ns-3 Based Framework for Simulating Communication Based Train Control (CBTC) Systems

Abdulhalim Dandoush, Alina Tuholukova, Sara Alouf, Giovanni Neglia, Sebastien Simoens, Pascal Derouet, Pierre Dersin

ns-3 Based Framework for Simulating Communication Based Train Control (CBTC) Systems*

Abdulhalim Dandoush†, Alina Tuholukova, Sara Alouf, Giovanni Neglia
Inria, Sophia Antipolis, France
name.surname@inria.fr

Sebastien Simoens, Pascal Derouet, Pierre Dersin
Alstom Transport, France
name.surname@transport.alstom.com

Abstract

In a Communication Based Train Control System (CBTC), a central zone controller server (ZC) exchanges signaling messages with on-board carborne controllers (CC) inside the trains through a wireless technology. The ZC calculates and sends periodically to each train its Limit of Movement Authority (LMA), i.e. how far the train can proceed. A CC triggers an emergency break (EB) if no message is received within a certain time interval to avoid collision. Clearly, it is not desired to have an EB due to signaling messages losses (called spurious EB) and not to real risks for the trains. Quantifying the rate of spurious EBs and predicting correctly CBTC system performance are hard tasks with important industrial relevance.

This work aims at filling this gap using simulation to better predict CBTC system performance and avoid extra provisioning before deployment. A typical CBTC system implementation for metro by Alstom Transport is considered. New ns-3 modules (CBTC protocol, Video traffic generator, multi-channel scanning mechanism, 3D antennas patterns) are developed and a piece of existing code is enhanced. The simulation is also used to investigate the dimension of the radio access networks in a realistic environment (specific modems and access point antennas, radio frequencies, train and track models), another aspect also ignored in the previous literature. Last, our approach can be useful to validate some analytical works.

Keywords

Communication Based Train Control, Signaling System, ns-3, Video Streaming Generator, Directional Antennas, Performance Evaluation

*This is an author version of the 8-page paper that has appeared in the Proceedings of WNS3, June 15-16, 2016, Seattle, WA, USA.

†This author is now with ESME Sudria, Paris Sud, France, email: dandoush@esme.fr.

1 Introduction

In the traditional train control systems track lines are divided into fixed blocks. Each of them can only contain one train at a time to avoid collision. Improving the traditional system is needed due to the increasing demand for efficient mass transit transport. In other words, we have to utilize train lines more efficiently taking advantage of the new on-board processing and communication technologies. In fact, the moving-block control allows to safely increase the number of trains traveling over a track line as it takes into account the real-time information about train position rather than the information provided by the fixed blocks of large enough length. The moving-block control is being deployed as CBTC for urban mass transit system.

In moving-block systems, the zone controller (ZC) computes the Limit of Movement Authority (LMA) for each train based on the information received for all the trains in its zone. Then, it sends the End-of-Authority (EOA) control message carry the LMA to the on-board carborne controllers (CC) in the train through a wireless technology. Currently, Alstom uses WiFi technology. However, the use of 4G/5G may be considered for the future. To increase the system reliability against losses and delays, the messages can be sent redundantly through two separated networks called red and blue network. To avoid the risk of collision, CC will stop automatically the train if no valid EOA is received during a given time interval by triggering an Emergency Break (EB). Clearly, it is not desired to have an EB due simply to signaling messages losses (called spurious EB) when there is no eventual risk for the train. For this reason, the so-called performance based contracts (similar to service level agreements for network operators) can bind rail transport companies to specify the maximum number of spurious emergency brakes over a given period of time.

Therefore quantifying the rate of spurious EBs and predicting correctly CBTC system performance to avoid extra provisioning before deployment are important industrial and research problems. Until now this was only done through an-

alytical approaches that ignore many important aspects such as packet losses, handovers and congestion due to bursty traffic. The more reliable solution at the moment is based on costly field measurements of deployed systems. For that reason we decided to develop a framework that helps the evaluation of the system performance and the prediction of spurious EBs.

To that end, we have chosen the network simulator ns-3 [4] because it is a GPLv2 licensed and an open source discrete-event network simulator supported by Inria and the University of Washington.[1] It is highly modularized and thus can be easily modified or extended. In addition, for our industrial project, ns-3 includes many useful modules for the wireless channel, modeling co-channel interference, transmission error and the propagation, as well as mobility models and the TCP/IP protocol stack.

However, the current ns-3 version is missing some functionalities needed to simulate train-trackside communications. Thus, the need for implementing new modules and enhancing some existing ones. Some modules cannot be shared at this stage for confidentiality reasons. The current shared code of this work is available at [5, 6].

