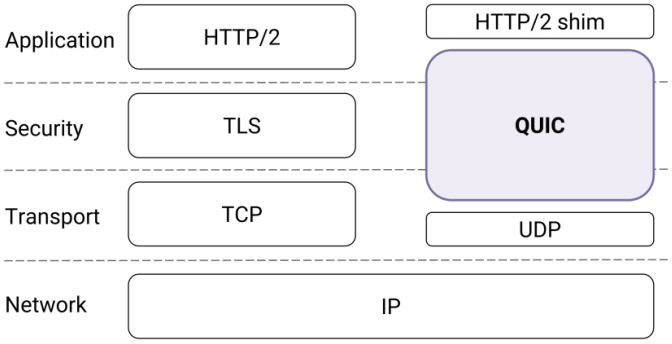
5.5.1 Protocol description

DCCP is a connection-oriented protocol, providing unicast services. During the connection setup, performed through a three-way handshake, client and server can negotiate several functionalities. Moreover, it provides middleboxes with information about the intended use of the connection by the applications of DCCP.
As in TCP and UDP, the established connection is identified by the four-tuple: local and remote IP addresses and local and remote port numbers. DCCP is a message-based protocol generally with size fitting into IP payload. Each message is identified by a sequence number used to know acknowledged and unacknowledged messages, to measure RTT, etc… Unacknowledged messages are not retransmitted. Moreover, it provides unordered delivery. Messages can be protected by a Cyclic Redundancy Check (CRC) field, to enable endpoints to detect application data corruption.
DCCP supports congestion control mechanisms (not just one as in TCP), negotiated during the connection setup. Examples are TCP-like, TCP-friendly, TCP-friendly for small packets and some others (see http://www.iana.org/assignments/dccp-parameters). Receiver flow control is supported, and it is implemented using the slow receiver function. A DCCP Reset packet may be used to force a DCCP endpoint to close a session, aborting the connection.

5.6 Quick UDP Internet Connections (QUIC)

As reported in
Figure 21, Quick UDP Internet Connections (QUIC) protocol is between the application layer and transport layer. Nevertheless, it is hard and could be misleading trying to classify QUIC as a pure transport protocol [26]. QUIC has been designed by Google to improve performance for HTTPS traffic. Since it is a recent transport protocol (presented in 2013), it can be easily and continuously evolved and rapidly deployed in existing systems and networks. QUIC runs in the user space and it replaces most of the traditional HTTPS stack such as HTTP/2, TLS, and TCP as highlighted in Figure 21.

Figure 21: QUIC protocol inserted in the traditional HTTPS stack

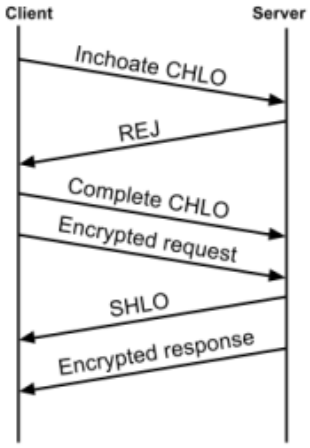


QUIC tries to solve the necessity to reduce the latency of web browsing applications as well as the provision of a secure traffic without adding further delays. QUIC is deployed above UDP. From one side, this allows to traverse middleboxes limiting the network ossification. From the other side, it allows to insert functionalities of transport protocols easy extendible, completing UDP itself.
QUIC is a connection-oriented protocol that creates a stateful interaction between a client and a server [28]. It provides also encryption (both authentication and confidentiality), by using known server credentials on repeat connections and by removing redundant handshake-overhead at multiple layers in the network stack, thus minimizing handshake latency. Public key cryptography is used. Finally, due the connection identifiers, a client can transfer the connection to a new network path after the network topology has changed or NAT has mapped new address binding.
The main features of QUIC protocol are summarized in the following points.

5.6.1 Protocol description

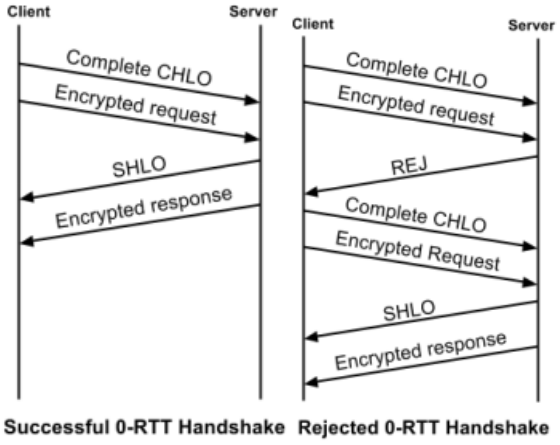
QUIC provides applications (or application protocols) of a connection, which can exchange data via ordered and byte-based streams. Streams based on a lightweight data-structuring abstraction may be unidirectional or bidirectional according to whether one or both endpoints are allowed to send data.
Similarly, to other protocols (e.g., SCTP and HTTP/2), QUIC eliminates head-of-line blocking delays by using streams, which are multiplexed within a single connection so that loss of a single packet blocks only streams with data in that packet. Moreover, QUIC provides congestion control and an error control function to avoid network congestion as well as to implement a reliable delivery, respectively [29].
Connection establishment: Each connection starts with a handshake phase, during which the two endpoints establish a shared secret using the cryptographic handshake protocol. QUIC combines cryptographic and transport handshakes, by multiplexing multiple requests/responses over a single connection, thus reducing the RTTs. We distinguish between initial handshake (1-RTT handshake) and 0-RTT subsequent handshake. The initial 1-RTT handshake is depicted in Figure 22.

Figure 22: initial 1-RTT handshake



At the beginning, the client has no information about the server, thus it performs an initial handshake by sending inchoate client hello (CHLO) message to the server to elicit a reject (REJ) message. The REJ message contains some information, such as the client IP address, the server timestamp, the Diffie-Hellman public value, the authenticating certificate, the signature of the server config using the private key and the source-address token. These information are used by the client in the subsequent sent message "*complete CHLO*", containing the token, the IP address and the client's ephemeral Diffie-Hellman public value. Now, the client is in possession of initial keys for the connections and it is free to start sending application data to the server. Once the handshake is complete, endpoints are able to exchange application data freely.
When the client wants to establish an encrypted connection with the same server, it does not need any further additional round trips and data can be sent immediately following the client handshake packet without waiting for a replay from the server. As shown in Figure 23, the client can start sending data encrypted with its initial keys before waiting for the server's replay to the complete CHLO, thus exploiting the 0-RTT handshake with reduced latency for data. If the handshake is successful, the server returns a "server hello" (SHLO) message Figure 23 (left), otherwise the "REJ message" is sent again from the server to the client as shown in Figure 23 (right).

Figure 23: initial 0-RTT handshake



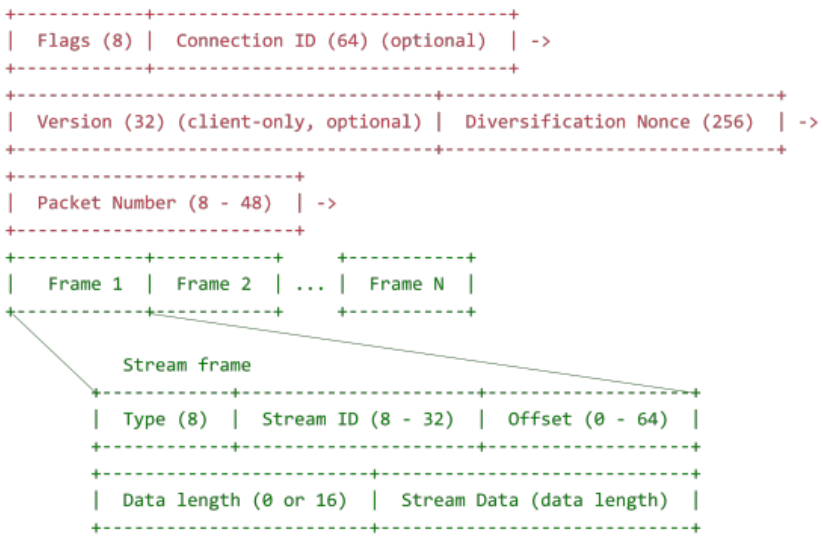
Stream multiplexing: during the connection establishment, connection ID is selected above all to ensure that changes in addressing at lower protocol layers (i.e., UDP and IP) do not cause that QUIC packets are delivered to the wrong endpoint. An endpoint is identified by the set of URI scheme, hostname, and port number (RFC6454). QUIC supports multiple streams within a connection, ensuring that a lost UDP packet only impacts those streams, whose data was carried in that packet. This eliminates the head-of-line blocking typical of byte-oriented transfer protocols (e.g., TCP). Application messages are fragmented into frames sent within one or more than one stream, subject to flow control constraints and stream limits.
Streams are byte-oriented and identified within a connection by a numeric value, referred to as the stream ID. A stream ID is a 62-bit integer (0 to 2^62-1) that is unique for all streams on a connection. Usually, odd IDs are used for client-initiated streams, while even IDs are used for server-initiated streams (see Figure 24). Note that QUIC can send interleaved data between streams but it does not order bytes on different streams.

Figure 24: Stream types in QUIC.

Bits	Stream Type
0x0	Client-Initiated, Bidirectional
0x1	Server-Initiated, Bidirectional
0x2	Client-Initiated, Unidirectional
0x3	Server-Initiated, Unidirectional

The management of the streams in terms of creation, ending, cancelling, and managing flow control is quite simple in QUIC. For example, a single stream frame can create an “as-yet unused” stream, can use it by sending data and can close it by setting a “FIN” bit on the last stream frame. If either the sender or the receiver determines that the data on a stream is no longer needed, then the stream can be canceled without having to tear down the entire QUIC connection.
The format of a QUIC packet is depicted in Figure 25.

Figure 25: structure of a QUIC packet



The QUIC packet is composed of a common header followed by one or more frames, as shown in Figure 25. QUIC stream multiplexing is implemented by encapsulating stream data in one or more stream frames, and a single QUIC packet can carry stream frames from multiple streams. The rate at which a QUIC endpoint can send data will always be limited also by considering flow and congestion control algorithms. An endpoint must decide how to divide available bandwidth between multiple streams. Fields and their meaning of QUIC packet are:

- Header Form (1 bit). It is equal to 0 (for short packet header) or to 1 (for long packet header);
- Fixed Bit (1 bit). It is equal to 1 (normal) or to 0 (for negotiation);
- Long Packet Type (2 bits). Possible types are: Initial, 0-RTT, Handshake, Retry;
- Type-Specific Bits (4 bits). It is based on the packet type;
- Version (32 bits): The QUIC Version is a 32-bit field that follows the first byte. This field indicates the version of QUIC that is in use and determines how the rest of the protocol fields are interpreted;
- Destination/Source Connection ID Length (8 bits, each). It contains the length in bytes of the Destination/Source Connection ID;
- Destination/ Source Connection ID (0..160 bits, each). The long header contains two connection IDs. They are selected by the sender and the recipient during the handshake to ensure consistent routing;
- Type-Specific Payload (..). It is based on the packet type;
- Packet number (8..48 bits). This number is used in determining the cryptographic nonce for packet protection.

The payload of QUIC packets, after removing packet protection, consists of a sequence of complete frames. Each frame begins with a Frame Type, indicating its type, followed by additional type-dependent fields. Types of frame are reported in Figure 26.

Figure 26: Frame types in QUIC packet.

Type Value	Frame Type Name
0x00	PADDING
0x01	PING
0x02 - 0x03	ACK
0x04	RESET_STREAM
0x05	STOP_SENDING
0x06	CRYPTO
0x07	NEW_TOKEN
0x08 - 0x0f	STREAM
0x10	MAX_DATA
0x11	MAX_STREAM_DATA
0x12 - 0x13	MAX_STREAMS
0x14	DATA_BLOCKED
0x15	STREAM_DATA_BLOCKED
0x16 - 0x17	STREAMS_BLOCKED
0x18	NEW_CONNECTION_ID
0x19	RETIRE_CONNECTION_ID
0x1a	PATH_CHALLENGE
0x1b	PATH_RESPONSE
0x1c - 0x1d	CONNECTION_CLOSE
0x1e	HANDSHAKE_DONE

Loss recovery: concerning the error control and error recovery functionality, each QUIC packet carries a new packet number, including those carrying retransmitted data. This protocol choice overcomes the issue requiring the necessity of a separate mechanism to distinguish the ACK of a retransmission from that of an original transmission, thus avoiding TCP's retransmission ambiguity problem. The packet number represents an explicit time-ordering, supported by the stream offsets contained in the frames.
QUIC acknowledgments explicitly encode the delay between the receipt of a packet and its acknowledgment being sent, thus RTT estimation and the packet numbers, network RTT estimation, which monotonically increase allows to detect the packet loss.
Flow control: to prevent a fast sender from overwhelming receiver or a malicious sender from consuming a large amount of memory, QUIC implements mechanisms for flow control. The receiver advertises the limit of total bytes it is prepared to receive on a given stream or for the entire connection. Flow control in QUIC acts at two levels:

- Stream flow control: it prevents a single stream from consuming the entire receive buffer on any given stream;
- Connection flow control: it limits the aggregate buffer that a sender can consume at the receiver across all streams.

Similar to HTTP/2, a QUIC receiver advertises the absolute byte offset within each stream up to which the receiver is willing to receive data. As data is sent, received, and delivered on a particular stream, the receiver periodically sends window update frames that increase the advertised offset limit for that stream, allowing the peer to send more data on that stream. Connection-level flow control works in the same way as stream-level flow control, but the bytes delivered and the highest received offset are aggregated across all streams.
Congestion control: The QUIC protocol does not rely on a specific congestion control algorithm. Nevertheless, it provides the necessary feedback to implement reliable delivery and congestion control. Examples are in [29].
QUIC discovery for HTTPS: as a general rule, a client does not know a priori whether a given server implements TCP, UDP or QUIC. Then, when the client makes an HTTP request to an URI for the first time, it sends the request over TLS/TCP. Google servers usually advertise to support QUIC by including an "Alt-Svc" header in their HTTP responses. Due to this information, the client can now attempt to use QUIC in subsequent requests to the same URI. On a subsequent HTTP request to the same origin, the client may send connection two requests, one for a QUIC connection and a second for a TLS/TCP connection. It tries to establish a QUIC connection by delaying connecting via TLS/TCP by up to 300 ms. Whichever protocol successfully establishes a connection first ends up, getting used for that request. If QUIC is blocked on the path or if the QUIC handshake packet is larger than the path's MTU, then the QUIC handshake fails, and the client uses the fallback TLS/TCP connection.
Anyway, QUIC is designed for HTTP/2 application protocol, thus also machine states on sending or receiving parts of streams can be easily mapped between HTTP/2 and QUIC.

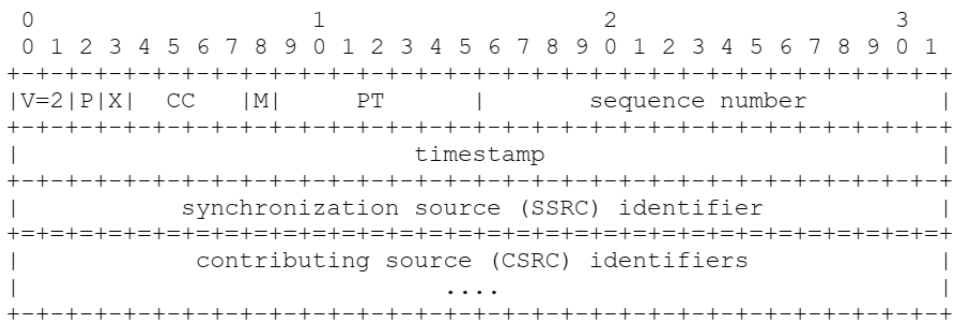
5.7 Real Time Protocol (RTP)

The Real Time Protocol (RTP) has been developed to transport real-time data such as audio, video or data, over multicast or unicast transport services. It is described in RFC 3550 in 2003 [30]. Similarly, to QUIC also RTP could be classified as a hybrid application/transport protocol. RTP normally runs in user space over UDP (in the operating system) but it can use other transport protocols such as including TCP, UDP-Lite, DCCP, TLS, and DTLS.

