Tesis de Grado de Ingeniería en Informática

Construcción y Evaluación Experimental de un Sistema de Banco de Pruebas Intravehicular para Ethernet Automotriz

Director: Dr. Esteban Carisimo

Co-director: Dr. Ion Turcanu

Alumna: Sofía Morseletto, *Padrón Nro. 96.107*

Facultad de Ingeniería, Universidad de Buenos Aires

Índice general

Índice de figuras

Índice de listados de código

Agradecimientos

Quiero expresar mi agradecimiento al Dr. Ion Turcanu y al Dr. Ridha Soua, mis supervisores de investigación del laboratorio SECAN-Lab de la Universidad de Luxemburgo por su guía y enseñanza a lo largo de los experimentos realizados en ese establecimiento.

Asimismo, doy las gracias al Profesor Dr. Thomas Engel por permitirme formar parte del equipo de investigación de SECAN-Lab.

También, agradezco profundamente la dirección y el acompañamiento del Dr. Esteban Carisimo quien me brindó su orientación detallada y constante desde el primer día.

Por último, quiero agradecer inmensamente a mis amigos y a mi familia quienes me dieron su apoyo incondicional a lo largo de toda la carrera y de quienes me llevo millones de enseñanzas. Realmente no podría haber llegado a este punto sin ellos.

Abstract

Los avances tecnológicos de la industria automotriz plantean como desafío la integración y comunicación de un gran número de dispositivos electrónicos, tales como pueden ser sensores y cámaras de alta resolución. Dada la existencia de información prioritaria ens una red intravehicular, como pueden ser los datos vinculados a los sistemas de control, se requiere que estos sistemas cuenten con entrega determinística de la información. Actualmente los protocolos que dominan en las redes vehiculares, como Controller Area Network (CAN), Media Oriented Serial Transport (MOST) y FlexRay, no cuentan con la capacidad para soportar las exigencias de banda ancha que son requeridas actualmente. Bajo esta necesidad, surge el concepto de Ethernet Automotriz (EA), la idea de incorporar Ethernet al entorno vehicular, un protocolo conocido por poseer una alta capacidad de banda ancha, un bajo costo y gran flexibilidad. Dentro de Ethernet, predominan dos protocolos: Audio Video Bridging (AVB)/ Time-Sensitive Networking(TSN) y Time-Triggered Ethernet (TTEthernet).

El presente trabajo de Tesis se enmarcará dentro del proyecto llamado: *Building an In-Car Ethernet Testbed System* [1], propuesto por *SECAN-Lab*, laboratorio dependiente de la *Universidad de Luxemburgo*. El objetivo del proyecto es desarrollar un banco de pruebas de Ethernet Automotriz intravehicular. Con tal fin, se plantea llevar a cabo la implementación mediante bibliotecas de código abierto y dispositivos simples y económicos, de manera tal de que el sistema sea fácilmente replicado. En particular, el trabajo de Tesis propuesto tendrá como objetivo la calibración del banco de pruebas, por medio de exigencias realistas de tráfico vehicular que reflejan los próximos avances en este tipo de tecnología.

En primer lugar, se ejecutarán simulaciones en un entorno virtual para prever los experimentos que luego serán llevados a cabo en el banco real. Luego, se realizará una evaluación de la ejecución de AVB/TSN bajo diferentes topologías y con diferentes clases de tráfico que exhiben los requerimientos de la comunicación intravehicular. Asimismo se profundizará la investigación de AVB/TSN realizando experimentos sobre uno de los algoritmos de conformado de tráfico que conforma este protocolo: Credit Based Shaper. Esta profundización tiene como objetivo demostrar que el banco de pruebas también

permite reflejar resultados reales basados en un modelos analíticos matemáticos [2].

Pre-Introducción

Esta tesis se fragmenta en los siguientes capítulos:

1. **Contexto Teórico**: Se describe el contexto actual en redes vehiculares junto con las tecnologías que predominan hoy en día y las motivaciones para migrar a un protocolo unificado basado en Ethernet. Asimismo, se describe en detalle el protocolo Audio Video Bridging/Time-Sensitive Networking y, finalmente, se da una introducción sobre Algoritmos de Selección de Transmisión.

2. **Simulación**: En este capítulo se presentan simulaciones realizadas en una plataforma virtual. Se describen distintos tipos de simuladores y se explica por qué se elige OMNeT++ para experimentar. Se muestra la interacción con el sistema y las limitaciones del mismo.