The paper is organized as follows. In Section 2 we introduce the new CBTC and MPEG video traffic generator modules. We describe the details of the CBTC protocol, the Train and Track objects and discuss their implementation and usage in Section 2.1. Section 2.2 presents the design and the implementation of the new video streaming generator. A brief description of the 3D Antenna module is introduced in Section 2.3. Selected results to illustrate some issues that can be addressed using our ns-3 based simulation are presented and discussed in Section 3. Section 4 reviews some related works and Section 5 concludes the paper. Last, the acronyms used in the paper are listed in Appendix A.

2 New and Enhanced Modules

In this section, we describe briefly the new modules and enhancements we have integrated within ns-3 in order to correctly simulate the train systems.

First, there are no ns-3 modules for train and track, and—as expected—no implementation of a moving block control system in ns-3. Second, there is no configurable and flexible video traffic generator that can generate traffic based on particular values of the basic characteristic of MPEG-compatible cameras such as the group of pictures (GOP) length, number of frames per second, average bit rate, etc. In fact, trains and ground controllers broadcast two video traffic profiles. The Platform TV flow is a video flow in the ground-to-train direction (e.g. the control monitor in the driving cabin). Closed-circuit television (CCTV) flow is the second type of video flow in the train-to-ground direction (e.g. video surveillance systems). Platform TV and CCTV are MPEG-4 video traffic and the traffic shape can be changed from camera to another one due to different resolution, frame rate, compression method, motion degree, and image quality settings. It is particularly important for the application we are considering to simulate these video traffic flows because they interfere with the critical signaling messages on the wireless link and they require large bandwidth.

Also, the current version is missing some functionalities that we need to simulate train-trackside communications. ns-3 considers only omni-directional antennas for the Yans-WiFi model even if some theoretical directional antenna models are implemented for the simulation of other wireless technologies. However, the gain of the transmitter/receiver antennas is a key parameter for the train system because it changes the received signal strength (rss) and then the signal to noise (SNR) ratio. SNR affects the packet loss probability, the behavior of the handover process, and many of other parameters such as the distance between two APs on a track. Moreover, ns-3 does not yet implement a multi-channels scan mechanism of beacon signals' strength transmitted by access points (APs) operating on different frequencies, which is necessary for a realistic layer-2 handover implementation in WiFi. In addition, ns-3 does not implement a switch object because the impact of layer 2 switching operations on the network performance is assumed to be negligible. Hence, changing routes after a handover at the MAC level as done in the real implementation, that we consider, is tricky. Using the layer-3 routing algorithms (e.g. OLSR, AODV) as a workaround is not an option because the convergence to the correct path may require a few seconds causing the loss of several control messages (major limitation for CBTC).

2.1 New CBTC Modules

In this section we describe the detailed operation of a moving block system considering as reference the specific CBTC implementation by Alstom Transport.[2]

Figure 1 shows a typical messages' exchange between the on board controller (the CC) and the ground controller (the ZC). Observe that both the controllers operate in discrete time on the basis of clock periods of hundreds of milliseconds. This is due to the fact that they are actually e-out-of-f voting systems where different processors perform in parallel the same calculations and a time-slotted operation simplifies the synchronism of the processors. The clock periods at the ZC and at the CC (respectively T_{ZC} and T_{CC}) are in general different because the subsystems are provided by different vendors and also because they have different computational loads during one period.

The most important CBTC messages are location reports (LOC) and end-of-authority ones (EOA). A LOC is a mes-

[1] A main objective of Inria in the Inria-Alstom Transport lab is to apply modeling and simulation tools in industry.

[2] The parameters' values have been slightly changed and some specific implementation details are hidden to protect the industrial know-how of the company.

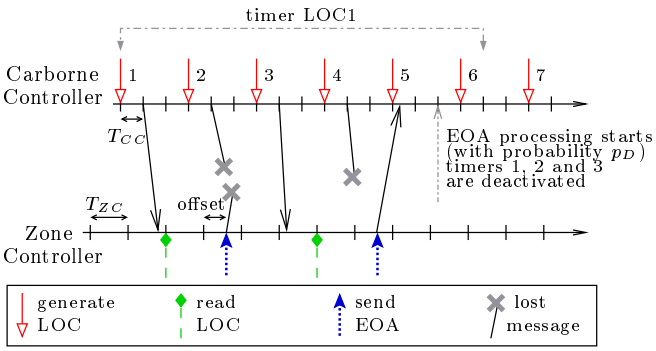


Figure 1: Illustration of LOC-EOA exchanges.

sage periodically transmitted from the on board CC through the Data Communication Sub-System (DCS) to the ZC. The message is actually sent twice to increase reliability through the blue and the red radio networks (i.e. two separated WiFi networks with separated Access Points APs and two different on-board modems OBMs inside each train). We denote by TRE (Track-side Radio Equipment) a pair of redundant APs, one blue and another red AP that are installed at the same location along a track. The first LOC arriving at the ZC is processed. Each LOC is acknowledged by an EOA message in the reverse direction (again sent through the two networks). The EOA indicates to the CC how far the train can advance. For each generated LOC, a timer is activated. The timer expires when its value exceeds a validity duration (TM) before it can be acknowledged and therefore an emergency break (EB) is triggered to avoid collision. The details of the protocol is described hereafter.