5.7.1 Protocol description

The main functionalities of RTP are: *i.* to transport audio and video data in packets, by multiplex several real-time data streams onto a single stream of UDP packets; *ii.* to support the receiver to provide audio and video data in the right time.
RTP provides fragmentation as well as small packets from application can be bundled together into a single RTP packet. It does not offer any special guarantee about delivery, thus packets can be lost, corrupted and delayed. Partial protection can be exploit by lower transport protocols (e.g., DCCP). RTP has no acknowledgements, and no mechanism to request retransmissions, since it is not practical for real-time applications.
An application provides audio/video data (multiple streams) to the RTP sender entity, which encapsulates these data in an RTP packet, delivered to the UDP. RTP Control Protocol (RTCP) supports RTP in monitoring the transmission statistics of the network, thus helping it in synchronizing the multiple streams. Examples are the highest sequence number received, and the inter-arrival jitter, RTT and the number of packets lost. Sender and receiver send periodically RCTP packets to report the network performance. Based on received RTCP feedback, an RTP sender can adjust the transmission by properly adapting the application rate. The RTP does not directly provide connection setup, instead it relies on other protocols to setup, negotiate parameters, or tear down a session such as SIP (see the corresponding paragraph in appendix for its description).
RTP has been designed to carry a multitude of multimedia formats. To this aim, RTP defines a profile and associated payload formats for each group of applications. The profile defines the codecs used to encode the payload data and their mapping to payload format codes (this is included in the Payload Type field of the RTP header Figure 27). For example, a single audio stream may be encoded as 8-bit PCM samples at 8 kHz using delta encoding, predictive encoding, GSM encoding, MP3 encoding. Examples of audio payload formats are G.711, G.723, G.726, G.729, GSM, QCELP, MP3, and DTMF, and examples of video payloads are H.261, H.263, H.264, H.265 and MPEG-1/MPEG-2. In Figure 27 it is reported the format of the RTP header.

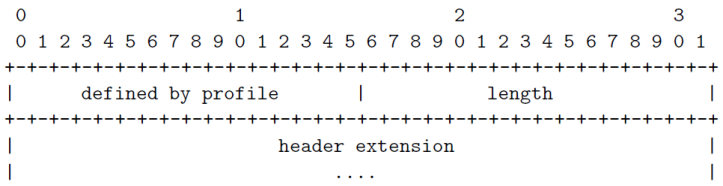
Figure 27: RTP header format.



It consists of three 32-bit words and potentially some extensions. In the following it is reported the meaning of each field:

- Version (2 bits): Indicates the version of the protocol. Current version is 2;
- P (Padding) (1 bit): it indicates if there are extra padding bytes at the end of the RTP packet;
- X (Extension) (1 bit): it indicates the presence of an extension header between the header and payload data. The extension header is application or profile specific. This allows to extend the protocol features; extensions are not defined except the first word (see Figure 28);

Figure 28: extension header in RTP.

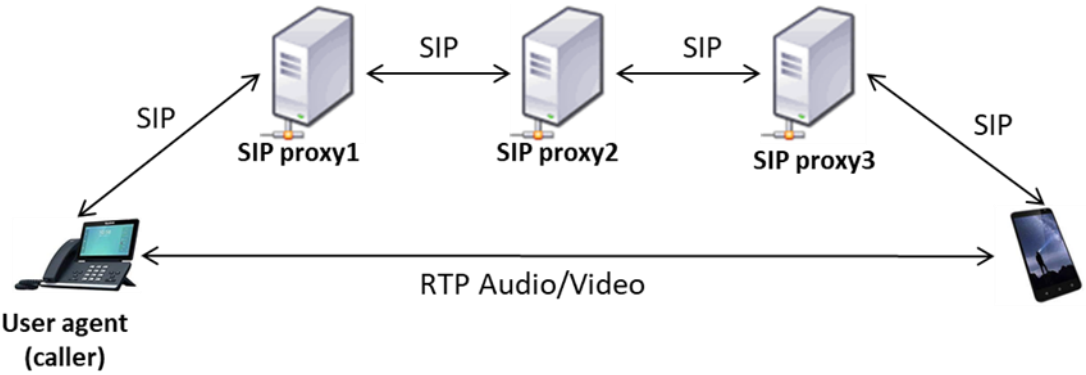


- CC (CSRC count) (4 bits): it contains the number of Contributing Source (CSRC) identifiers (defined below) that follow the Synchronization Source identifier (SSRC) (also defined below), from 0 to 15;
- M (Marker) (1 bit): it indicates the signaling used at the application level in a profile-specific manner. If it is set, it means that the current data has some special relevance for the application (e.g., the start of a video frame or audio frame or something else that the application understands;
- PT (Payload type) (7 bits): it indicates the encoding algorithm used for the payload, thus supporting its interpretation by the application. Since it may change on a packet basis, the encoding may be dynamically assigned;
- Sequence number (16 bits): it is incremented for each RTP data packet sent and is used by the receiver to detect packet loss and to accommodate out-of-order delivery. Note that RTP just signals this information to the application. Then, it is up to the application take care of lost packets for example by the application may skip a video frame (if the packets are carrying video data), or to approximate the missing value by interpolation (if the packets are carrying audio data);
- Timestamp (32 bits): it used by the receiver to play back the received samples at appropriate time and interval. When several media streams are present, the timestamps may be independent in each stream;
- Synchronization source (SSRC) identifier (32 bits): it uniquely identifies the source of a stream, whose packet belongs to. This method is necessary to separate the multiplexed streams into one single UDP stream;
- Contributing source (CSRC) identifier (32 bits each): it is the number of entries is indicated by the CSRC count field. It enumerates contributing sources to a stream which has been generated from multiple sources.

5.7.2 Relationship between SIP and RTP

As described in the appendix, the SIP protocol is used to initiate a session between two endpoints. For this reason, it is usually used by the RTP to allows the two endpoints to set up a connection for transferring voice or video data. Similar to an SMTP server, the SIP protocol exchange information on receiver location through one or more SIP proxy servers before reaching its destination. Moreover, SIP uses a format similar to HTTP with the same error codes. For example, both HTTP and SIP use 408 as the error code to signal a timeout error, 404 for 'not found', etc.
SIP listens on port 5060 (usually UDP, but can be TCP), while RTP uses a dynamic port range (and is only ever UDP), generally between 10000-20000. This range can usually be customized on the client to suit differing firewall configurations. Now, while SIP traffic passes from one server to the next to get to its destination, RTP sessions are set up directly between SIP clients. The RTP session setup through SIP is detailed in Figure 29.

Figure 29: RTP session setup through SIP

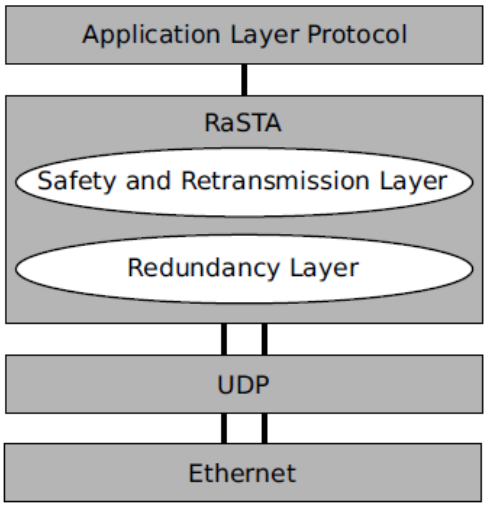


The scheme in figure can be explained simply with an email example. One user wants to call Bob on the phone, but Bob's number is unknown. However, the user owns the Bob's email address and send to him an email to telling him to call me on the user's phone number. The email passes through several email servers and reaches Bob's inbox. When Bob reads the email with the user's phone number, picks up the phone, and calls the user directly.

5.8 Rail Safe Transport Application (RaSTA)

Rail Safe Transport Application (RaSTA) is a transport protocol, which has been designed for transmitting control messages between railway interlocking systems in the safety-critical domain of railway signaling. Its main security properties are message integrity, reliable in-time and in-sequence delivery, and high availability [31].
RaSTA has been described in DIN VDE V 0831-200 [32]. In Figure 30 it is reported the scheme of the RaSTA protocol stack. RaSTA is used by the railway application and it exploits the UDP to send messages in the communication network.

Figure 30: Scheme of the RaSTA protocol stack



RaSTA protocol is composed of two layers: the safety and retransmission layer and the redundancy layer. Functionalities of redundancy layer are:

- correct message sequence;
- providing timeliness;
- providing availability.

Note that the communication availability is obtained by duplicating messages, thus sending identical messages over more than one distinct UDP channels. Duplicates are managed within this layer, thus only a single copy of each messages is presented at the above safety and retransmission layer.
The second layer in RaSTA is the safety and retransmission layer, which is responsible of:

- request corrupted or lost messages;
- integrity;
- message authenticity.

The integrity of a message in this layer (header and payload) is obtained by calculating (and inserting it in the sent message) an 8 byte-checksum, through the MD4 hash function (truncated at 8 B). RaSTA allows to alter the MD4 initialization vector (IV) from the default value defined in RFC 1320.

Considerations on the status of the RaSTA protocol
Even if RaSTA has been designed to allow security communications between railway interlocking systems as well as signaling and switching equipment, it presents some weaknesses. Some security attack can be performed to reduce its effectiveness. Examples are related to replay attacks and to secrecy of the key used for hashing.
The RaSTA protocol is standardized by German Institute for Standardization (DIN) and it remains a protocol whose specifications are contained in non-public documents. Furthermore, only a pre-version exists (in German at this link https://www.vde-verlag.de/normen/0800234/din-vde-v-0831-200-vde-v-0831-200-2015-06.html and there is also a non-free English version). It is a technical recommendation which is not mandatory. Due to the private nature of RASTA protocol specification, we have avoided to insert more details on its operations (e.g., the packet organization, the fields of the packets, and so on) and in this review we have reported and elaborated on what is publicly available in [31]. In addition, as expected, no public software implementation of RaSTA protocol is available and this inevitably will limit, or may render very difficult, the investigation activities to be carried out in the next Tasks on the RaSTA performance.

5.9 Summary and comparison of Transport Protocol functionalities

In this section, we report a possible comparison of the surveyed existing transport protocols. Comparison results are in Table 7.

Table 7: comparison among transport layer protocols

	TCP/MPTCP	UDP	SCTP	DCCP	QUIC	RTP
Congestion control	Yes Window-based	No	Yes	Yes	Yes	Defined in each RTP profile (reducing the encoding)
Flow control	Yes, advertising window	No	Yes	Yes	Yes	No

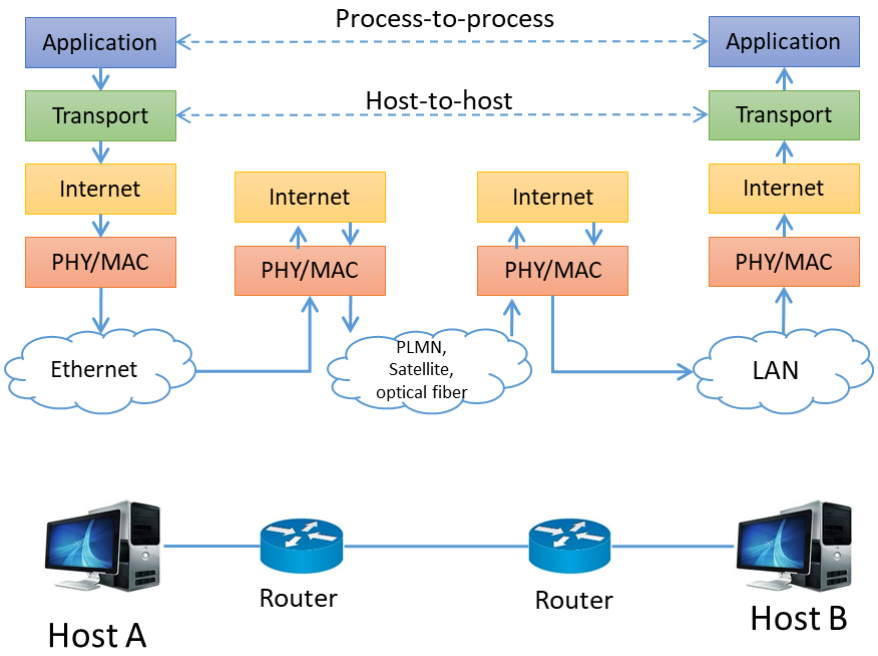
Error control	Yes	No	Yes	No	Yes	Just detection
Connection establishment	Yes	No	Yes	Yes	Yes (1-RTT, 0-RTT)	No (Yes, if supported by other protocols, e.g., SIP)
Addressing	Yes	Yes, optional	Yes	Yes	Yes	Synchr. Source ID
Checksum (for misdelivery)	Yes, 16-byte	Yes, 16-byte	Yes, 32-byte			No
Multiplexing	Yes	Yes	Yes	Yes	Yes	Yes
Connection-oriented	Yes	No	Yes	Yes	Yes	No
Casting	Uni	Uni, any, multi, broad	Uni	Uni	Uni	Uni, multi (using IP)
Stream/message-oriented	Byte	Message	Message	Message		Message
Reliable	Yes	No	Partial	No	Yes	No
Data bundling	Optional (Nagle)	No	Yes	No		Yes

6 Application protocols

6.1 General concepts

Communications between two computers (Host A and Host B) occur according to the OSI model or the Internet model (TCP/IP). Its model is reported in Figure 31. Data are sent in the connecting networks and pass through several routers. Higher layers are implemented only at the external elements, while lower layers are also in the intermediate elements in the networks (e.g., routers). The application layer protocols are implemented on the source host and the destination host and provide rules for communication between applications, thus allowing application process on host A and application process on host B to communicate.
Defined rules in the application layer protocols include message exchange (i.e., way to send the message and the expected response), message format and meaning of each field.

Figure 31: Principle scheme of data flow between two applications in a network.



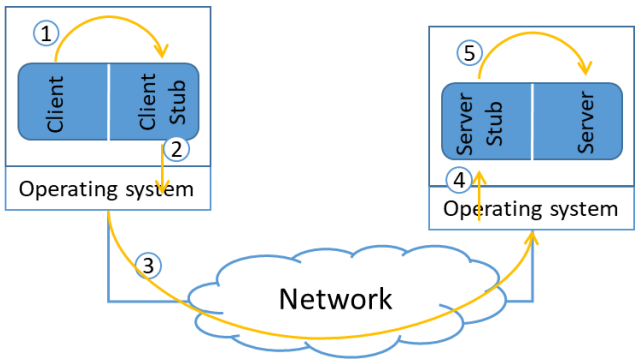
The application layer is important to support and overcome the application to the problems related to data transmission on a real network (or more sub-networks). Then, it abstracts and hides the details of lower layers (subsystems and networks), by providing interface methods used by hosts in a communications network.
A lot of application protocols have been designed for the Internet [33]. The most of them are quite similar from a functional behavior point of view, even though the actual details vary considerably. As a general rule, there are at least two ways to proceed for developing an application protocol:

1. Find an existing exchange protocol that (more or less) does what you want;
2. Define a new protocol from scratch that does exactly what you want.

Depending on the specific application, the application protocols can exhibit many features. Some important features are:

- Is it connection-oriented or connectionless? Connection-oriented application protocols usually work on top of TCP. Connectionless protocol consists of those that do not want the delay or overhead of establishing and maintaining a reliable stream.
- Does it use requests and responses to exchange messages? The "message" is simply structured data exchanged (usually by inter-process communication or a network) between loosely-coupled systems. In tightly-coupled systems, the remote procedure call (RPC) is used as the exchange paradigm (see Figure 32 how RPC works).
- Allows for asynchronous message exchange.

Figure 32: Working of the Remote Procedure Call [8].



Concerning on the message exchanged asynchronously we can consider the typical Internet scenario, where a client sends a request to a server, which processes the request and then sends back a response. These request/response protocols can be implemented over either connectionless or connection-oriented protocols. Communication between a sender and one or more receivers can be synchronous. In this case, the caller is blocked until one or all responses are returned. In the asynchronous case, the caller returns immediately after sending the message and has to retrieve the response(s) later and correlate it/them to the request sent (usually using some form of message numbering). The choice between synchronous or asynchronous message exchange is affected by two factors (RFC 3117 [33]):

- The interrelatedness of requests and how to retrieve them
- The latency of the underlying protocol or communication media.

For what concerns the tasks that an application protocol must perform and how it goes about performing them, the most important mechanisms are:

- Framing, which tells how the beginning and ending of each message is delimited;
- Encoding, which tells how a message is represented when exchanged;
- Reporting, which tells how errors are described;
- Asynchrony, which tells how independent exchanges are handled;
- Authentication, which tells how the peers at each end of the connection are identified and verified;
- Privacy, which tells how the exchanges are protected against third-party interception or modification;
- Naming.