3. **Banco de Pruebas**: Este capítulo relata el armado y calibración del banco de pruebas de código abierto realizado en esta tesis. Se evalúa el protocolo AVB/TSN en distintas topologías y se analizan los resultados obtenidos de demora, jitter y pérdida de paquetes. Luego, se profundiza la investigación del protocolo centrando el foco en uno de los algoritmos que compone AVB/TSN: Credit Based Shaper.

4. **Conclusiones**: Se detallan las conclusiones obtenidas en este trabajo de investigación.

Capítulo 1

Contexto Teórico

En este capítulo presentaremos las cuestiones teóricas esenciales sobre las cuales se basará el trabajo de esta tesis. En particular, cubriremos el estado del arte actual en cuanto a tecnologías vehiculares, para presentar el contexto en el que se plantea la presente investigación. Luego, se hará hincapié en Ethernet Automotriz (EA), puesto que es la tecnología que ha tomado relevancia en los últimos años. Dentro de lo que es EA, se explicará en detalle el protocolo Audio Video Bridging / Time-Sensitive Networking que cuenta con el mayor apoyo por parte de la comunidad hoy en día. Por último, se dará una introducción sobre Algoritmos de Selección de Transmisión, los cuales son esenciales para proporcionar determinismo a Ethernet.

1.1. Introducción

La comunicación intra-vehicular consiste en la transmisión de información entre distintos dispositivos que componen una red en común. Estos componentes, denominados ECUs (Electronic Control Units, en inglés), son microcontroladores que se encargan de administrar y controlar distintos dominios del vehículo.

Los avances en la industria automotriz y electrónica de los últimos años ha llevado a la proliferación de diferentes componentes electrónicos para los vehículos modernos. Esto trae como consecuencia un incremento significativo en el número de dispositivos que se agregan a las redes vehiculares junto con su complejidad. Asimismo, traen el surgimiento de nuevas tecnologías tales como Advanced Driver Assistance System (ADAS)[3], sistemas de seguridad, vehículos híbridos y eléctricos, comunicación vehículo a vehículo, sistemas de entretenimiento, navegación, entre otros. Estas innovaciones traen demandas más exigentes de ancho de banda, latencia y sincronización de los componentes vehiculares.

La existencia de requerimientos indispensables para la operación de los sistemas de

control del vehículo acarrean distintos requerimientos de Calidad de Servicio. Por consiguiente, las redes vehiculares han sido desarrolladas a lo largo de los años como un conjunto de sistemas heterogéneos, uniendo diferentes tecnologías. A medida que nuevos requisitos son introducidos a la tecnología vehicular, no todos los dispositivos logran satisfacerlos.

1.2. Comunicación Intra-vehicular

Existe una gran variedad el protocolos y tecnologías de comunicación de datos a bordo de los vehículos. En esta sección se presentarán las tecnologías más frecuentemente usadas por la industria.

En el comienzo, en la comunicación vehicular predominaban los dispositivos de extremo a extremo. Esto requería que el interconexionado dispusiera de un enlace físico por cada par de ECUs que se buscaba interconectar. Por lo tanto, al requerir de un medio físico por cada interconexión, este sistema tendría $O(N^2)$ $((N-1)*N/2)$ en caso de necesitar interconexión entre todas sus partes. Visto el aumento del número de dispositivos en los vehículos, sumado al espacio requerido para trazar el cableado, los sistemas de comunicación vehicular migraron hacia otras alternativas

1.2.1. Tecnologías Preexistentes

El siguiente paso fue avanzar hacia una comunicación multiplexada. En este nuevo paradigma surgen los siguientes protocolos:

1. Controller Area Network (CAN): Es un estándar basado en un bus de comunicación para transmitir información de control entre los distintos ECUs de un vehículo. A pesar de su bajo ancho de banda, su robustez, flexibilidad y bajo costo le brinda a CAN la popularidad con la que cuenta en el presente. Sin embargo, cada transmisión realizada mediante CAN puede demorar un tiempo indeterminado causando que mensajes de alta criticidad no lleguen a tiempo. De esta forma, no es un protocolo ideal para comunicación de seguridad crítica.

2. Media Oriented Systems Transport (MOST): Su función principal es soportar la transmisión de tráfico de tipo audio y video dentro de un vehículo. Justamente al ser diseñado para cumplir este propósito, no cuenta con las características necesarias para soportar adecuadamente tráfico de criticidad alta. Debido a esto, no es posible utilizarlo como el protocolo principal de una red vehicular sino como uno dedicado a un sector enfocado a la transmisión multimedia.