2.1.1 Implementation Overview

We created new modules/Objects as follows: (i) a separate module called "cbtc-manager" lives in the directory "src/" and (ii) new application level modules "LocEoaClient" and "LocEoaServer" live in "src/applications". The cbtc-manager module consists of the following objects:

1. Train: It contains mainly two OBMs (red/front and blue/back OBM) with a CC server (i.e. Nodes, NodContainers, NetDevices, Interfaces, smart pointers, etc.). A train has several properties such as ID, length, initial position, velocity, and direction.
2. Track: It defines the required number of TREs along a track. The code is optimized such that referring initially only to 5 TREs (10 APs) nodes, and then reuses/cycles them to extend the track length as the train moves. This is important to be able to simulate long track without any "out of memory" problem or execution time limitation. The attributes of this object are the number of the required TREs, a flag to activate the reuse/recycle feature, the position of the first TRE, the distance between two TREs, the distance between twp APs of the same TRE, and the velocity of the train (for recycling the APs at appropriate times). The object will change the APs positions using their mobility model smart pointer.
3. TrainCcServer: It handles the received EOAs by LocEoaClient application (sent by the LocEoaServer application), manages the clock and the timers for the CC, activates or deactivates an EB and calculates the EB rate.
4. CbtcManager: The objective of this object is to simplify the creation of the simulation scenarios by automatically creating the networks topologies (on-board LAN, radio network, ground LAN). It has member functions to create all the networks elements, interconnect them and install the applications on. In particular, it will do the following: (i) create all the nodes in the scenario (e.g. the Zc server, APs and OBMs of trains); (ii) create the wired and wireless channels that interconnect the nodes (e.g. Ethernet, a ppp link and a WiFi connection); (iii) create the Net devices and attache them to the corresponding nodes and channels; (iv) identify the PHY/MAC properties of TREs/APs and OBMs and install the right antennas; (v) create and install a mobility model, internet stack and routing instances for the created nodes. Then, in order to run CBTC protocol the application LocEoaServer should be created and installed on the node that corresponds to the ground ZC (ground server) and the application LocEoaClient needs to be installed on the nodes that correspond to the OBMs of each train (back and front OBM). A TrainCcServer object is then created to refer to the OBMs and the LocEoaClient applications installed on them, and to handle the EOA messages sent by the ZC server (LocEoaServer) and received by the front and back OBMs (LocEoaClients). An example of the usage is given in the next subsection.

The LocEoaClient, LocEoaServer modules with their helpers represent the CBTC message generators. Table 1 depicts their attributes with the default values. The message exchange between these two applications and the TrainCcServer instance works as follows:

1. a LOC is generated at the CC every T_{LOC}, multiple of the CC clock period T_{CC},
2. the LOC (say LOC k) is ready to be emitted and passed to the DCS after a processing delay equal to T_{CC} since its generation,
3. the delivery delay introduced by the DCS is random and depends on the network conditions,
4. at the ZC the LOC is available for computing at the next tick of the local clock,
5. the computing time at the ZC required to process the LOCs from all the trains in the zone and generate the corresponding EOAs is T_{ZC},

6. the EOA k is emitted within the next cycle of the ZC at an offset O depending on the train (the ZC sends sequentially the EOAs to all the trains in the zone),
7. the EOA is delivered to the CC after a random delay depending on the network conditions,
8. at the CC the EOA gets in a processing queue, at the next tick of the CC clock the most recent EOA present in the queue is processed unless there are higher priority tasks arrived during the same CC clock period (which happens with probability p_D); in any case an EOA processing is not delayed more than an additional CC period,
9. the EOA k is actually processed only if it remains valid until the end of the current CC clock; once processing starts, all the pending timers for older LOCs (i.e. LOC h for $h \leq k$) are deactivated,
10. if the timer of a LOC is not deactivated before its expiration, the EB procedure is triggered.

2.1.2 Building Scenarios

We show in this subsection how easy it is to create a scenario script using the cbtc-manager rather than implementing a scenario from scratch (i.e. creating nodes for TREs and Trains, Containers, Network topology, channels, Antennas, mobility, internet stack, interfaces, etc).

We first describe briefly the creation of a default scenario template that can be modified easily according to the needs. The default scenario template has one train that moves along the track that is simply a line. The track consists of 5 TREs. In order to create this scenario we need to create the object of CbtcManager, then create the Train object and pass its reference to the CbtcManager instance. After that, we call the method DefaultScenario() of the CbtcManager as follows:

```
Ptr<CbtcManager> manager = CreateObject<CbtcManager>();
Ptr<Train> train = CreateObject<Train>();
  manager->CreateTrainTopology(train);
  manager->DefaultScenario();
```

In Table 1 the attributes of CbtcManager and their default values are presented.