To conclude this introduction on application protocols, the designer as well as who has to select it should consider the capacity of the application protocol to be scalable, efficient, simple, extensible and robust against possible situations that can occur in network.
HTTP is the most famous and important application layer protocol used for web browsing and web applications and it is detailed in the next Section.

6.2 Hypertext Transfer Protocol (HTTP)

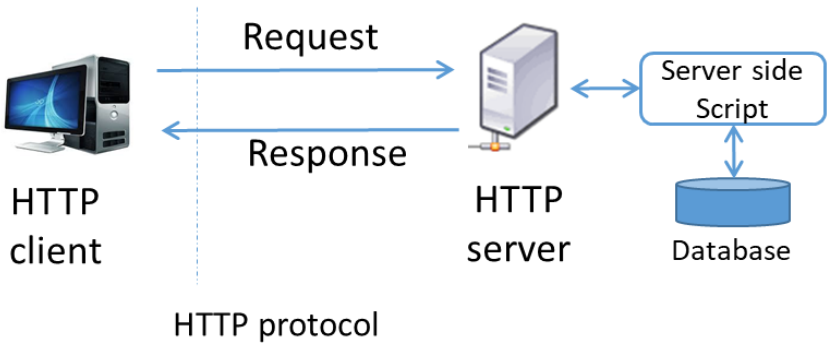
The Hypertext Transfer Protocol (HTTP) is probably the most popular application protocol, since it has been in use by the World-Wide Web (www) since in the 1990. Following the definition given by developers [34],

> "The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems."

6.2.1 Overall operations

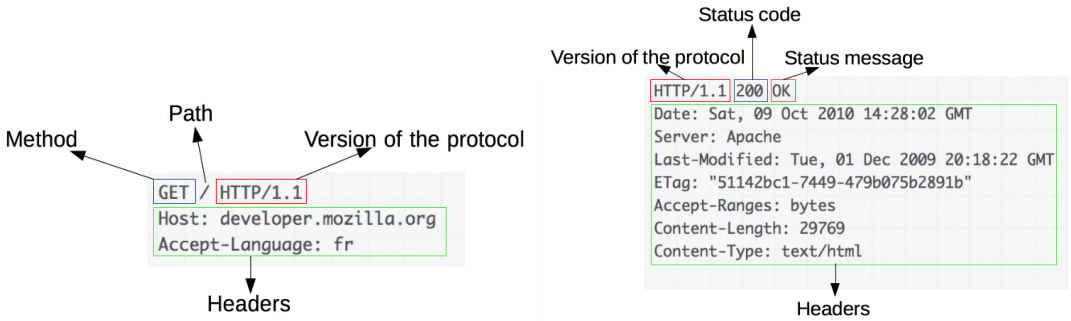
The HTTP architecture reported in Figure 33 is very simple. A HTTP client (e.g., a web browser, a bot or a search engine) sends a request to the HTTP server containing: a method (called Uniform Resource Identifier (URI)), the protocol version and other extra information (e.g., message Multipurpose Internet Mail Extensions, MIME [35]) over a TCP/IP connection. The HTTP server sends back a response containing a status line, the success or error code, server and meta-data and possible entity-body content.

Figure 33: Model of the HTTP architecture



Examples of messages of HTTP protocols are reported in Figure 34.

Figure 34: Examples of exchanged HTTP messages: (a) request; (b) response.



The **HTTP Request** contains three elements:

1. An HTTP method describing the action to be performed. For example, GET indicates that a resource should be fetched or POST means that data is pushed to the server (creating or modifying a resource, or generating a temporary document to send back).
2. The request target, usually a URL, or the absolute path of the protocol, port, and domain are usually characterized by the request context.
3. The HTTP version, which defines the structure of the remaining message. It gives information on the expected response.

The start line of an **HTTP response** contains the following information:

1. The protocol version, usually HTTP/1.1.
2. A status code, indicating success or failure of the request. Common status codes are 200, 404, or 302.
3. A status text, which briefly describes the status code to help a human understand the HTTP message.

In Table 8 the main classes of status code in the response are reported.

Table 8: Main classes of status code in the response in HTTP/1.1 [34].

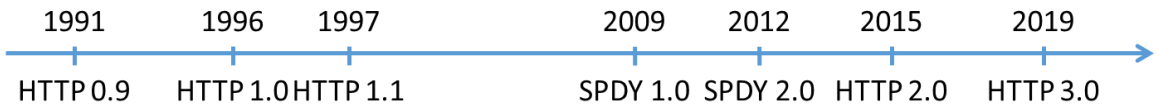
Code	Type	Description
1xx	Informational	Request received, continuing process
2xx	Success	The action was successfully received, understood, and accepted
3xx	Redirection	Further action must be taken in order to complete the request
4xx	Client Error	The request contains bad syntax or cannot be fulfilled
5xx	Server Error	The server failed to fulfill an apparently valid request

The basic features of HTTP are:

- Connectionless: each request corresponds a response. Further requests are treated as independent from each other. Although HTTP does not require a connection to be established, it requires that data transmission be reliable. For this reason, TCP is used as a transport protocol. Note that HTTP/1.0 uses a new connection for each request/response exchange, where as HTTP/1.1 connection may be used for one or more request/response exchanges;
- Stateless: The server and client are aware of each other only during a current request; thus, the browser can retain information between different requests across the web pages;
- Media independent: data are considered valid as long as client and server can handle them, also by specifying the content type using appropriate MIME-type.

The timeline of the http protocol is depicted in Figure 35.

Figure 35: Timeline of HTTP



The HTTP/0.9 was a simple protocol for raw data transfer across the Internet. Even with limitations (unable to manage proxies, caching and persistent connections), in 1996 HTTP/1.0 was defined in RFC 1945 [36], by allowing messages to be in the format of MIME-like messages. This version needed to evolve into HTTP/1.1 [34], since the proliferation of incompletely-implemented versions in the web. HTTP is at the base of the access to hypermedia resources available from diverse applications, through methods and headers that indicate the purpose of a request giving the reference of the object to be retrieved according to the URI. Then, in addition to www application, HTTP can be used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems, including those supported by the Simple Mail Transfer Protocol (SMTP), Network News Transfer Protocol (NNTP), File Transfer Protocol (FTP) and Gopher. Then, HTTP allows basic hypermedia access to resources available from diverse applications.
In 2015, HTTP/2.0 version has been defined [37] with the aim of managing network resources more efficient and reducing the latency perception. It implements the header compression and allows multiple concurrent exchanges on the same connection. Anyway, HTTP/2.0 does not obsolete HTTP/1.1 message syntax.
In 2019 HTTP/3.0 [38] has been drafted and its standardization activities are ongoing. Differently from HTTP/1.1 and HTTP/2.0, HTTP/3.0 adopts QUIC as transport protocol. When two endpoints agree on use HTTP/3.0, QUIC provides protocol negotiation, stream-based multiplexing, and flow control, while the internal framing layer is similar to HTTP/2.

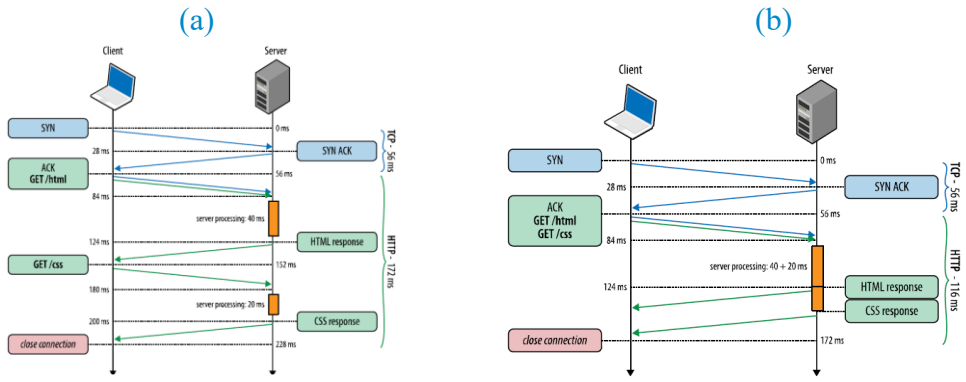
6.2.2 Improvements in HTTP 1.1

In HTTP/1.1, some changes have been introduced in the standard in order to resolve protocol ambiguities in earlier versions and to optimize the performance. Some enhancements are (https://hpbn.co/http1x/):

- The keepalive connection, to reuse previous connection;
- Data pipeline to allow parallel request processing;
- Chunked transfer encoding to allow response streaming
- Byte serving to allow range-based resource requests
- Improved and much better-specified caching mechanisms

Keepalive Connection
In previous versions different requests need for the establishment of different TCP connections. Each TCP connection begins with a TCP three-way handshake, which takes a full roundtrip of latency between the client and the server. In the case the client performs two requests (see the example in Figure 36a) HTML and CSS, they need two separate TCP connection establishments doubling the three-way procedures (and another round of TCP slow-start) and the latency. With HTTP/1.1 the requests containing HTML and CSS can be performed after the response of the previous one, by reusing the underlying connection.

Figure 36: Enhancements in HTTP/1.1: (a) keepalive connection; (b) pipeline [53].



Using Multiple TCP Connections

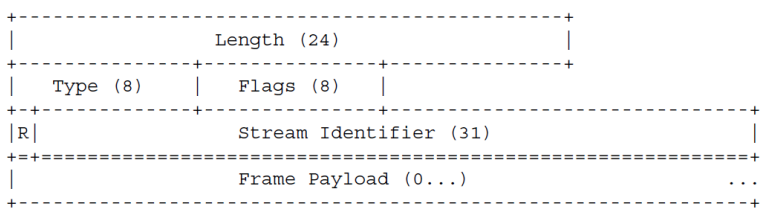
If request multiplexing is not available, the browser can send them by queuing in a single kept alive connection. Unfortunately, to speed up the webpage download, the browser opens up to six parallel TCP connections, in practice. This allows the client and the server to process six requests and response in parallel and the TCP congestion window is larger of a multiplied by the number of open connections. Nevertheless, more resources such as sockets, CPU and buffer are consumed in the client, server and intermediate network elements. Moreover, competition for shared bandwidth between parallel TCP streams can occur.

6.2.3 Improvements in HTTP/2

In order to reduce "perceived latency" (defined as the time the user sees between making a page request and it being rendered), some improvements in HTTP/2.0 have been implemented. To overcome the limitations of HTTP/1.0 (requests made sequentially) and HTTP/1.1 (pipeline and opening of multiple TCP connections), two new concepts are introduced in how client and server communicate over the wire: frames and streams.

Frames in HTTP/2.0 replace the familiar header & body format of current HTTP requests/responses. The full list and details of their usage are in the HTTP/2 RFC 7540 [37]. Data and header frames are separated, thus allowing the header compression as well as the body. Figure 37 reports the format of the HTTP/2.0 frame.

Figure 37: Frame format in HTTP/2.0 [37].

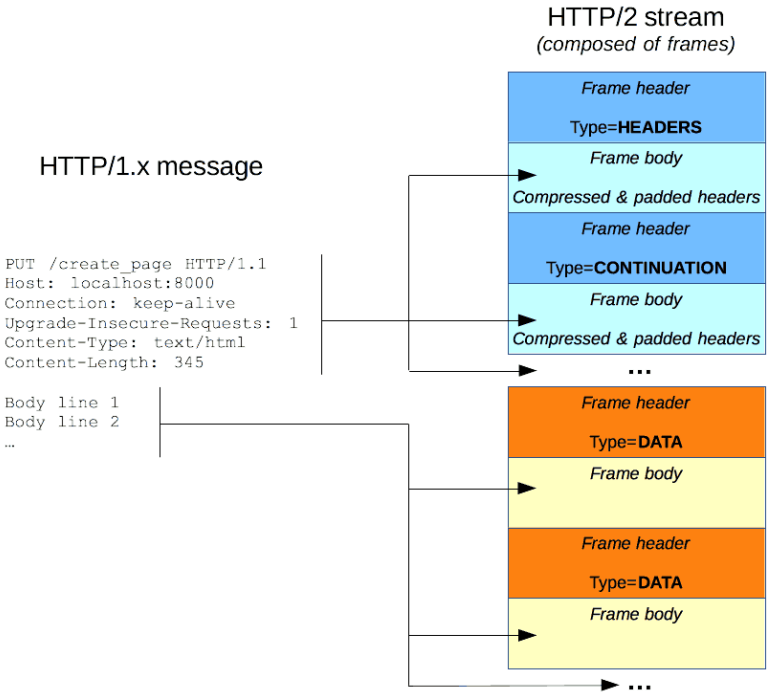


The header fields are:

- Length (24 bits): The length of the frame payload;
- Type (8 bits): The 8-bit type of the frame;
- Flags (8 bits): Field reserved for Boolean flags specific to the frame type;
- R (1 bit): set to 0, (reserved);
- Stream Identifier (31 bits): the identifier of a stream.

In Figure 38, taken from MDN web Docs (by Mozilla) [53], the relation between the HTTP/1.1 message and the frame in HTTP/2.0 is reported.

Figure 38: Relation between the HTTP/1.1 message and the frame in HTTP/2.0



As defined in RFC 7540 [37], a **stream** is just an independent, bidirectional sequence of frames exchanged between the client and server within an HTTP/2 connection, that share a common identifier (the Stream Identifier). A single HTTP/2 connection can contain multiple concurrently open streams, with either endpoint interleaving frames from multiple streams (i.e., the stream multiplexing is implemented). This solves the “head-of-line” blocking, thus requests made on the same connection can now be responded to out of order.

6.2.4 HTTP/2 vs HTTP/3

The main difference between HTTP/2.0 and HTTP/3.0 is the transport protocol used to communicate: TCP in HTTP/2.0, while QUIC for HTTP/3.0. The following table summarizes the main differences between HTTP/2 and HTTP/3:

Table 9: Comparison between HTTP/2 and HTTP/3

Feature	HTTP/2.0	HTTP/3.0
Header compression algorithm	HPACK	QPACK
Handshake protocol	TCP + TLS	iQUIC

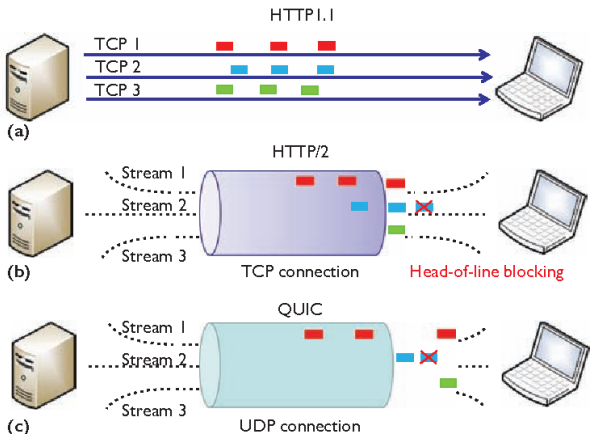
Handshake negotiation	At the certificate stage via ALPN=Application-Layer Protocol Negotiation (ALPN) protocol	After certificate negotiation via "Alt-Svc:" HTTP response header
HTTP scheme	HTTP (not well adopted) / HTTPS	HTTPS
Prioritization	Yes	No, although HTTP/3.0 streams can have a "PRIORITY" frame to implement it

Both versions support the header compression, multiplexing over a single connection using streams and offer the possibility to push the server. Main differences are due to the transport protocol capabilities:

- QUIC may speed up the transmission through 0-RTT handshakes, while TCP Fast Open and TLS usually send less data and have less ability to handle traffic spikes;
- HTTP/2.0 may implement unsecure case (TCP without TLS, thus just HTTP and not HTTPS), while HTTP/3.0 uses QUIC, which implements encryption;
- HTTP/3.0 has no prioritization, while HTTP/2.0 may implement it even it is quite complicated;
- HTTP/2 can be negotiated directly during the TLS handshake with the ALPN extension, while HTTP/3 is based on QUIC, so it needs an Alt-Svc header response.

Finally, in Figure 39 it is reported how HTTP versions manage multiple connections. HTTP/1.1 uses multiple TCP connections. HTTP/2.0 single connection and several streams. This can represent a bottleneck for data in a low network quality environment. In fact, when the network quality degrades, and packets are dropped, the single connection slows the entire process down, and no additional data can be transferred during this time of retransmission. HTTP/3.0 solves this issue through QUIC. In fact, QUIC is not blocking stream in case of packet loss. All transmission will be continuing and lost packet will attempt independently for retransmission without any stream blocking.