3. Local Interconnect Network (LIN): Sistema de comunicación basado en un bus de difusión y de tipo maestro-esclavo. Se utiliza principalmente para aplicaciones poco críticas, como pueden ser el ajuste del espejo o la activación de las cerraduras de las puertas. Dado el limitado ancho de banda y al ser propenso a colisiones, este protocolo no es apropiado para mensajes críticos o tiempo sensibles.

4. FlexRay: se basa en un sistema de dos canales en el que uno funciona de reserva si el primero falla. Posee un esquema Time Division Multiple Access (TDMA) activado por tiempo que aporta determinismo. Su tamaño máximo de payload es bajo (254B), lo que puede ser un limitante para la transmisión de archivos multimedia.

Los ECUs pueden agruparse en distintos tipos según la función que cumplen. Por un lado se encuentran los encargados de manejar sistemas relacionados con la seguridad del vehículo (trabas de las puertas, estabilidad, frenado) y, por el otro, los responsables de recolectar e intercambiar información (sistema de entretenimiento, asistencia al estacionar, diagnostico del vehículo).

Para poder transmitir eficientemente la información entre los diferentes dispositivos, es necesario contar con un sistema de bus robusto que cumpla garantías de tiempo real, alto rendimiento y baja latencia. Estas exigencias raramente pueden ser cumplidas por una sola tecnología, por lo que al manufacturar un vehículo se terminan utilizando diferentes medios de comunicación para distintos dominios. Por ejemplo, puede utilizarse CAN y LIN para enviar mensajes de alta criticidad, y MOST para información orientada al entretenimiento en donde se requiere un alto ancho de banda.

Puesto que la comunicación intra-vehicular es un campo en crecimiento, cada vez más tecnologías se incorporan y se construyen dentro del vehículo, aumentando la complejidad de la red de comunicación. Coordinar y sincronizar el intercambio de información entre el gran número de ECUs que conviven en el vehiculo y manejan diferentes tecnologías agregan un desafío adicional. Asimismo, permanecer con esta infrastructura no sólo brinda conflictos relacionados con la complejidad del cableado, sino también aumenta el peso del vehículo, lo que puede llegar a disminuir el desempeño del mismo.

1.2.2. Ethernet Automotriz

Aunque la solución que integra distintas tecnologías mediante ECUs es robusta y económica, la gran heterogeneidad que presenta termina ocasionando una complejidad que perjudica el desempeño de la comunicación intra-vehicular. En este contexto surge Ethernet como una opción para unificar las necesidades y comportamientos que prevalecen en un vehiculo bajo un sólo protocolo.

En los últimos años, Ethernet ha progresado desde un punto de partida en el cual era considerado incompatible para redes vehiculares debido a su Interferencia Electromagnética (EMI) / Compatibilidad Electromagnética (EMC)[4] a ser una potencial solución para el futuro de las redes intra-vehiculares. Este avance ha sido posible gracias al trabajo de varios grupos de investigación. Los principales aportes surgen, por un lado, de Broadcom desarrollando BroadReach[5], una tecnología que habilita la utilización de Ethernet en un cable de tipo Unshielded Twisted Pair (UTP) a 100 Mbit/s. Y, por otro lado, de parte del Instituto de Ingeniería Electrica y Electrónica (IEEE) con sus trabajos de estandarización y la creación del protocolo Audio Video Bridging (AVB)/Time-Sensitive Networking (TSN)[6], el cual será analizado en detalle posteriormente en la sección 1.3.

Ethernet Automotriz provee el ancho de banda necesario para las aplicaciones y requisitos de la actualidad. Una de las ventajas principales es que por varios años ha sido la tecnología estándar para Local Area Networks (LANs). Esto implica que una gran cantidad de métodos de transmisión y protocolos, confiables, seguros y eficientes, se ejecutan sobre Ethernet. De esta manera, al incorporar Ethernet a las redes vehiculares, estos protocolos también son habilitados.

Para poder ser adoptado al entorno vehicular, Ethernet debe cumplir con los requisitos de ancho de banda, latencia, sincronización que demanda el mercado automotriz y que hoy en día no brinda por sí solo. Con este propósito, se han propuesto especificaciones y protocolos, de los cuales los tres más prominentes son los estándares: AVB/TSN, Time-Triggered Ethernet (TTEthernet)[7] y 1-Gbps Ethernet.