Now, we can easily change the default parameters values using Config::SetDefault() or SetAttibute() methods.

To add CBTC traffic it is enough to call the method SetUpLocEoaApplication() that will create and install the LocEoaClient/Sertver applications. The easiest way to change the default behaviour of CBTC protocol is to modify the values of the attributes of the LocEoaClient and LocEoaServer applications depicted in Table 1.

An important point is to activate the TREs reuse/cycle in order to decrease the memory usage and optimize the ns-3 execution time as we explained in the Track object. This can

Table 1: Attributes of LocEoaClient, LocEoaServer and CbtcManager.

Name of the attribute	Description
Attributes of LocEoaClient	
LocProcessingDelay (200ms)	The time to wait after generation of the LOC and before its emission
InputProcDelay (5ms)	The time to process the input
ClockPeriod (200ms)	The period of the clock
Locfrequency (3)	The number of ticks to wait before generating new LOC message
ProbStartProcEoa (0.99)	Probability to start the processing of the EOA on the next tick
TimeOutValue (5s)	The time during which the generated LOC is valid
OutputDelayMin (5ms)	Minimal output delay
OutputDelayMax (40ms)	Maximum output delay
ObmId	The ID of the OBM
TrainCC	Train CC server object
Attributes of LocEoaServer	
InputProcDelay (12ms)	The time to process the input
ClockPeriod (275ms)	The period of the clock
ProcessingDelay (275ms)	Time to process the received LOC
Offset (40ms)	The offset delay for sending the EOA response
PacketLoss (0.1)	Packet loss Probability
OutputDelayMin (5ms)	Minimal output delay
OutputDelayMax (40ms)	Maximum output delay
Attributes of CbtcManager	
NumberOfTre (5)	The number of the TREs along the track
NumberOfChannels (5)	Number of channels that are used for TREs
SimulationTime	The total time of the simulation
EnergyDetection-Threshold (-80dbm)	Energy detection threshold
CcaMode1Threshold (-90dbm)	CCA Mode 1 Threshold
ObmTxPower (16dbm)	OBM transmission power
ApTxPower (16dbm)	AP transmission power

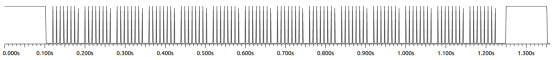


Figure 2: An example of a generated GOP.

be done easily by setting the corresponding boolean property to true as follows:

```
Config::SetDefault ("ns3::Track::DoCycle",
    BooleanValue (true));
```

Note that when we have several trains that have different directions on the same track, we cannot activate this property.

Last, we can add MPEG video traffic generator and a UDP server applications to the ZC server and the OBMs nodes.

During the simulation, if we want to print some information such as the current EB rate, or the association/disassociation moments with train positions, we simply call the corresponding member function of the cbtc-manager.

```
manager->PrintTime (periodToPrint);
manager->PrintEbRate (periodToCountEB);
manager->PrintAssocInfo();
```

A complete example is located at (/src/cbtc-manager/examples/cbtc-manager-example.cc).

2.2 New Video Traffic Generator Module

The goal of the video generator module is to provide a flexible configurable packet generator of MPEG-4-like UDP traffic. The source code for the new module called (MpegPktGenClient) lives in the directory "src/applications". A helper module is also implemented to facilitate its use. We can change the video streaming traffic shape, i.e. the group of pictures (GOP) structure, by simply changing the key parameters values of the simulated camera explained hereafter. The Video generator client can work with any UDP based server like the already implemented UdpServer. An example of usage, expected results, and a brief analysis of the output pcap trace are presented in the next subsections. Table 2 summarizes these attributes with their typical values.

2.2.1 Implementation Overview

The methodology to create video streaming is: (i) to build the GOP structure (e.g. the size and the type of frames in the GOP and the spacing between frames) based on the attributes values and save the characteristics in the text file pointed by the user or in the mpeg-4.dat if no name was provided by the user were; (ii) calculate the number of packets in each frame and the inter arrival time between two packets of the same frame type; (iii) schedule the sending events of the generated packet at the appropriate moments; (iv) repeat the process until the stop time of the application.

Detailed calculation We determine the moments of the sending events as follows[3].