Figure 39: Managing multiple connections in HTTP/1.1, HTTP/2.0 and HTTP/3.0.



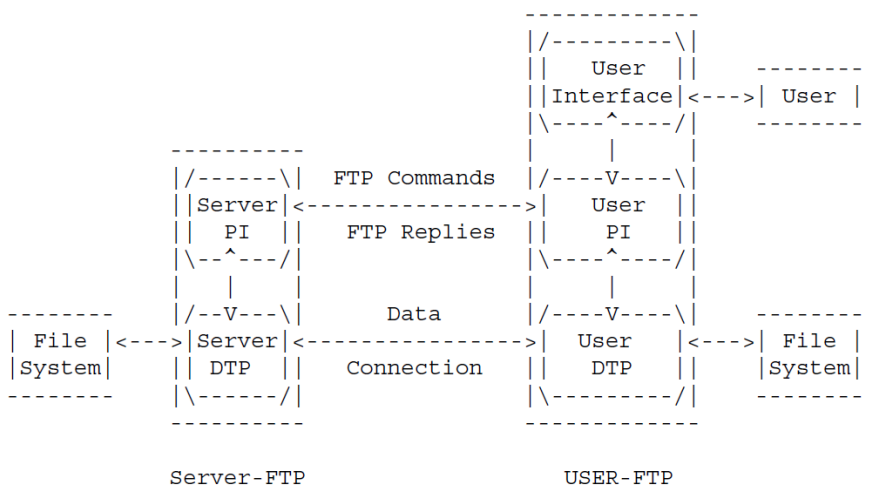
6.3 File Transfer Protocol (FTP)

The File Transfer Protocol (FTP) is a network protocol for exchanging files from a server to a client, according to a client-server model. The connections for control and data are different. From the authentication point of view, FTP users can access to the server with username and password in clear text or secured through encryption scheme based on Transport Layer Security (TLS) with Secure Sockets Layer (SSL). Abhay Bhushan prepared the first technical specification on FTP. The work was published as RFC 114 on 16 April 1971. FTP was supported by Network Control Protocol (NCP) until 1980. When NCP was replaced by TCP/IP version RFC 765 (June 1980) and RFC 959 (October 1985) are the reference technical specification on FTP with TCP/IP [39]. Successive RFC standards introduced some amendments to RFC 959, as:

- RFC 1579 (February 1994): Firewall-Friendly FTP in passive mode is added (passive mode);
- RFC 2228 (June 1997): add security features are added;
- RFC 2428 (September 1998): support for IPv6 is added (passive mode).

The FTP model is reported in Figure 40, the FTP protocol at user level initiates the control connection following Telnet. Then, commands are transmitted to the serve process via the control connection. Replies are sent from server-PI (protocol interpreter) to the user-PI over the control connection in response to the commands. The FTP commands specify the parameters for the data connection (i.e., data port, transfer mode, representation type, and structure) and for the file system operation (i.e., store, retrieve, append, delete, etc.). The user-DTP (data transfer process) should "listen" on the specified data port. Then, the server initiates the data connection and data transfer in accordance with the specified parameters.

Figure 40: FTP model

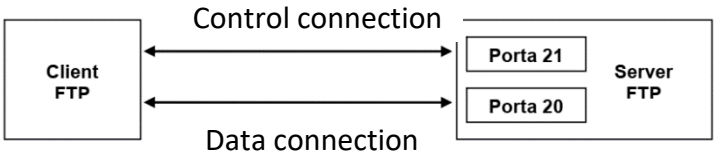


FTP is characterized by a stateful control connection. The server can track all client's activities and two different TCP connections are used: one for control and the other one for data. The establishment of the FTP control connection is slow since all required information and commands must be exchanged between the client and the server. For this reason, a control connection is active for multiple file transfers. If the data transfer is too slow, the firewall or NAT can decide to stop the control connection and then to interrupt the client tracking.

6.3.1 Protocol operations

Unlike other protocols such as HTTP/3, FTP uses two parallel TCP connections for file transfer. One is used for control connection (e.g., to exchange information about the identification or password), the other one for data connection (file data). For this reason, FTP uses the control information "out of band". The Figure 41 shows the control and data connection for FTP.

Figure 41: Control and data connections in FTP.



The FTP server allows to client the possibility to:

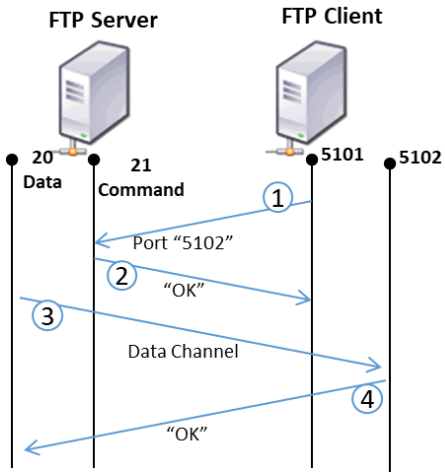
- Download / upload the file
- Recovery the interrupted file transfers

The FTP session is initiated by the client to the server as follow:

- The client uses its ports with a random number greater than 1023 both for connection and data connections
- The client establishes the control connection to the server port n. 21 sending the username and password and the commands necessary to change the remote directory
- The server initiates the data connection to the client using the port n. 20. For a second file transfer, the same control connection is still valid, but a second FTP data connection is established

The FTP message sequence is shown in Figure 42.

Figure 42: FTP setup and data transfer



During a session, the FTP server must know the user account and the relative state. Due to the necessity to track the client information by the server, the maximum number of FTP sessions to be open in parallel is limited.
FTP protocol can be used in active or passive mode and it defines the type of data connection [39].

- Active mode: the client sends the FTP command PORT <number> to communicate to the server its listening port number to server. The server opens a data communication with the client on its port number 20.
- Passive mode: The client uses the control connection to the server for receiving the information about the server (server IP address and server port number). These information are used for the data connection.

The passive mode is a more secure modality than the active mode, since the client does not send information about its port number. This approach can prevent malicious attacks, although the FTP protocol does not implement any security feature, as encryption. In this way information as username, password, and commands, can be intercepted and sniffed by attackers. This lack of security also affects other protocols developed before SSL 9, as HTTP. A solution is represented by the FTPS 10 protocol, introduced with the RFC-4217.

6.3.2 NAT and firewall traversal

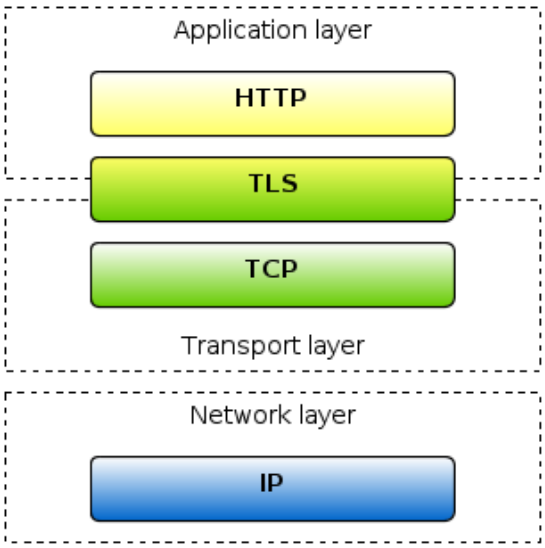
If a NAT and firewall are present, FTP can suffer from the unavailability to know the port numbers sent by the client [40]. Moreover, from a NAT perspective, the IP address and port number put in the PORT command are related to the host and not to the NAT.
The possible solutions are two essentially:

1. The FTP passive mode can be used;
2. NAT can modify the PORT command information through an application-level gateway.

6.4 Hypertext Transfer Protocol Secure (HTTPS)

Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP) in order to secure communication over a computer network, defined in RFC 2818 [43]. In HTTPS, HTTP data are transmitted on a TLS connection (formerly SSL). Then, HTTPS is usually referred as HTTP over TLS as also reported in Figure 43.

Figure 43: Protocol stack for HTTP and HTTPS.

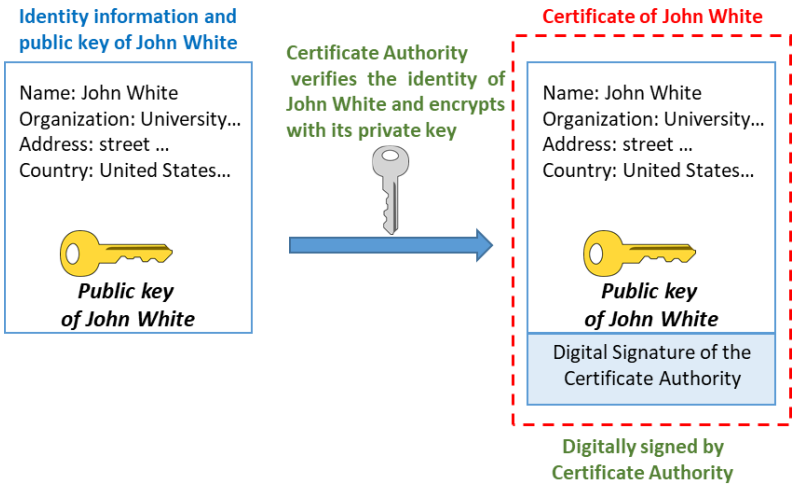


Functionalities implemented in HTTPS are:

- Encryption: the protocol provides confidentiality in communication between the user browser and the server as well as the activity tracking across multiple pages;
- Data integrity: data cannot be modified or corrupted during transfer, intentionally or otherwise, without being detected.
- Authentication: the protocol is able to certificate that your users communicate with the intended website, avoiding man-in-the-middle attacks favoring applications such as home banking.

In order to guarantee the authentication of the server, a trusted third-party or a certification authority (CA) requires the server to register delivering a digital certificate. A digital certificate certifies the ownership of a public key by the named subject of the certificate generally adopting X.509 standard. Then, the client web browser can use the public key of the server he is intended to communicate for ciphering transmitted data as attested by the CA signature.
In Figure 44 it is reported how the digital certificate works and what information contains.
Through the certificate the user is sure that the public key of that server is valid, and it is who he claims to be.

Figure 44: Example of how the digital certificate works.



A public key certificate contains the following fields[7]:

- Issuer: it indicates the CA that issued the certificate. If the CA is trustable and if the certificate is valid, then the user can trust the certificate;
- Period of validity: it indicates the expiration date of the certificate;
- Subject: includes information about the entity that the certificate represents.
- Subject's public key: it is the public key of the subject;
- Signature: the CA signs digitally the digital certificate guaranteeing its validity and information contained.

[7] Refer to https://www.ibm.com/support/knowledgecenter/SSYKE2_7.0.0/com.ibm.java.security.component.%2070.doc/security-component/jsse2Docs/publickeycertificates.html.

Differently from HTTP, which uses port 80, HTTPS is served by port 443. According to RFC 2818 [43], the HTTP client should act as TLS client. Then, it should start the connection on the correct port within the TLS handshake. After the conclusion of the TLS handshake, the HTTP client may send the HTTP request, inserting it in the TLS payload frame. The URI scheme of HTTPS has identical syntax to the HTTP scheme, except the 'https' protocol identifier in place of the 'http' protocol identifier. Note that HTTPS is not slower than HTTP since latest versions allow multiple requests simultaneously.

6.5 File Transfer Protocol Secure (FTPS)

The File Transfer Protocol Secure (FTPS), also known as FTP-SSL or FTP Secure, is an important enhancement of FTP in terms of security. It substitutes the FTP protocol with TLS and SSL (now deprecated by RFC7568), and cryptographic protocols. FTPS is different from the SSH File Transfer Protocol (SFTP), which is a secure file transfer method for SSH protocol based on tunnels. FTPS provides more security features to the basic version of FTP protocol.
The methods for invoke the security of client can be implicit or explicit.

- Implicit FTPS: the data are encrypted via FTP using a different port. An SSL connection is established using the TCP port n. 990 before proceeding. The server can interrupt the connection if needed, although a TLS is established from the beginning of the connection. Negotiation is not supported with implicit FTPS configurations.
- Explicit FTPS (also FTPES): a first traditional FTP connection is established on the same standard port as FTP, after which a secure SSL connection is initiated on TCP port 21. The explicit method is defined in RFC 4217 [41]. In the later versions of the document, FTPS compliance required that clients always negotiate using the AUTH TLS method.

Explicit FTPS is today supported by many servers due to the capability to protect the data: before the data transfer, the client can request details about the encryption so to know which data are protected. If security features are not implemented or the connection is not accepted, the data transfer is performed using traditional FTP. Implicit FTPS is deprecated and not used yet, although some providers continue to implement it. The RFC 2228 [42] introduces the negotiating of authentication and security for FTP, including the new FTP command AUTH to enable a response with error code 504 ("not supported") if the FTPS initiates the interaction using a not recognized security procedure. Clients can invoke security features by querying the FTPS server using FEAT, AUTH TLS and AUTH SSL commands.
FTPS provides support for cryptography schemes (as TLS and SSL), including:

- certificates for server-side public key authentication
- certificates for client-side authorization
- ciphers schemes, as AES, RC4, RC2, DES/Triple DES
- hash functions, as SHA, MD2, MD4 and MD5

Reasons to disable encryption

There are scenarios where encryption on data channel may be disabled, as follow:

- transferring files with a non-sensitive nature: the encryption is not necessary;
- transferring files already encrypted or together with VPN connection: the encryption is redundant;
- encryption protocols (as TLS or SSL) are not compliant with the target encryption level.

There are scenarios where encryption on control channel may be disadvantageous, as follows:

- FTPS is used when a client or server are connected to firewall or NAT

- AUTH and CCC/CDC commands are used in an iterative way by unknown FTP clients: it can be the prelude to a Denial Of Service (DOS) attack, generating multiple TLS/SSL each time

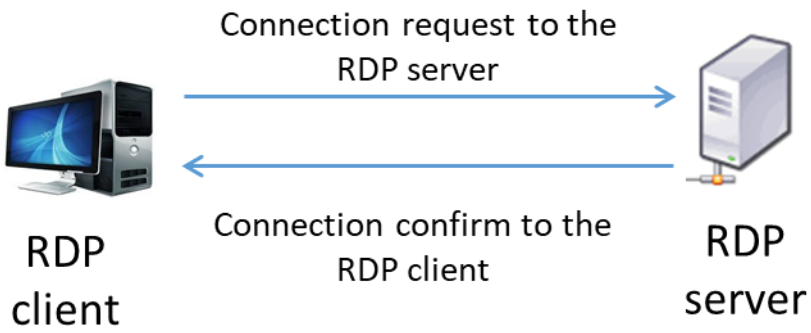
SSL certificates
FTPS servers must provide a certificate for public key, as HTTPS, using software tools (e.g., OpenSSL). A trusted Certificate Authority (CA) will sign these certificates, so to protect the client-server connection and to prevent a Man In The Middle (MITM) attack. Otherwise, the FTPS client may notify the used certificate is not valid and then it can decide to accept it or reject the connection. This does not happen with SFTP, which based on Out-of-band authentication of public keys and not on signed certificates.

Firewall incompatibilities
Since FTP uses two different TCP connections (control connection and data connection), many firewalls need to verify FTP control messages to identify which data connections will be admitted. When TLS/SSL protocols are used for encrypting the FTP control connection, the firewall cannot identify the TCP port number of a data connection exchanged between the client and FTP server. This can be avoided using a reduced port number for data and setting the firewall to open the allowed ports.

6.6 Remote Desktop Protocol (RDP)

The Remote Desktop Protocol (RDP) is a Microsoft proprietary protocol with the aim of using one device to control another by opening a graphical user session on a remote computer. Thus, RDP allows the local user (acting as a client) to view the graphic interface of the remote device (acting as a server) and the client can act directly on the remote PC using the local PC's I/O tools (see Figure 45). It has been developed for Windows but it is available also for MacOS, Linux e Unix. Differently from sending data in a cloud, RDP stores data securely on the user's desktop.

Figure 45: remote desktop components



6.6.1 Protocol description

RDP supports multichannel transmission (i.e., virtual channel separation) for carrying information such as presentation data, serial device communication, licensing information and highly encrypted data (e.g., keyboard and mouse activity). It opens a dedicated network channel for sending data back and forth between the local and remote machines, by always using network TCP port n. 3389.