A continuación se resumen los primeros dos y en la sección siguiente se elabora sobre AVB/TSN que es el estándar que se exploró en el presente trabajo.

1.2.2.1. 1-Gbps Ethernet

Para poder satisfacer las demandas actuales del entorno automotriz surge 1000BASE-T1, también llamado Gigabit Ethernet o 1-Gbps Ethernet, el cual cumple con los requerimientos EMC para sistemas automotrices y, además, soporta transmisión de video no comprimido de alta definición y alto throughput.

Estas dos características son esenciales para los sistemas ADAS que basan su funcionamiento en el video y audio que obtienen del exterior. Asimismo, brindan a Ethernet la capacidad de procesar el alto volumen de información generada por los sensores que se encuentran en los autos autónomos de la actualidad.

1.2.2.2. Time-Triggered Ethernet

Time-Triggered Ethernet [4], también llamado TTEthernet, fue implementado con el propósito de habilitar la coexistencia de comunicación de tiempo real sincronizada y activada por tiempo con mensajes de prioridad más baja activados por eventos. Esto se logra aplicando un esquema multiplexado de división por tiempo.

Zeng *et al.* [8] indica que este estándar no soporta la reserva online de recursos (como sí lo hace AVB/TSN, como se explica en la sección 1.3.1.3), pero regula la transmisión de datos mediante una tabla establecida previamente.

TTEthernet maneja tres tipos de tráfico con diferentes prioridades:

1. TT (Activado por Tiempo o *Time-Triggered*, en inglés): es el de prioridad máxima. Se asignan intervalos de tiempo de transmisión exclusivos para este tipo de tráfico y, por lo tanto, presenta latencia constante y poco jitter.

2. RC (Tasa Restringida o *Rate Constrained*, en inglés): tráfico que se activa mediante eventos, posee una latencia garantizada para un ancho de banda específico.

3. BE (Mejor Esfuerzo): sigue el Ethernet estándar y no presenta garantías de Calidad de Servicio.

La principal ventaja de este estándar es que los switches que son aptos para TTEthernet poseen *preemption* que consiste en la interrupción de la transmisión de mensajes de menor prioridad para dar lugar a mensajes de prioridad más alta. Esto elimina el problema de Head of Line Blocking (HoLB), un problema que sigue presente en el estándar AVB/TSN.

Aunque TTEthernet no garantiza un valor de demora máxima de extremo a extremo, gracias al esquema offline en donde planifica las transmisiones es posible determinar manualmente el peor caso de demora.

Sin embargo, posee la desventaja de ser una tecnología industrial y privada lo que no permite evaluarla en un entorno físico libremente.

Si bien tanto TTEthernet como AVB/TSN son consideradas las dos tecnologías más aptas para entornos intra-vehiculares, existe una preferencia por la segunda por parte de la comunidad. Varias empresas automotrices forman parte de la agrupación Avnu Alliance [9], conglomerado de firmas automotrices cuyo objetivo consta en establecer y certificar AVB/TSN. Además, posee el apoyo del Instituto de Ingeniería Eléctrica y Electrónica (IEEE), dado que el grupo desarrollador de AVB/TSN forma parte de esta organización [10].

1.3. Audio Video Bridging/Time Sensitive Networking

En esta tesis nos enfocaremos en el protocolo AVB/TSN dado el protagonismo que posee en el ambiente de redes vehiculares y al gran apoyo que cuenta de la comunidad.

El protocolo AVB fue desarrollado por el grupo de investigación *Audio Video Bridging task group* perteneciente a IEEE. En 2012, este grupo fue renombrado a *Time-Sensitive Networking task group* para reflejar los avances que habían realizado sobre el protocolo que incluye el desarrollo de un estándar que incorpora mejoras: Time-Sensitive Networking (TSN). A partir de este momento, AVB también pasó a ser renombrado como AVB/TSN.

Este trabajo de investigación gira entorno a la exploración del comportamiento de AVB bajo distintas implementaciones físicas y virtuales, a continuación se describe este protocolo y se mencionan también los avances que luego incorpora TSN.

AVB

Audio Video Bridging (AVB) es un protocolo basado en Ethernet por el grupo de trabajo Audio Video Bridging de IEEE perteneciente al comité de los estándares IEEE 802.1. Su principal objetivo es agregar a Ethernet el soporte necesario para lidiar con audio, video y aplicaciones de control en tiempo real.