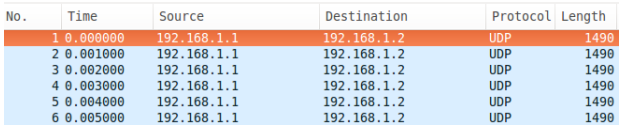
- Calculate the duration of a GOP "burstPeriod" in seconds by dividing the values of gopLength over the imageRate.
- Compute the burst duration of the I-frame: burstDuration = iFrameSize/peakBitRate.
- The amount of data in each GOP is computed as follows: avDataSizeInGop = avBitRate * burstPeriod.
- Subtract the I-Frame-Size from the total amount of data in a GOP to get the summation of the P/B-Frames sizes (sum_frames_P_sizes).
- The size of each P/B-Frame and the bitRate at which we send them are calculated as follows:
  pFrameSize = sum_frames_P_sizes / (gopLength - 1) (Bytes)
  pFRate = sum_frames_P_sizes / (burstPeriod - burstDuration- interval_P_frames).
- Given the burstDuration (duration of the I-Frame in a GOP), we calculate the inter-P-Frame durations as follows:
  interval_P_frames = (burstPeriod - burstDuration) / (gopLength -1)
- Then we can organize the Frames in each GOP. An example of typical GOP can be found in the auto generated .dat file. Figure 2 illustrates the structure of a GOP obtained with the default values of attributes.
- Our next step is to transfer the characteristics into a packets generator while respecting the key attributes (e.g. average bit rate and packet size). Thus, we have to calculate the number of packets of each frame and then the interval between two consecutive packets of I-frames and P/B-Frames by sending packets of I-frame with "peakBitRate" and with "pFRate" for packets of frames P/B. In other words, the key parameter "avBitRate" for transmitting each GOP's data must still be verified.

  Therefore, the next time to send a packet is (maxPacketSize / dataRate), where dataRate is either the PeakBitRate or the pFRate according to the frame type.
- Using this calculated intervals, we can schedule sending events for packets of all the frames in each GOP, and we repeat the iteration again and again until the stop time of the application.

[3]We will delete the $m\_$ from the beginning of all variables for readability.

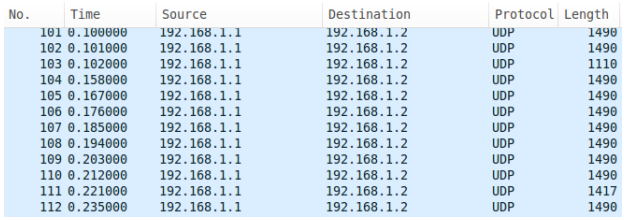
Table 2: MpegPktGenClient attributes.

GopLength	number of frames (I + P or B) per Burst period	15f
ImageRate	number of frames per Second fps created by the Camera	12fps
IframeSize	size of the I-frames in the GOP in kbits	1200kbits
AvBitRate	average Bit rate in mbps	2mpbs
PeakRate	peak Bit rate at which we send the I-Frames in mbps	10mbps
MaxPacketSize	the maximum size of a packet in Bytes without the headers	1468B
VideoFilename	name of the text file that will contain GOP calculated structure	mpeg-4.dat
RemotePort	The port on which the server is listening	-



No.	Time	Source	Destination	Protocol	Length
1	0.000000	192.168.1.1	192.168.1.2	UDP	1490
2	0.001000	192.168.1.1	192.168.1.2	UDP	1490
3	0.002000	192.168.1.1	192.168.1.2	UDP	1490
4	0.003000	192.168.1.1	192.168.1.2	UDP	1490
5	0.004000	192.168.1.1	192.168.1.2	UDP	1490
6	0.005000	192.168.1.1	192.168.1.2	UDP	1490

Figure 3: Received packets of 1st frame (I-Frame) in the GOP.



No.	Time	Source	Destination	Protocol	Length
101	0.100000	192.168.1.1	192.168.1.2	UDP	1490
102	0.101000	192.168.1.1	192.168.1.2	UDP	1490
103	0.102000	192.168.1.1	192.168.1.2	UDP	1110
104	0.158000	192.168.1.1	192.168.1.2	UDP	1490
105	0.167000	192.168.1.1	192.168.1.2	UDP	1490
106	0.176000	192.168.1.1	192.168.1.2	UDP	1490
107	0.185000	192.168.1.1	192.168.1.2	UDP	1490
108	0.194000	192.168.1.1	192.168.1.2	UDP	1490
109	0.203000	192.168.1.1	192.168.1.2	UDP	1490
110	0.212000	192.168.1.1	192.168.1.2	UDP	1490
111	0.221000	192.168.1.1	192.168.1.2	UDP	1417
112	0.235000	192.168.1.1	192.168.1.2	UDP	1490

Figure 4: Received packets of 2ed frame (P-Frame) in the GOP.

2.2.2 Usage and Validation

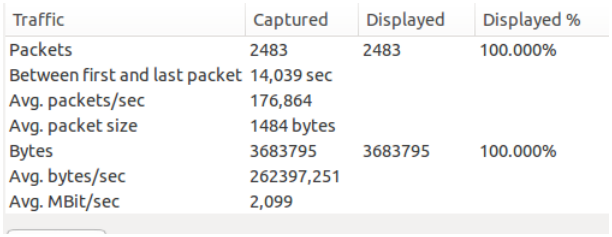
The module usage is extremely simple. The helper will take care of most things. Default values can be changed easily as shown in the code below.