Mouse movements, keystrokes, the desktop display, and all other necessary data are sent over this channel via TCP/IP. Nevertheless, RDP has been designed to work in different network topologies (as ISDN, POTS) and many LAN protocols (as IPX, NetBIOS). RDP also encrypts all data so that connections over the public Internet are more secure. Current activities transmitted in RDP use only one channel, but RDP provides 64,000 separate channels for data transmission.

As application protocol, RDP activities transmit data within the OSI protocol stack. Data from an application or service are fragmented, assigned to a channel, encrypted, inserted in a network packet, framed and then sent into a physical channel.
The protocol stack is managed by one RDP component, namely Multipoint Communication Service multiplexer (MCS mux). It has been standardized in the International Telecommunication Union (ITU) T.120 standard [44]. It is in charge of:

- Assigning channels by multiplexing data into predefined virtual channels within the protocol;
- Assigning priority levels;
- Fragmenting data that should be sent.

These functionalities are specified in T.122 standard regarding the multipoint services definition [45] and in T.125 regarding data transmission protocol specifications [46].
The Generic Conference Control (GCC) is a second RDP component in charge of the management of the multiple channels. The GCC belongs to T.120 standard family. It allows the creation and deletion of session connections and controls resources provided by MCS. To conclude the RDP, the Terminal Server device implements two further components:

- an RDP driver (Wdtshare.sys) for UI transfer, compression, encryption, framing, and so on.
- a transport driver (Tdtcp.sys) to package the protocol onto the underlying network protocol, TCP/IP. However, other driver of transport protocols can be added to make RDP independent of TCP/IP.

Further details on RDP are:

- the encryption adopting the RC4 algorithm with a 128-bit key. This does not require a secure VPN;
- the possibility to redirect the audio of the remote computer towards the local computer
- the support for multi-monitors
- the implementation of H.264 / AVC video compression algorithms for bandwidth optimization (in version 10).

The RDP performance depends on the quality of the connections between two computers. In addition of the low bandwidth, further slight delays often occur, since for example keyboard and mouse activity have to be encrypted and transmitted over the Internet from one side and the desktop display has to be transmitted back to the user from the other side.

7 Conclusions

In this deliverable we have summarized the most important features of IETF transport and applications protocols that should be used in the ACS in addition to classical TCP/UDP and HTTP protocols for unsecure/secure data exchange between hosts (on-board and on trackside) connected by ACS. The contents of this deliverable are the input for the next WP3 activities indicated in the following Tasks especially for Task 3.3. focusing on the setup of the emulator/simulator to be used for the testing of transport protocols (Task 3.4) and applications/transport protocols (Task 3.5) and their secure versions (Task 3.6) for the considered set of railway applications in the different railway scenarios indicated in Section 4. To this purpose the ACS functional model presented and discussed in Section **Errore. L'origine riferimento non è stata trovata.** will be the starting point for all the subsequent activities concerning the protocol performance analysis.
Taking into account for the specific characteristics of the ACS system and of the security aspects related to the application and transport protocols, in this Deliverable we have also reviewed two important protocols: the session initiation protocol (SIP) and the transport layer security (TLS). We have evidenced the main problems related to the usage of SIP in the presence of NAT devices in the network. These aspects will be further investigated in Task 3.2 concerning the IPv4/IPv6 interworking issues in ACS.

8 References

[1] SYSTRA: Final Report, "Implications of bearer independent communication concept", 2016 ERA 2016 17 RS25/07/2017 Ref. No. FR01T16H90_ERA_RS17_DLV_020, Available at: https://www.era.europa.eu/sites/default/files/library/docs/studies/systra_study_on_implications_of_the_bearer_independent_communication_concept_en.pdf

[2] K. Kernstock, C. Gruet, P. Beicht, M. Mikulandra, J. Arrieta, F. Parrilla Ayuso, J. Martinez Vivanco, F. Kaiser, L. Brühl, A. S. Chazel, G. Fontana, W. Zhao, J. Mattisson, B. Barth, "Specification of the Communication System and Guideline for Choice of Technology", Deliverable D3.3, X2Rail-1, 29-01-2019.

[3] B. Allen, B. Eschbach, P. Marsch, P. Hallner, J. Eriksson, G. Fontana, J. Mattisson, A. S. Chazel, P. Cotelle, P. Beicht, M. Mikulandra, T. Lane, "User & System Requirements (Telecommunications)", Deliverable D3.1, X2Rail-1, 5-12-2018.

[4] B. Allen, K. Keil, B. Holfeld, S. Guillemaut, L. Bruehl, M. Stoppa, G. Fontana, G. Ravera, "Adaptable Communications System Field Test Strategy", Deliverable D3.3, X2Rail-3, 29-09-2020.

[5] G. Fairhurst, B. Trammell, M. Kuehlewind, "Services Provided by Transport Protocols and Congestion Control Mechanisms", Internet Engineering Task Force (IETF), Request for Comments: 8095, Category: Informational, ISSN: 2070-1721, March 2017.

[6] J. Postel, "Transmission Control Protocol", STD 7, Request for Comment RFC 793, DOI 10.17487/RFC0793, Sep. 1981, https://tools.ietf.org/pdf/rfc793.

[7] J. Duke, M., Braden, R., Eddy, W., Blanton, E., and A. Zimmermann, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 7414, DOI 10.17487/RFC7414, February 2015, http://www.rfc-editor.org/info/rfc7414.

[8] A. S. Tanenbaum, D. J. Wetherall, "Computer networks", 5th ed., Pearson, 2011.

[9] Braden, R., Borman, D., and C. Partridge, "Computing the Internet checksum", RFC 1071, DOI 10.17487/RFC1071, September 1988, http://www.rfc-editor.org/info/rfc1071.

[10] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, http://www.rfc-editor.org/info/rfc7323.

[11] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, http://www.rfc-editor.org/info/rfc5681.

[12] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", RFC8312, February 2018, https://tools.ietf.org/html/rfc8312.

[13] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, DOI 10.17487/RFC2018, October 1996, http://www.rfc-editor.org/info/rfc2018.

[14] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, http://www.rfc-editor.org/info/rfc3168.

[15] J. Mo, R. J. La, V. Anantharam and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320), New York, NY, USA, 1999, pp. 1556-1563 vol.3.

[16] G. Huston, "BBR, the new kid on the TCP block", 9 May 2017, available at: https://blog.apnic.net/2017/05/09/bbr-new-kid-tcp-block/

[17] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, "TCP Extensions for Multipath Operation

with Multiple Addresses", RFC 6824, January 2013, http://www.rfc-editor.org/info/rfc6824.

[18] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, http://www.rfc-editor.org/info/rfc768.

[19] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, http://www.rfc-editor.org/info/rfc8085.

[20] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, http://www.rfc-editor.org/info/rfc4960.

[21] P. R. Egli, "Overview of SCTP (Stream Control Transmission Protocol)", 2013. Available at: https://de.slideshare.net/PeterREgli/overview-of-sctp-transport-protocol.

[22] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, http://www.rfc-editor.org/info/rfc5061.

[23] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, http://www.rfc-editor.org/info/rfc4895.

[24] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, DOI 10.17487/RFC6525, February 2012, http://www.rfc-editor.org/info/rfc6525.

[25] Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol", Work in Progress, draft-ietf-tsvwg-sctp-ndata-08, October 2016.

[26] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, http://www.rfc-editor.org/info/rfc4340.

[27] A. Langley et al. "The QUIC Transport Protocol: Design and Internet-Scale Deployment," SIGCOMM '17, August 21-25, 2017, Los Angeles, CA, USA.

[28] J. Iyengar, M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," draft-ietf-quic-transport-34, 15 January 2021, https://tools.ietf.org/html/draft-ietf-quic-transport-34.

[29] J. Iyengar, and I. Swett, "QUIC Loss Detection and Congestion Control", Work in Progress, Internet-Draft, draft-ietf-quic-recovery-34, 15 January 2021, https://tools.ietf.org/html/draft-ietf-quic-recovery-34.

[30] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003, https://tools.ietf.org/html/rfc3550.

[31] M. Heinrich, J. Vieten, T. Arul and S. Katzenbeisser, "Security Analysis of the RaSTA Safety Protocol", 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), Miami, FL, 2018, pp. 199-204, doi: 10.1109/ISI.2018.8587371.

[32] DKE, "Electric signalling systems for railways – Part 200: Safe transmission protocol according to DIN EN 50159 (DIN VDE V 0831-159)," DIN, Standard DIN VDE V 0831-200, Jun. 2015.

[33] M. Rose, "On the Design of Application Protocols," RFC 3117, November 2001, https://tools.ietf.org/html/rfc3117.

[34] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1,", RFC 2616, June 1999, https://tools.ietf.org/html/rfc2616#page-7.

[35] N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies,", RFC 2045, November 1996, https://tools.ietf.org/html/rfc2045.

[36] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC

1945, May 1996, https://tools.ietf.org/html/rfc1945.
[37] M. Belshe, R. Peon, M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, May 2015, https://tools.ietf.org/html/rfc7540.
[38] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-29, 9 June 2020, https://tools.ietf.org/html/draft-ietf-quic-http-29#page-5.
[39] J. Postel, J. Reynolds, "File Transfer Protocol (FTP)," RFC959, October 1985, https://tools.ietf.org/html/rfc959.
[40] "The File Transfer Protocol (FTP) and Your Firewall / Network Address Translation (NAT) Router / Load-Balancing Router", https://www.ncftp.com/ncftpd/doc/misc/ftp_and_firewalls .html
[41] P. Ford-Hutchinson, "Securing FTP with TLS", RFC 4217, October 2005, https://tools.ietf.org/html/rfc4217.
[42] M. Horowitz, S. Lunt, "FTP Security Extensions", RFC 2228, October 1997, https://tools.ietf.org/html/rfc2228.
[43] E. Rescorla, "HTTP Over TLS,", RFC 2818, May 2000, https://tools.ietf.org/html/rfc2818.
[44] ITU-T Recommendation, "Data protocols for multimedia conferencing", Series T: terminals for telematic services, T.120, Jan. 2007.
[45] ITU-T Recommendation, "Multipoint communication service – Service definition", Series T: terminals for telematic services, T.122, Feb. 1998.
[46] ITU-T Recommendation, "Multipoint communication service protocol specification", Series T: terminals for telematic services, T.125, Feb. 1998.
[47] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018, https://tools.ietf.org/html/rfc8446.
[48] Microsoft, "SSL/TLS in Detail", 10 Aug. 2009, https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc785811(v=ws.10)?redirectedfrom=MSDN
[49] H. Hooper, "CCNP Security VPN 642-684", official certification guide, Cisco Press, 2012
[50] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002, https://tools.ietf.org/html/rfc3261.
[51] Ingate Systems, "Solving the Firewall/NAT Traversal Issue of SIP: Who Should Control Your Security Infrastructure?", https://www.ingate.com/files/solving-firewall-NAT-traversal.pdf.
[52] 3GPP, "Study on Future Railway Mobile Communication System", Stage 1, (Release 15), 3GPP TR 22.889 V15.1.0 (2017-09) Technical Report, Sep. 2017
[53] MDN web Docs, "HTTP Messages" (https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages
[54] Ilya Grigorik, "HTTP/1.X", https://hpbn.co/http1x/

9 Appendix A

In the following two sections, we illustrate the main features of two important protocols for ACS operations: SIP and TLS. For the purpose of AB4Rail project it is not necessary entering into the deep details of the protocol operations i.e., the detailed description of the associated procedures and algorithms. It is not the goal of AB4Rail project to conduct studies concerning the possible introduction of (non-standard) modifications into the IETF existing protocols that are the basic constituents of the ACS system. In principle, it is difficult to classify these two protocols as application or transport protocols since they can be used to support all communications over IP-based networks. SIP is a signaling protocol that can be used to setup the connection parameters among hosts, while TLS is the main protocol used to secure communications using many application and transport protocols.

9.1 Transport Layer Security (TLS) protocol

Transport Layer Security (TLS) is a cryptographic protocol aiming to provide security to network communications. TLS was standardized in 1999 by IETF, while the current version is TLS 1.3 defined by IETF RFC 8446 in August 2018 [47]. The security layer in HTTPS is the most important application based on TLS, although it is also used for email, instant messengers, and VoIP. The TLS protocol is formed by two different layers: TLS record and TLS handshake protocols.
The TLS provides the following properties to a client-server connection:

1. The connection is secured thanks to a symmetric-key algorithm used data encryption
2. Each connection has the own symmetric encryption with a shared secret
3. Information about the encryption algorithm and cryptographic key to be used are negotiated in the client-server connection before the data transfer
4. The negotiation of a shared secret is both secure (since it avoids the Middle In The Middle attack) and reliable (the attacker cannot disturb the communication negotiation without being detected).
5. The authentication of the client (optional) and server (mandatory) identities can be provided with a public-key cryptography.
6. The message integrity check functionality makes the connection reliable

Different methods are supported by TLS the key exchange, data encryption and message integrity authentication.

9.1.1 Protocol operations

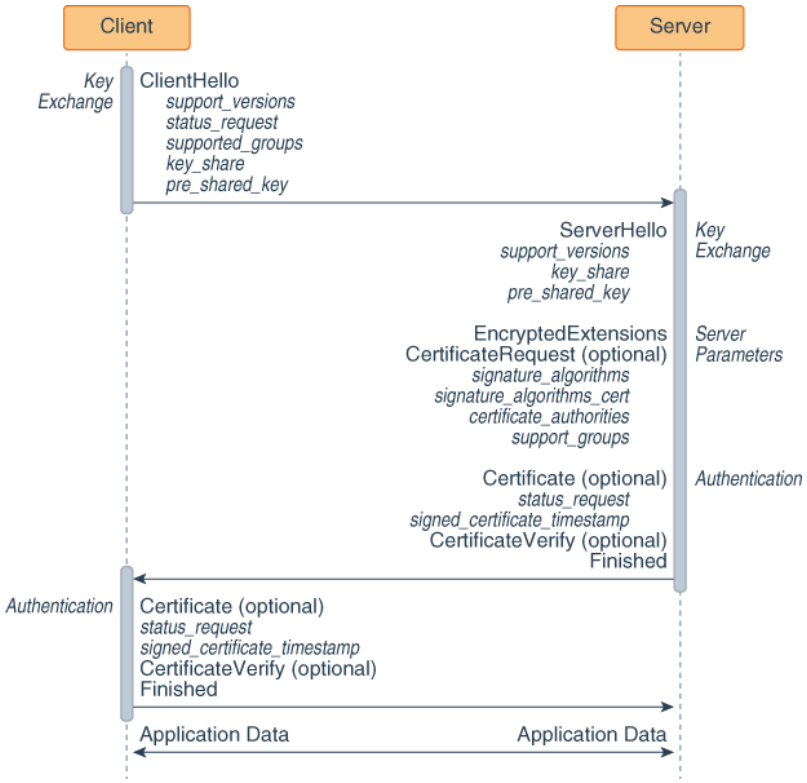
From an implementation point of view, the client-server communications can use different number for TLS connections (e.g., port n. 443 in case of HTTPS) or can foresee specific requests from client to server in order to switch on TLS connection (e.g., sending a STARTTLS message in case of e-mail).
When the choice of TLS is confirmed between client and server, a stateful connection is initiated with a handshaking procedure based on asymmetric cipher messages for the exchange of session-related shared key [48].
During the connection handshake, the client and server need to negotiate and accept several parameters to guarantee the target security levels, as follow:

- The handshake starts when a client sends a secure connection request to TLS server, presenting a list of supported cipher features (ciphers and hash functions)
- The TLS server selects a specific cipher feature and replies to the client

- The server supports identification procedure based on digital certificate, containing all the main information about it, as the server name, the trusted certificate authority (CA) and public encryption key of the server.
- The client accepts and agrees on the certificate
- Before generating the session, keys used for the secure connection, the client needs to:
    - Send to the server a random number encrypted with the public key. The server is the only entity able to decrypt it using its private key). Each session is characterized by a different random number.
    - To use Diffie–Hellman algorithm to generate a random and unique session key for encryption/decryption before exchanging the key.

In this way, the handshake is finished, and the connection can start in a secure way: it is encrypted and decrypted with the session key until the connection is closed.
The TLS handshake fails if one step fails. It means that the secure TLS connection is not created.
The Figure 46 shows the message sequence exchanged for TLS handshake procedure.