The typical use is to create a UDP server Application that is listing on a particular port, and then to create the MPEG Packet generator client application using the desired values of its attributes as follows:

```
/*Create one udpServer applications on node one to
    receive the video packets from our generator*/
UdpServerHelper server (port);
ApplicationContainer apps = server.Install (n.Get (1));
apps.Start (Seconds (1));
apps.Stop (Seconds (100));
/*Create one MpegPktGenClient application on node
    zero*/
std::string fn = "mpeg.dat";
MpegPktGenClientHelper client (serverAddress, port,"");
client.SetAttribute ("MaxPacketSize", UintegerValue
    (1460));
client.SetAttribute ("VideoFilename", StringValue
    (fn));
client.SetAttribute("GopLength",UintegerValue (15));
client.SetAttribute("ImageRate",UintegerValue (12));
client.SetAttribute("IframeSize",UintegerValue (1200));
client.SetAttribute("AvBitRate",DoubleValue (2.0));
client.SetAttribute("PeakRate",DoubleValue (10.0));
apps = client.Install (n.Get (0));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (100.0));
```

A complete example is located in examples/mpeg-gen-pkt-client.

To validate the application, we present and discuss snapshots from the exchanged packets between the client and the server applications. The presented data are extracted from



Traffic	Captured	Displayed	Displayed %
Packets	2483	2483	100.000%
Between first and last packet	14,039 sec		
Avg. packets/sec	176,864		
Avg. packet size	1484 bytes		
Bytes	3683795	3683795	100.000%
Avg. bytes/sec	262397,251		
Avg. MBit/sec	2,099		

Figure 5: Statistics from the pcap file generated by wireshark.

the pcap captured files. A graphical presentation of the captured data is displayed in Figures 2, 3 and 4, that are drawn using the packet analyzer program "wireshark".

We notice from Figure 2 that we have 15 frames, the first one is the I-frame that lasts for 0.102 s (burst duration) over the 1.25 s of the GOP duration (burst period), where the rest are the P/B frames. The first I-Frame consists of 103 IPv4 (IPv6 is supported) packets with an interval around 0.001 s between them because they are sent at the peak rate (e.g. 10 Mbps) as shown from Figure 4. The other frames consists of 8 IPv4 packets (7 of size 1468 B and one of size 1389 B as shown from packet N104 to N111 in the Figure 4) with inter packet arrival time around 0.009 s because they are sent with pFRate (< avBitRate) as explained in the previous subsection. We remark as well that the first packet of each new frame starts at the corresponding starting time of the given frame as calculated in the GOP structure (the file mpeg.data). Also, from the statistics information obtained from wireshark and depicted in Figure 5, we see that the average bit rate is verified (e.g. 2 Mbps).

2.3 New 3D Antennas Patterns Modules

To run a realistic simulation, it is crucial to model the real radiation patterns of the different types of antennas that are used in the real CBTC implementation. In the real system, two types of antennas are used. A track-side antenna 'Ground antenna' and an on board shark antenna 'OBM antenna'. The first one consists of two back-to-back antennas pointing in the direction tangent to the track and connected by a coupler (this yields to some db loss, usually 3db). The on board shark antenna is physically installed on the rooftop of the train, and screwed onto a horizontal metal ground plane. As described in Section 2.1, for increasing reliability, two OBMs are installed inside a train. Each OBM is attached to an OBM antenna. One antenna is pointing to the movement direction of the train (front antenna) and the second one is pointing at the opposite direction (back antenna). Both antennas should be oriented parallel to the track.

We implemented a new antenna model that imitates the radiation pattern of real antennas. In order to use this module, we need to provide the desired radiation pattern as an input text file as the one provided by the H&S antennas vendor [1].[4] The vendors have usually a tool via their web site that generate the data file for a given antenna model. Usually, the text file consists of the three columns that correspond to the azimuth, elevation angles and the gain values. The angles are used to define the apparent position of an OBM antenna or a Ground antenna in the polar coordinate system used by H&S. The module performs the mapping with the ns-3 coordinate system.

2.4 Handover and Fast Rerouting After it

Enhancing handover. We added a multi-channel scanning mechanism and modified the related objects at Channel, Physical, and MAC level, based on the received signal strength of beacons and the active probe response messages. An OBM will scan for some fixed time the current channel and the x next and previous channels, after switching to a new frequency, where x is typically 2. The OBM will then select the best channel to switch to and the corresponding AP to associate with.[5]

Fast rerouting after handovers. We have modified the static routing module and added some methods to the MAC layer objects to statically and immediately update the routing tables of the nodes affected by the association/disassociation events.

3 Selected Experiment Results

The new modules help in understanding the impact of many factors on the performance of the system. In a real implementation of the CBTC system, the engineers have to adapt different system parameters (e.g. transmission power levels, distance between TREs, change Antenna's model, timer values, ...) when deploying a new line. The cost of these experiments in terms of time and equipment is a main difficulty for the rail transport companies. In particular, the estimation of spurious EBs rate through experimental measurements when important changes are done on a track is very costly, because we want to estimate events that occur at most once every 10^5 hours. An alternative solution is to use a detailed simulation. For illustrative purpose of the use of simulation for engineering the system, we will show visual presentation of selected results of three scenarios. In the first one we will see how to check the radio frequency (RF) model and study the impact of different antennas gain, transmitter/receiver power, the mobility and propagation models. We will show in the second one the behavior of the RF with handovers. We can study the impact of some important factors such as train's length as well as its speed and the distance between TREs. Last, in the third one, we show the exploitation of our modules for the estimation of the spurious EBs with a particular configuration. In fact, our ultimate goal is a fast estimator based on normal wireless network and traffic conditions. However, the simulation approach is not computationally efficient to evaluate a very rare event in normal conditions. Instead, we can evaluate the EB rate using simulation in degraded and critical scenarios (e.g. a high packet loss rate and a large latency due to interference between several trains of different speed and direction, handovers, different inter-TREs distance, different antennas gain, variable transmission power levels, and bursty multimedia traffic). For normal conditions, our approach to evaluate the EB rate is analytical. The robust developed analytical tool is a first step in this direction. We advocate to use ns-3 for validating EB rate estimations under degraded conditions and network

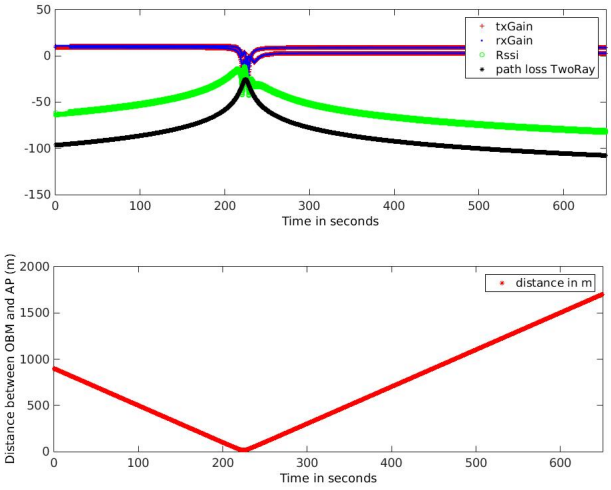


Figure 6: Evolution of RSSI, antenna gains and path loss versus train position around one AP.

[4] One of the largest constructors of antennas for CBTC systems.

[5] Sharing the details and the code of this part is not authorized for confidentiality reasons.

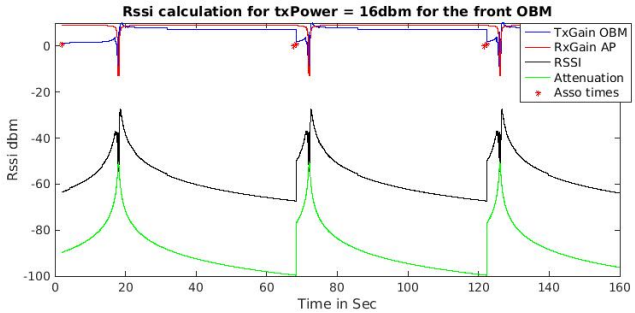


Figure 7: Evolution of RSSI, antenna gains and path loss versus train position with mobility and multible TREs.

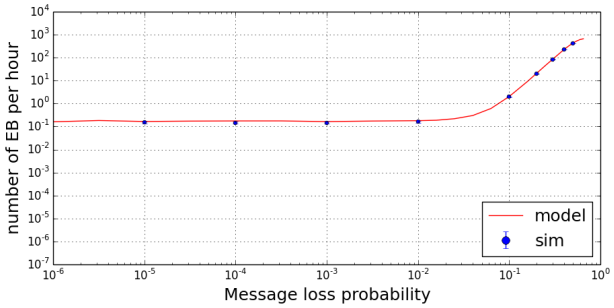


Figure 8: Number of emergency brakes per hour.

provisioning.

First Experiment:

An important parameter to be observed and checked before tracing the CBTC messages is the received signal strength indicator (RSSI). In Figure 6, we depict the evolution of the RSSI with respect to the evolution of the antennas gain and the train position while assuming a single traveling train rightward at constant speed (e.g. 30 km/h) along a track that has only one AP. In fact, the RSSI determines when the amount of radio energy is above the modem sensitivity or the threshold power that must be received to correctly decode the packet (e.g. -79 dbm for data transmission rate of 36 mbps). It is impacted by the transmitter power, the transmission/receiving antenna gain, the orientation of antennas and the propagation model. In Figure 6, the two extreme points at initial and final time correspond to the front or back lobe for OBM antenna and a main front lobe for the Ground antenna model. Note that the former represents physically one single antenna whereas the latter represents two back-to-back antennas pointing in the direction tangent to the track as explained in Section 2.3. Notice that the AP transmits with the max gain and the OBM receives by the front lobe in the left part of the figure (i.e. before crossing the AP at time 250 seconds) and then it receives by the back lobe after crossing the AP. Also, when the train is close to the AP, the gain of both antenna's types degrades. However, when it is too close to the AP, the attenuation is less important and then the rssi remains above the threshold of the handover, consequently the OBM keeps its association with the current AP.