Figure 46: Sequence of messages for the full TLS handshake



The TLS protocol cannot be fitted with a specific single layer of the OSI model or the TCP/IP model [49]. Nevertheless, it requires that the underlying transport is a reliable, in-order data stream (e.g., TCP). It means that TLS:

- Is above the transport layer;
- Provides encryption schemes to the upper layers (Note: usually this feature is in charge of the presentation layer)

- Is used by many applications as a transport layer although the applications are able to control the TLS handshakes and to managed the exchange of authentication certificates.

As an example, the TCP/IP protocol stack including some application protocols and TLS is shown in Table 10.

Table 10: presence of TLS in the Internet protocol stack.

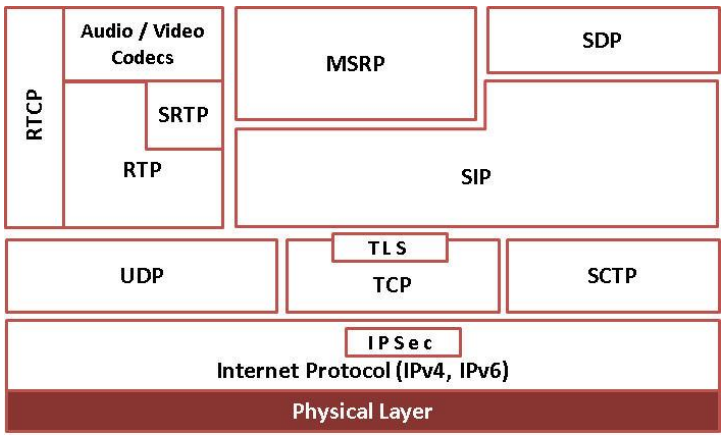
TCP/IP Layer	Protocol
Application Layer	HTTP, NNTP, Telnet, FTP, and so on
Transport Layer Security	TLS
Transmission Control Protocol	TCP
Internet Layer	IP

TLS protocol is the reference protocol of security in Internet world. For this reason, it is always under review to identifies vulnerabilities and to apply the required countermeasures.

9.2 Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) is an application-layer control (signaling) protocol used for initiating, maintaining (and modifying), and terminating real-time sessions with one or more participants. These sessions include Internet voice and video calls, multimedia distribution including instant messaging, and multimedia conferences. It I described in RFC 3261 [50]. In simple terms, the main goal of the SIP protocol is to allow the exchange, among all the involved parties, of the information required to setup the links for subsequent data exchange in accordance with the selected transport protocols.
SIP is not a vertically integrated communications system. Thus, it is generally used in combination with the Session Description Protocol (SDP) for the session description (e.g., the audio codec of the VoIP connection) or other protocols such as the RTP) (RFC 1889) and the Real-Time streaming protocol (RTSP) (RFC2326). In Figure 47 it is reported the protocol stack where SIP is located.

Figure 47: General protocol stack for SIP and other multimedia protocols.



SIP is designed to be independent of the underlying transport layer protocol, and can be used with UDP, TCP and SCTP, even using TLS.
SIP provides five functionalities in a multimedia communication:

- User location: determining the end system
- User availability: determination of the willingness of the called party (callee) to engage in communications;
- User capabilities: determination of the media and media parameters to be used;
- Session setup: "ringing", establishment of session parameters at both called and calling party;
- Session management: including transfer and termination of sessions, modifying session parameters, and invoking services.

9.2.1 SIP protocol operations and corresponding elements

The SIP protocol specifies the exchanged message formats and the sequence of communications for cooperation of the participants, similarly to HTTP.

RFC 3261 defines six methods:

- REGISTER for registering contact information, i.e., bind a SIP URI address to an IP address;
- INVITE: sent by the originating party to initiate a SIP session or to modify an existing session;
- ACK,
- CANCEL: it is used to cancel a pending request, for example, if INVITE was answered with 200OK, but final ACK is missed;
- BYE for terminating sessions,
- OPTIONS for querying servers (or a SIP proxy) about their capabilities.

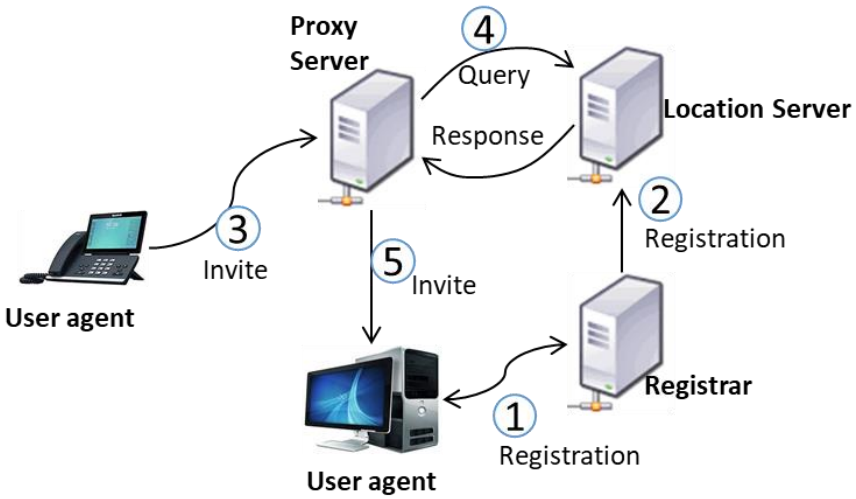
Responses contain the status codes similar to HTTP. In Table 11 the main classes of the status codes of SIP protocol are reported.

Table 11: status code of responses in SIP protocol [50].

Code	Type	Description
1xx	Provisional	Request received, continuing process the request
2xx	Success	The action was successfully received, understood, and accepted
3xx	Redirection	Further action needs be taken in order to complete the request
4xx	Client Error	The request contains bad syntax or cannot be fulfilled
5xx	Server Error	The server failed to fulfill an apparently valid request
6xx	Global Failure	The request cannot be fulfilled at any server

The SIP protocol defines several network elements, described in the following. The **user agent client** (UAC) requires a service function, while the **user agent server** (UAS) responds to a service request. A user agent is a logical network endpoint that sends or receives SIP messages and manages SIP sessions. The roles of UAC and UAS only last for the duration of a SIP session. In fact, agents can implement both roles. When a UAC sends a request, the request passes through some number of proxy servers, which forward the request towards the UAS. When the UAS generates a response, the response is forwarded towards the UAC.
Between the UAC and UAS, the **proxy server** is an intermediary component with the aim of call routing of requests on behalf of other network elements to another entity closer to its destination. SIP proxy servers that route messages to more than one destination are called forking proxies. The forking of a SIP request establishes multiple dialogs from the single request. Thus, a call may be answered from one of multiple SIP endpoints. For identification of multiple dialogs, each dialog has an identifier with contributions from both endpoints.
A **registrar** is a SIP endpoint that provides a location service. It accepts REGISTER requests, recording the address and other parameters from the user agent. For subsequent requests, it provides an essential means to locate possible communication peers on the network. The location service links one or more IP addresses to the SIP URI of the registering agent. Multiple user agents may register for the same URI: all registered user agents receive the calls to the URI. SIP registrars are logical elements, and are often co-located with SIP proxies. To improve network scalability, location services may instead be located with a redirect server.
A **redirect server** is a user agent server that generates "3xx" (redirection) responses to requests it receives, directing the client to contact an alternate set of URIs. A redirect server allows proxy servers to direct SIP session invitations to external domains.
The SIP **location server** is used to resolve a SIP address such as xxxxx@example.com to the IP address of xxxx's device. The typical interaction among some of the SIP elements in the network is shown in Figure 48.

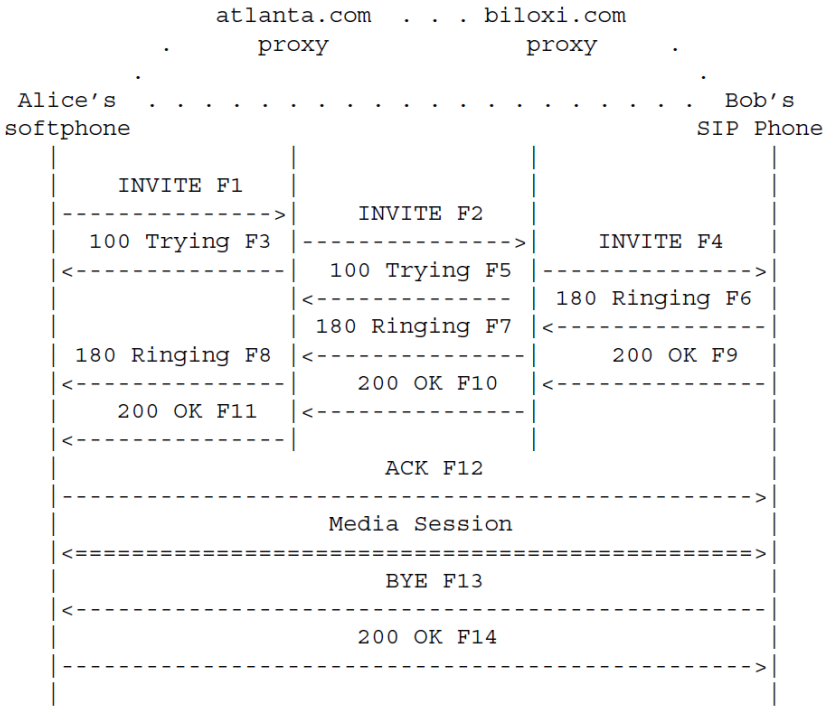
Figure 48: Interaction of SIP elements in the network.



A user agent registers itself to the registrar (step 1), which communicate it to the location server (step 2). Now, another user agent is going to establish a connection with the previous user agent. To this aim, it interrogates its proxy server (step 3). The local proxy tries to know the location of callee, by querying the location server (step 4). By its response, the local proxy forwards the

INVITE to the user agent callee (step 5).
In Figure 49, it is reported the typical example of a SIP session setup between Alice (the caller user agent) and Bob (the callee user agent). It is noteworthy to see the trapezoid communication established by Alice, its proxy, the Bob's proxy and Bob. Once the proxies have sent the invitations, the communication is directly through Alice and Bob, without passing through the proxies.

Figure 49: Example of SIP session setup.



9.2.2 Problems of SIP with NATs and firewalls

SIP and Network Address Translators (NATs)
In current networks, it is usually that a VoIP call will involve NAT and SIP. When a host sends a request from its local network to the public Internet, its IP address is a private one (typically 192.168.x.x or 10.0.x.x.), thus not directly reachable by an external host. NAT replaces the local source IP address with an IP address which is routable on the public Internet, hence the "translation". Unfortunately, SIP embeds address information into the payload of the data packet. The NAT device is going to replace the local source IP address with a publicly routable source address, but only for the IP header on this packet. When the provider's VoIP equipment sees this packet, it may be OK and route a response back as needed. The problem arises is in the second half of the payload – at the SDP. The two VoIP devices will use the SDP information in the INVITE and OK SIP methods to negotiate a path over which the audio stream will travel. The SIP payload is telling the provider's VoIP equipment where to send the audio for the call and this is a local address. This causes problems since the provider's VoIP equipment cannot route the private 192.168.1.x address across the internet which means this call, if left as is, will result in 'no audio' or 'one way audio.' This is a common complaint when NAT is causing problems on a VoIP network.

SIP and firewalls

Another problem is when a user agent is trying to establish a communication with a user in a LAN behind firewalls. Since many firewalls installed today do not support SIP, the inbound traffic will be stopped in order to prevent inbound unknown communications.

9.2.3 SIP solutions for firewalls and NAT traversal

There are several available methods proposed to overcome the NAT/firewall traversal. Unfortunately, several of these solutions have serious security implications. The choice of a method depends on who has the control: the firewall administrator, the user or a service provider. Main solutions have been proposed and summarized in the following [51].

STUN, TURN, ICE

In case of client's control, the IETF proposed three protocols for NAT/firewall traversal: Simple Traversal of UDP through NATs (STUN), Traversal Using Relay NAT (TURN) and Interactive Connectivity Establishment (ICE). These protocols are based on intelligence of the client (which has to emulate the SIP behavior), and on external servers (which have to let SIP signaling and media pass through). These methods do not work in all scenarios and require that firewalls are quite open to allow users to create the pinholes needed to let the communication through.

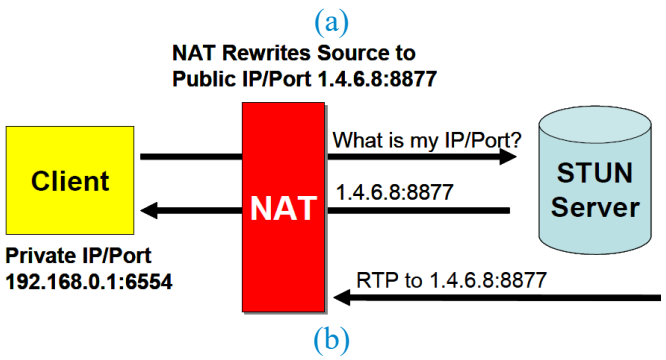
The STUN protocol requires a STUN client on the phone and a STUN server on the Internet. They exchange messages on information about the IP address and ports to receive and send SIP messages. Thus, STUN requires that the NAT device allow all traffic that is directed to a particular port, and that the traffic is forwarded to the client on the inside. This means that the internal client will be exposed to an attack from anyone who can capture the STUN traffic, thus it is not considered a viable solution for enterprises.

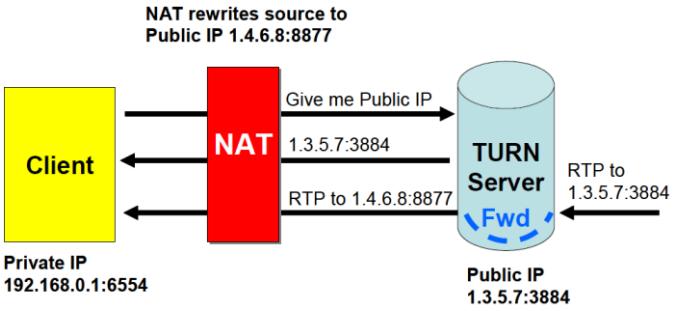
The TURN allows an end point behind a firewall to receive SIP traffic on either TCP or UDP ports. The TURN server acts as a relay. In fact, TURN connects clients behind a NAT to a single peer: any data received is forwarded. This solution is limited applicability in enterprises and hardly scale, since all traffic should pass in the TURN server.

ICE is based on multiple solutions in order to enable the connection, involving also other methods without not rely on the firewall or NAT device. Due to its complexity, it has limited applicability.

In Figure 50, STUN and TURN work is reported.

Figure 50: Client-based SIP solutions: (a) STUN protocol; (b) TURN protocol.





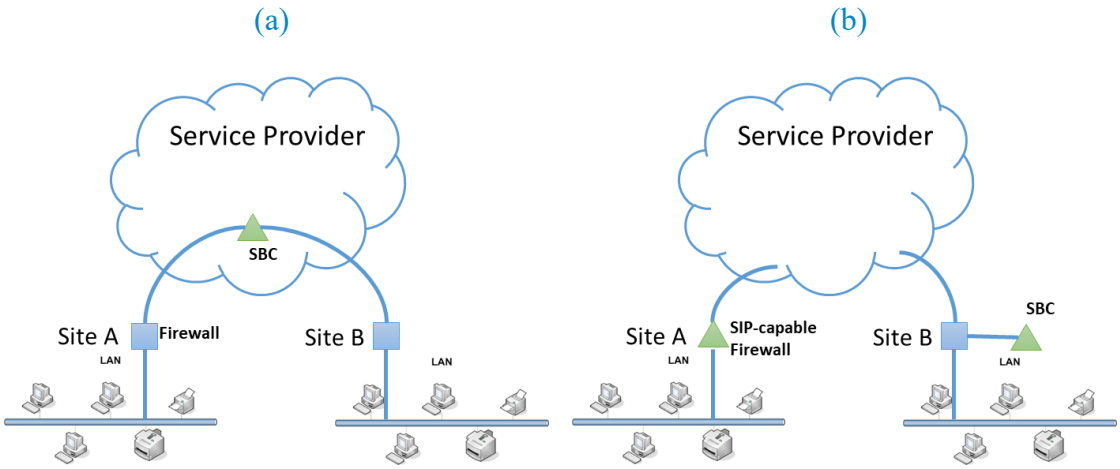
Session Border Controllers at Service Provider

With this solution (see Figure 51.a), the service provider is in control. Most service providers use some sort of session border controller (SBC) in their core network to perform a number of tasks related to their SIP services, for example this by acting as a server component for protocols such as STUN and TURN. One of these tasks of SBC is to make sure that the SIP services can be delivered to their customers. In case of clients do not support solutions as STUN, TURN or ICE, SBC can implement the Far-End-NAT traversal (FENT) technology, which has the of aiding remote SIP clients by transforming any SIP message by rewriting all relevant information and relay media, as well as keeping client on the NATed network reachable.

SIP-capable Firewalls or enterprise SBC

In this case (see Figure 51.b, the firewall administrator is in control, since it is used together with an SBC of the enterprise. The enterprise SBC typically has a built-in SIP proxy and/or back-to-back user agent (B2BUA) functionality to give the flexibility to firewall to maintain control of what is traversed between the LAN and the outside world. In addition, enterprise can deploy a SIP server to communicate over IP with the outside world. as long as the firewall must be SIP enabled.

Figure 51: Solutions for NAT traversal: (a) SBC at the Service Provider; (b) SIP-capable firewalls and enterprise SBCs.



SIP-capable firewalls can be based on the SIP Application-Level Gateway (ALG) architecture (namely ALG-based SIP-capable firewalls), which rewrites SIP packets with the correct IP address information as the traffic flows through them, making sure that they reach the right destination on the LAN or based on a SIP proxy architecture (namely proxy-based SIP-capable firewalls). In the

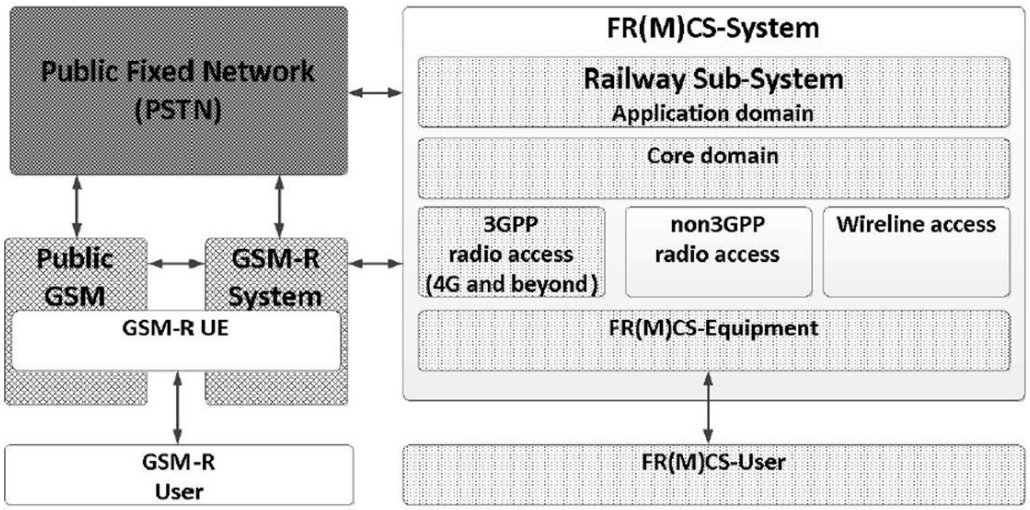
last case, a proxy is designed to briefly stop the packets so that each signaling packet can be inspected before the header information is rewritten and the packets are delivered to the appropriate endpoints. This provides the enterprise with a flexible, controlled implementation of SIP-based communications.

10 Appendix B

10.1 The Future Railway Mobile Communication System (FRMCS)

The Future Railway Mobile Communication System (FRMCS) is considered to be the future worldwide telecommunication system designed by UIC, in close cooperation with the different stakeholders from the rail sector. It is the successor of GSM-R but also it is considered to be a key enabler for rail transport digitalization.
FRMCS relies on mission critical telecommunication standards, consolidated in ETSI rail communications based on latest 3GPP standards. The main advantages are: independence of radio access and packet core network technologies due to standardized core network interfaces for bearer channel setup, quality of service management and broadcasting.
The railway adaption subsystem will become based on 3GPP mission critical framework providing functionalities for broadcasting, QoS management and group communication important for mission critical applications including FRMCS.
Although the specifications are not fully completed, Shift2Rail WP3 project will consider the requirements given by the FRMCS project as input rather than any requirements from the existing EIRENE project (see Section **Errore. L'origine riferimento non è stata trovata.**). It is expected that new requirements raised in EIRENE project is communicated to and managed by the FRMCS project. The principle system architecture of the FRMCS system is depicted in Figure 52 [52].

Figure 52: High level overview of FRMCS system as defined in 3GPP TR 22.889



As shown in Figure 52 the FRMCS is entirely based on 3GPP communication technologies. Wi-Fi, Satellite, 802.11-2016 can be part of FRMCS but the inter-operability among 3GPP and non-3GPP technologies requires standardization (i.e., rules and interfaces) to be defined in 3GPP.
FRMCS is based on bearer flexibility. Thus, a Railway Application may use one or several access systems as appropriate. The connection of FRMCS Equipment to different access systems is dynamic: for example, the most appropriate 3GPP or non-3GPP access technologies are selected automatically, potentially using multiple access technologies for one or more Railway Applications. The set of access systems chosen meets the defined QoS and the service requirements e.g., FRMCS User mobility and connectivity which are necessary to guarantee the functionality. The introduction of a new access system should not negatively impact existing Railway Applications. The approach

taken within FRMCS allows the integration of 3GPP and non-3GPP radio access evolution. The FRMCS objectives are:

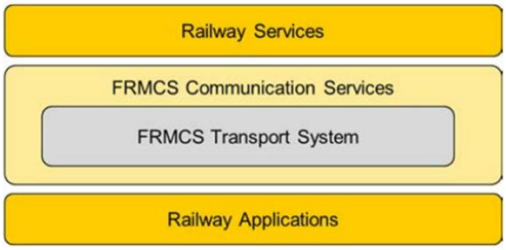
- Provide bearer flexibility to enable independence between Railway Applications and the underlying transport system
- Include wireless and wireline access as transport services
- Enable connectivity and interworking with other external networks and systems such as other FRMCS networks, public communication networks or fixed networks.

External system(s) and application(s), such as train location database(s), are not part of the FRMCS system, but their interfaces are in the scope of the FRMCS system. FRMCS relies on mission critical telecommunication standards, consolidated in ETSI rail communications based on latest 3GPP standards.
The main advantages are: *i.* independence of radio access and packet core network technologies thanks to standardized core network interfaces (developed in ETSI) for bearer channel setup; *ii.* quality of service management and broadcasting.
The railway adaption subsystem will become based on 3GPP mission critical framework providing functionalities for broadcasting, quality of service management and group communication important for mission critical applications including FRMCS.

10.1.1 High level description of FRMCS

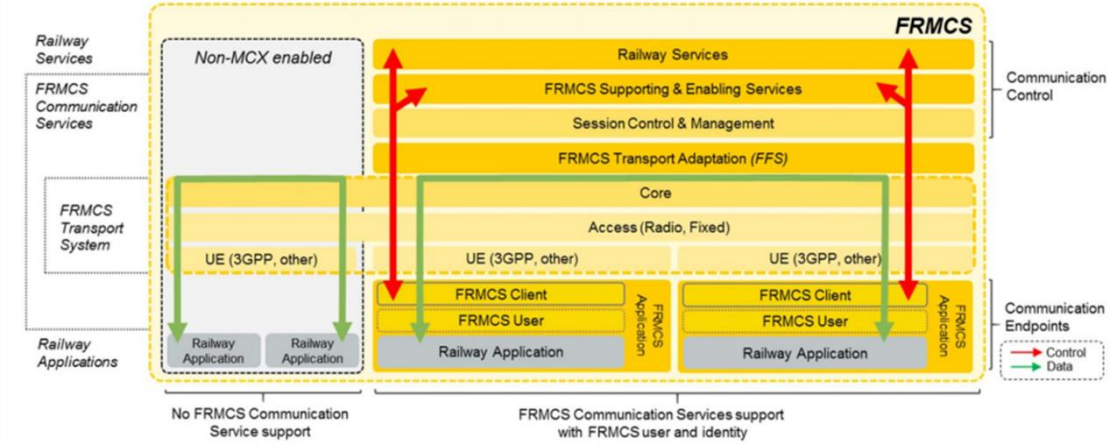
As shown in Figure 53, the FRMCS functional architecture objective is a **clear separation** between FRMCS Communication Services and FRMCS Transport System to enable bearer flexibility/multi-access support.

Figure 53: Layers in FRMCS architecture.



The FRMCS Transport System is embedded within the FRMCS Communication Services, which provide a generic interface for Railway Applications to support the communication between FRMCS Users. Railway Services address specific railway operational requirements, which are not covered by FRMCS Communication Services.
The vertical FRMCS can be split as shown in Figure 54.

Figure 54: Vertical split of FRMCS architecture.

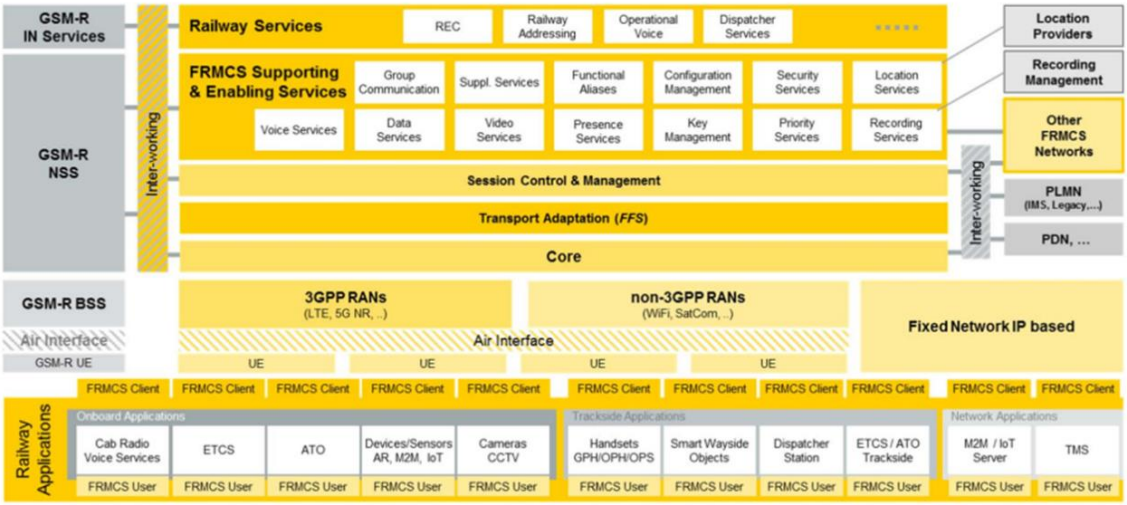


The vertical split of the FRMCS architecture divides the Railway Applications using FRMCS Communication Services and other applications which are non-MCX (non-mission critical) enabled. Railway Applications, which use functions from the FRMCS Communication Services start with registration, authorization and request communication services from the FRMCS System. These Railway Applications are supported by the FRMCS Communication Services and utilize related functionality, including addressing (functional aliases). The FRMCS Transport System consists of the:

- Core and
- Access services

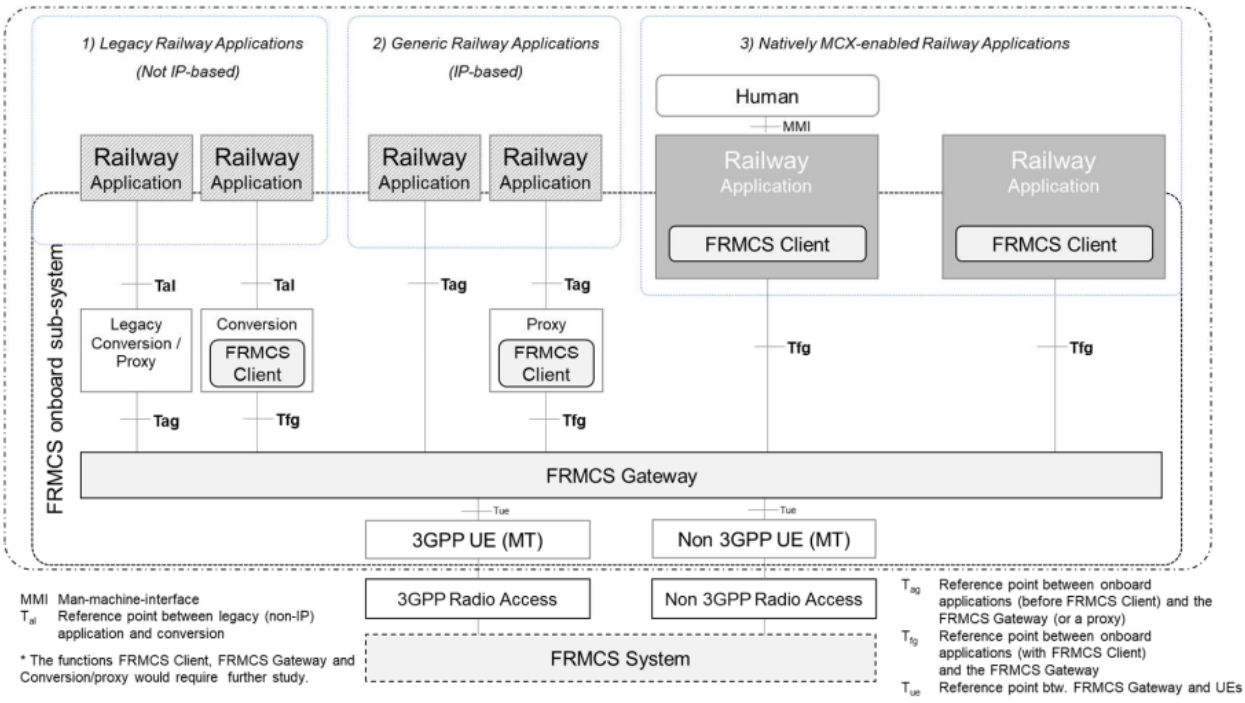
It encompasses various access systems which provide connectivity to the User Equipment (UE). FRMCS Communication Services obtain from FRMCS Transport System the required communication priority, latency and reliability. Non-MCX enabled applications rely on FRMCS Transport services and have no interaction with FRMCS Communication Services.
The functional FRMCS architecture is depicted in Figure 55.

Figure 55: Functional FRMCS architecture.



The picture concerns the applications and functionality that only obtain FRMCS Communication Services. In other words, the Railway Applications that use FRMCS Communication Services are associated with a FRMCS Users and identities. It should be noted that Non-MCX applications do not use FRMCS services. The FRMCS System consists of different functional layers to support Railway Applications as well as supporting the interworking with external systems, including legacy systems or supporting sub-systems.
The on-board system reference architecture represents the on-train FRMCS subsystem is represented in Figure 56.

Figure 56: On-board reference architecture.



It includes:

- On-train Railway Applications supporting voice, data and video services and requiring FRMCS Transport Services, and possibly natively including an FRMCS Client or utilizing a separate FRMCS Client to also use FRMCS Communication Services beyond the FRMCS Transport Services.
- On-train FRMCS Gateway (or multiple gateways, see below) to coordinate the FRMCS Transport and Communication Services between Railway Applications on-board, and between Railway Applications onboard and trackside.
- On-train User Equipment (UE) (or multiple UEs, see below) which supports the FRMCS Transport services for one or more radio access networks.

10.1.2 FRMCS and ACS

The objectives of the two communication systems are identical but they are based on different approaches: bearer flexibility contrasted to bearer independence.
The FRMCS architecture is 3GPP-like and adopts the IP Multimedia Sub-system (IMS). Core domain in FRMCS needs to be defined. A core network is needed to integrate the different technologies (this is not valid for ACS). FRMCS network is divided into three different parts aiming multiple RATs and according to typical 3GPP vision and network architecture:

- Core domain;
- Access Domain;
- User equipment/terminals.