Second Experiment:

Figure 7 reports the evolution of the same metrics as in Figure 6 while considering many TREs with some distance between them (e.g. 400 m). The train will associate with the first AP, then when the RSSI level decreases below some value or after loosing some consecutive number of beacons, it starts a handover process to associate with a new AP after the scanning and the authentication times (e.g. some tens to hundreds milliseconds). Here we can monitor many important factors again as the distance between the TREs in addition to the transmission power, etc. We want to verify that the time to re-associate is close to what we observe in a real implementation. Note that the probability of loss during the handovers is 1. The loss probability before the handover phases depends on the rssi value and other factors, and then we aim at reducing this probability by correctly choosing the handover threshold, the antenna's type, their orientation, the transmission power and the distance between TREs.

Third Experiment:

Figure 8 reports the EB rate with different values of the loss probability out of the handovers periods. In this simple scenario, we fix the loss probability rather than using the calculated value by ns-3 (based on the SNR, etc.) for two reasons: (i) to simulate simply a degraded scenario without consider many trains with congested traffic and interference, and (ii) to present a new use of the simulation for our project; the validation of a mathematical formula that estimates the EB rate. From the figure, we see that for a constant packet loss out of the handovers the analytical results are within the confidence interval of the simulated results.

4 Related Work

Studying train control systems has attracted considerable attention recently such as [7, 3, 2] that considered the abstraction of ETCS level 3 specifications. However, most of the work has been based on high level models using for instance the stochastic Petri Networks without a validation with realistic simulations. Some works developed Monte-Carlo simulation or numerical results of the Petri Networks. In both cases the dependence on the system parameters is hidden (e.g. handovers and track model). The proposed new modules aim at filling this gap.

5 Conclusion and Future Work

The proposed CBTC simulation modules are useful for driving experiments in intelligent transport domain. Some developed modules, i.e. Video generator and directional Antennas, can be used in a wide range of research areas. It is worth mentioning that we enhanced some modules and discovered some bugs (e.g. some mishandled packets in CSMA, and fixed OFDM transmission rates for some control messages (e.g. cts) in the constant rate WiFi manager model).

Using ns-3 with the new modules permits to explicitly consider the impact of many important factors that are the cause of losses or delay and then contribute to triggering EBs. More precisely, we jointly study the impact of trains mobility, the real directional antennas pattern, 802.11e/WMM-style QoS support, and the bursty traffic with the handovers on the performance. This aspect was ignored in the previous literature. Our framework can be used for validating analytical approaches.

Using our ns-3 implementation we could answer the following questions: How to utilize train lines more efficiently? What is the QoS level of the CBTC messages and video traffic perceived by the system in a particular case?

The remaining barrier to enhancing our tool in the future is the consideration of fast fading models and more elaborate track models.

6 Acknowledgments

This work is partially funded by the Inria-Alstom Transport joint lab.

References

[1] HUBER+SUHNER official site. http://www.hubersuhner.com.

[2] L. Carnevali, F. Flammini, M. Paolieri, and E. Vicario. Non-markovian performability evaluation of ERTMS/ETCS level 3. In *Computer Performance Engineering (Proc. of EPEW 2015)*, volume 9272 of *Lecture Notes in Computer Science*, pages 47–62. 2015.

[3] F. Flammini, S. Marrone, M. Iacono, N. Mazzocca, and V. Vittorini. A multiformalism modular approach to ERTMS/ETCS failure modeling. *International Journal of Reliability, Quality and Safety Engineering*, 21(1):1450001 (29 pages), 2014.

[4] ns-3 official site. https://www.nsnam.org.

[5] ns-3 codereview issue of the cbtc module. https://codereview.appspot.com/289110043.

[6] ns-3 codereview issue of the video generator. https://codereview.appspot.com/286160043.

[7] A. Zimmermann and G. Hommel. Towards modeling and evaluation of ETCS real-time communication and operation. *Journal of Systems and Software*, 77(1):47–54, 2005.

A List of Acronyms

The acronyms used in the paper are listed in below.

AP	Access Point
CBTC	Communication Based Train Control
CC	Carborne Controller
DCS	Data Communication Sub-System
EB	Emergency Brake
EOA	End-Of-Authority
ETCS	European Train Control System
LMA	Limit of Movement Authority
LOC	Location report
OBM	On Board Modem
RSSI	Received Signal Strength Indicator
TM	validity duration Timer of a LOC
TRE	Trackside Radio Equipment
ZC	Zone Controller