One issue in the adoption of FRMCS is related to the ownership of the entire FRMCS infrastructure. It could be the telecom operator(s) (Telco) or the railway operator. Sharing of the FRMCS infrastructure between the Telco and railway manager could be a viable solution to be investigated. ACS does not distinguish between 3GPP and non-3GPP bearers. ACS interfaces bearers at IP layer, thus realizing the concept of technology independence. FRMCS is mainly based on 3GPP bearers integrating (with proper extension of the 3GPP standards) some non-3GPP technologies such as Wi-Fi and Satellite.
As ACS, also FRMCS is able to use multiple bearers but it distinguishes on the basis of the bearer technology. However, differently from FRMCS, the ACS creates a sort of overlay network (with control and data planes) over the existing/available IP-based bearers, so to provide connectivity services to rail applications. The ACS overlay network is created, controlled and managed by the ACS GWs. Instead, in FRMCS the interoperability among 3GPP and non-3GPP technologies is managed/controlled in the 3GPP FRCMS multi access network.
So far, the FRMCS project is mainly concerned with requirements from the high-speed and mainline domains, which are typically served by GSM-R radio systems. From that perspective, FRMCS is primarily analyzing a successor system for GSM-R. In contrast WP3 scope is wider and in addition covers regional, freight, urban and metro line(s). While the FRMCS project solely relies on 3GPP specifications (even for interoperability) and building blocks to provide bearer flexibility and QoS management for applications, the ACS intends to provide these functions also independently of the 3GPP system. ACS and FRMCS approaches are not mutually exclusive. Nevertheless, Shif2Rail WP3 specification and prototyping activities focus in particular on applicability of FRMCS (USR) use-cases with the adaptable communication system (ACS).
In ACS there is no need to define the on-board architecture for railway apps (as in FRMCS). Anyway, railway app should support SIP for interfacing with ACS for control purposes. Furthermore, the ACS network realized by the ACS-GWs is simpler and more flexible than that realized by FRMCS. Finally, 3GPP technologies in FRMCS are a subset of the bearers that can be included in the ACS system that can easily incorporate alternative bearers technologies without requiring any additional standardization effort.

10.2 CONNECTA and CONNECTA 2

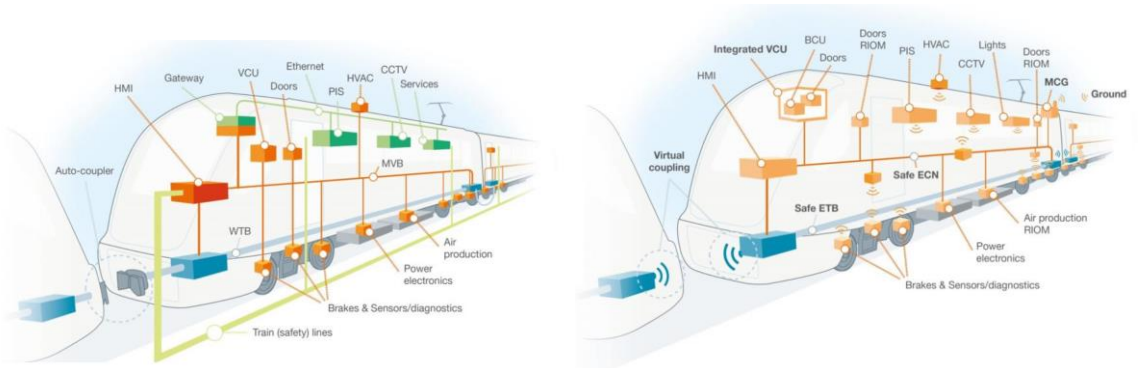
The CONNECTA projects (CONNECTA AND CONNECTA2) belong to Shift2Rail Innovation Programme 1 “Cost efficient and Reliable Trains”, and to the Technical Demonstrator TD1.2. CONNECTA (from 01/09/2016 to 30/09/2018) and CONNECTA2 (from 01/10/2018 to 31/07/2021) have the aim of analyzing the Train Control and Monitoring System (TCMS).
The Project objective is to design and develop two demonstrators of a TCMS new generation (for

urban and regional applications), through innovative features in terms of system architecture, components, electronic braking systems and wireless interconnections.
Two of the main goals of CONNECTA are:

1. to incorporate wireless technologies to train communication network solutions.
2. To provide a train-wide communication network for full TCMS support including the replacement of train lines, connecting safety functions up to SIL4 and support of "fail-safe" and "fail-tolerant" principles, to provide an optimal train network for TCMS and OMTS (On-board Multimedia and Telematics Services) as well as communication mean for non-TCMS functions.

The approach of CONNECTA project is summarized in Figure 57, which shows the current and the future system architecture of TMCS to be analyzed in the projects.

Figure 57: Architectures envisaged on CONNECTA project.



Current TMCS Future TMCS

From a technical point of view, TCMS is a crucial communication component both for the entire railway sector and the vehicle performance. It is deputed to:

- Manage all on-board information
- Support and take all control decisions about the train
- Perform communication between equipment, devices, cars and vehicles
- Communicate with different train subsystems

Most TCMS currently used are based on the IEC-61375 TCN standard, whose first version was released in the late 1990s. The network is formed by two buses: Multifunction Vehicle Bus (MVB) and Wire Train Bus (WTB). Since they are railway proprietary and cannot be adapted in other sectors, they are expensive, with a limited performance in terms of data rate provided (1.5 Mbit/s for MVB and 1.0 Mbit/s for WTB). Current research activities are focused on a standard physical TCMS network for train controls and an additional network (usually Ethernet-based) for additional functions, with a consequent increase of the system complexity. Moreover, some TCMS solutions are able to only meet limited safety requirements (SIL 2), not sufficient to allow the train movement on a safety lines (i.e., more cabling) or the data transmission for safe critical applications (SIL 4).
The next generation of TCMS must have new features, as:

- improvements on wireless connections;
- "driven-by-data" commands;
- seamless coupling;
- improvements on throughput and reliability;

- innovative system architecture able to distribute functionalities and to support safety and security;
- improvements on train-to-ground connectivity, supporting the self-configuration in a better way.

The CONNECTA Shit2Rail project defined and developed a new TCMS architecture, providing both an efficient interoperability combining wired/wireless communications and a "driven-by-data" approach. Trains are connected among them within a safe, secure, and reliable communication infrastructure in a digital ecosystem.
CONNECTA project approach is based on the evolution of the future technologies involved in TCMS:

1. Connected trains: wireless communications between vehicles and/or to/from ground
2. Function-related architecture: any processor can provide function from other system
3. Communication technologies able to support safety critical functions
4. Time and cost-efficient validation and homologation

10.2.1 CONNECTA TMCS and ACS

The studies, development and testing on the future TMCS and ACS are both supported by Shift2Rail community. They are the building blocks of the future communication infrastructure for the railway system. In CONNECTA, CONNECTA-2 and CONNECTA-3 are an integral part of the development activities undertaken under IP1 and X2Rail-5 S2R and the corresponding programmes. The problem of integration between the CONNECTA TMCS and the ACS is an interesting topic which is outside the perimeter of the AB4Rail project. The actual Wireless TMCS proposed in CONNECTA that is evolving in CONNECTA-2 and CONNECTA-3 is mostly based on LTE well proven radio technologies. We believe, the possibility of introducing the ACS paradigm to support the creation of bearer independent Wireless TMCS could be of potential interest to Shift2Rail so to avoid the W-TMCS to be tied to a specific radio technology for the next years.

10.3 DEWI/SCOTT Projects

DEWI (Dependable embedded wireless infrastructure) project started in 2015 and it was concluded in 2017. Under the ARTEMIS Joint Undertaking and national programmes, the DEWI project aimed to design and develop a wireless system for rail safety applications based on the two following features:

- Gathering the data related to the train (e.g., integrity, structure, …) and managed by the European rail transport management system (ERTMS) for Level 3 rail applications.
- Facilitating the freight management.

10.3.1 The DEWI approach

The DEWI approach is based on a Wireless Sensor Network (WSN), deployed within the train (e.g., wagons and locomotives) and connected to a gateway, committed to store and manage the received data from WSN.
The DEWI architecture consists of two main subsystems the DEWI Bubble and the DEWI Gateway:

- DEWI Bubble provides services for controlling train completeness by monitoring and controlling network nodes. It should also provide automatically relevant train composition data (such as length, weight, max load and breaks).

- DEWI Gateway implements – a safe train integrity control system using the Smart Integration Platform. It provides safe interfaces to existing train control systems (e.g., ETCS). This gateway provides the calculation required for a safe train integrity.

The DEWI Bubble is a locally adaptable wireless system, formed by three main elements:

- Sensor and actuators;
- Gateways (for connecting several bubbles among them or to external systems);
- Both human and machine users.

The DEWI Bubble provides important features as:

- Limited wireless internal and external access in a local environment;
- Secure and efficient wireless communication for safe applications;
- Better access to intelligent environments;
- Enhanced self-organization, reconfiguration, resilience and adaptability;
- Open solutions/standards for application interoperability.

In addition, DEWI Bubble provides more flexible functionalities for data acquisition, aggregation and fusion, smart and intelligent architecture, smart configuration, security features as data protection, authorization, energy smart management, more safety and more interoperable standards for wireless communications and sensors.
The main communication technologies used by DEWI Bubble are mainly short range, as IEEE 802.11, ZigBee, IEEE 802.15.4, Bluetooth, Near Field Communication (NFC), 6LoWPAN, ZWave, WirelessHART and DLNA. The TETRA, TETRAPOL (PMR), WIMAX and even LTE have been considered in DEWI for train-to-railway infrastructure communications (e.g., train-to-TMS, train-to-RBC etc.).

Use Cases in DEWI

Within the rail contest, DEWI project defined the scenarios reported in Table 12:

Table 12: Scenarios defined in DEWI project

ID	Name	Details
4.1	Train Integrity Detection System	The system must assure the train completeness
4.2	Train composition Detection System	The system has to be able to collect data coming from the train (e.g., wagons length or eight) and to transfer them to the on-board units
4.3	Smart Integration Platform	The system has to store and manage the collected data from sensors and to transfer them to the train systems.
4.5	WSN for freight advanced monitoring and management	The system is deputed to perform the freight monitoring

10.3.2 SCOTT Project for Rail

Based on the output from DEWI project, SCOTT (SECURE CONNECTED TRUSTABLE THINGS) project started in 2016 aiming to continue the research on safe and secure communications for rail applications.
SCOTT project consortium is formed by 57 industries and research partners from 11 European countries (Brazil included) operating in different IoT industrial sectors, as building, automotive, aeronautics, rail and healthcare.
The main objectives of SCOTT Project for Rail are as follow:

- to build up a cloud platform for collecting and managed all the information related to the entire rail infrastructure in order to provide added value services [WP18]
- to provide innovative wireless communications features both to the on-board and on-track systems [WP18]
- to enhance the power consumption increasing the energy autonomy of on-track devices. [WP18]
- to provide safe and secure communications, both Infrastructure-to-Vehicle (I2V) and Vehicle-To-Infrastructure, (V2I), compatible with technical standards of rail communications [WP18, WP20]
- to provide safe and secure Vehicle-To-Vehicle (V2V) communications technology compatible with technical standards of rail communications allowing the virtual train convoys [WP19]

A prototype of wireless communication platform is developed by SCOTT project for supporting safety applications within dangerous areas (e.g., level crossing without barriers or worker on tracks, etc.). The platform aims to enhance the actual rail infrastructure in a safe way, avoiding critical zones. The develop platform can also support Virtual Coupling functionalities (as safe maneuver coupling and uncoupling between train sets) and rail logistics and maintenance applications. Finally, the developed platform is able to manage the power consumption in an efficient way through the involvement of different energy sources.
The use cases for railway considered in the SCOTT project focus on:
UC18 Autonomous Wireless Network (AWN) for Logistics and Maintenance: The main objective of this Work Package is to develop an AWN for Rail Logistics and Maintenance that will provide all the information from the infrastructure and the train to a centralized system (Cloud, TMS) using wireless technologies that will reduce in both, wire costs and civil works going towards I2I communications.
UC19 Smart Train Composition Coupling (STCC): This system addresses the possibility to achieve a Smart Train Composition Coupling between two or more trains in order to get a unique composition. It would allow the circulation of the trains keeping a shorter distance between them, and as consequence, it would be possible to increase the capacity of the lines.
UC20 Trustable Warning System for Critical Areas: The main innovation consists of a smart sensor system that detects dangerous obstacles on the tracks and replacing wired functionality and adding warning functionality for the train driver
Inside these use cases different developments are envisaged. Major points, as part of the development of Use Cases 19 and 20 further improvements on the TI developed during the DEWI project are planned.

10.3.3 DEWI/SCOTT Projects and Shift2Rail

The innovative solutions developed by DEWI and SCOTT projects within S2R IP5 for Freight Environment are oriented to the future rail services and to the digital ecosystem within the rail sector. The focus is both on technological and economical features and is compliant with the safety and security requirements of rail applications.
Different protocols compliant with OSI model for also guaranteeing the train integrity are analyzed

by DEWI (https://cordis.europa.eu/project/id/621353) and SCOTT (https://cordis.europa.eu/project/id/737422) project. The distribution, access and transport layer are investigated, together with Sensor Network solutions, the Semantic Sensor Network Ontology and corresponding procedures. The results from DEWI and SCOTT are used in IP5 FR8RAIL project (https://cordis.europa.eu/project/id/826206/it) from the stack protocol of communications, ontology, and data models. The ontology defined in IP5 FR8RAIL for freight services are the input for the Canonical Data Model defined by S2R for freight rail context.

10.4 Summary and conclusions

In this Section we highlight the possible relations among the AB4Rail and ACS to the other related projects described in this Appendix.
In Table 13 we evidence the primary and secondary objectives of the projects reviewed in this Appendix in terms of the railway communication aspects that have been or are currently under study and development within these projects.

Table 13: summary of (primary or secondary) objectives for the considered projects.

Project/Main objective	Train-to-Infrastructure (and vice versa) communications	Intra-train (inter-train if any) communications
CONNECTA	Secondary	Primary
DEWI	Secondary	Primary
SCOTT	Secondary	Primary
3GPP – FRMCS	Primary	Integration with CONNECTA/DEWI/SCOTT platforms should be envisaged
X2Rail – ACS (Projects)	Primary	Integration with CONNECTA/DEWI/SCOTT platforms should be envisaged

In X2Rail 1-5 several primary and complementary projects have been devoted to the study, definition, development and demonstration of the future railway communication system i.e., the ACS for train-to-infrastructure (and vice versa) communications. At the same time ERA, UIC and 3GPP are conducting research and development activities on the FRMCS. These activities are conducted in parallel and the two proposed systems are developed in accordance with two different visions i.e., the TELCO view (for FRMCS) and over the top (OTT) view (for ACS). A lot of attention is currently devoted to the effective integration of the two railway communication systems proposals. It should be remarked that ACS as well as FRMCS focuses on the creation of a TLC infrastructure for railway use, effectively creating a layer of abstraction between applications and communication technologies. The ACS was mainly designed to offer wide and flexible connectivity for all services / applications on board the train that require train-to-infrastructure communications (and vice versa) and this for all TLC services (described in FRMCS and extended) regarding human-to-human, human-to-machine and machine-to-machine communications.
On the other side, CONNECTA (2 and 3), DEWI and SCOTT projects mainly deal with aspects related to intra-train (and also inter-train) communications for safety operation and monitoring of the train in accordance with different approaches and views as indicated in the previous paragraphs

in this appendix. However, as indicated in Table 13 it should be remarked that none of these last three projects explicitly focus on the communication system to be used for train-to-infrastructure communication (e.g., TMS, RBC etc.) and vice versa. In fact, in these projects in order for the train to communicate outside data gathered from its internal sensor network, it is assumed the gateway can directly interface with the existing and available radio technologies such as GSM-R, TETRA, TETRAPOL, LTE. In particular, it is assumed the train can access to these radio technologies using the available standard modems and communication protocols envisaged for each single technology. This means that the problem of study and developing a railway train-to-infrastructure communication able to offer connectivity for supporting the several railway applications services (indicated in [52] and [3]) is not the objective of these projects.
In principle, the deployment of the ACS includes one OBU indicated as ACS-GW. This system can act as a natural support for providing external connectivity to the CONNECTA and/or DEWI/SCOTT platforms inside the train. In addition, devices and sensors belonging to these platforms could use the routing/gateway features provided by the on-board ACS-GW to achieve evolved connectivity taking into account that the ACS-GW can efficiently and intelligently interface at IP level any TLC standard. It means that the single on-board device or sensor belonging to CONNECTA and DEWI/SCOTT can communicate with any other device/sensor using its own communication technology by routing its message through the ACS-GW. To this purpose we retain it could be of interest to investigate within the CONNECTA and SCOTT projects the problem of interfacing these platforms with ACS i.e., with the on-board ACS-GW to provide internal connectivity among sensors/devices and to provide train-to-infrastructure communications.