Está compuesto por tres estándares:

1. IEEE 802.1AS: establece coordinación y sincronización entre los nodos de la red
2. IEEE 802.1Qat: garantiza la reserva online de recursos
3. IEEE 802.1Qav: brinda mejoras en el envío y encolado de mensajes con respecto a Ethernet
4. IEEE 802.1BA: se enfoca en los sistemas de envío de audio y video

TSN

Time Sensitive Networking (TSN) es una evolución de AVB, basándose en las especificaciones del protocolo pero expandiendo el rango, funcionalidad y aplicaciones del mismo. Incluye nuevas características como *preemption*, más opciones de conformado de tráfico y mejoras en cuanto a control del tráfico y latencia.

Entre los estándares que forman TSN se incluyen:

1. IEEE Std 802.1Qbu-2016: agrega preemption al protocolo. Esto implica la interrupción de la transmisión de tramas de prioridad no crítica para abrir el flujo de tramas de mayor prioridad.

2. IEEE Std 802.1Qbv-2015: especifica un algoritmo de conformado de tráfico para planificar el envío de tramas según un esquema de tiempo

3. IEEE Std 802.1Qca-2015: incorpora extensiones al protocolo de ruteo IS-IS [11] para configurar múltiples caminos en las redes

1.3.1. Estándares de AVB

AVB cuentra con dos tipos de clase de tráfico con diferentes garantías de demora de extremo a extremo: Clase A, con una garantía de 2 ms y Clase B, con 50 ms.

Cada estándar que compone el protocolo brinda las funcionalidades necesarias para cumplir con la cota de demora que asegura el mismo, agregando Calidad de Servicio y determinismo a Ethernet.

1.3.1.1. IEEE 802.1AS: generalized Precision Time Protocol (gPTP)

Los relojes presentes en cada dispositivo funcionan de manera autónoma, cada reloj tiene su propia deriva de frecuencia y offset. Para que el protocolo pueda asegurar determinisimo en cuanto a la demora del tráfico, es necesario que todos los miembros de la red puedan sincronizarse en conjunto.

Esto requiere la presencia de nodos *time-aware* (tiempo-concientes, en español), nodos que seleccionan una fuente de reloj común en una red y se sincronizan constantemente con la misma. Es decir, se necesitan nodos que sean capaces de actuar tanto como fuentes o como destino de mensajes de sincronización. Asimismo, los puentes de la red deben poder sincronizarse y transmitir este tipo de mensajes.

Ante estos requisitos surge el estándar IEEE 802.1AS: generalized Precision Time Protocol (gPTP), el cual se encarga de crear y distribuir mensajes en la red para sincronizar sus miembros bajo un sólo reloj maestro.

Para cumplir este objetivo el estándar trabaja siguiendo tres conceptos:

Dominio gPTP

Mediante el envío de mensajes a través de la red, se detectan los dispositivos que poseen las características de hardware necesarias para cumplir con los requisitos de sincronización buscados. Si un dispositivo no cumple con estos requerimientos, se lo remueve

del alcance de la comunicación. Los miembros que cumplen se marcan como aptos conformando, así, un subconjunto de nodos llamado dominio gPTP.

Este dominio está compuesto, como se mencionó previamente de:

1. Nodos *time-aware*: dispositivos transmisores y receptores de mensajes de sincronización

2. Puentes *time-aware*: enlaces que reciben información de sincronización, corrigen y compensan posibles demoras de propagación, configuran sus relojes propios y retransmiten la información.

Medición de la demora de propagación

La demora de propagación es considerada como el intervalo de tiempo en el cual un bit se propaga desde su fuente a su destino. La forma de calcular esta demora es la misma que en el estándar IEEE Std 1588-2008 Precision Time Protocol (PTP)[12], la cual consiste en realizar un intercambio de tres mensajes cada dos nodos:

1. `PDelay_Req`

2. `PDelay_Resp`

3. `PDelay_Follow_up`

El nodo que inicia el intercambio envía el mensaje `PDelay_Req`, el receptor lo toma y lo responde enviando el `PDelay_Resp`, a continuación envía también un último mensaje `PDelay_Follow_up` que indica el timestamp en el que se envío el mensaje anterior. Con este intercambio, los nodos logran encontrar los desfasajes de sincronización que pueden surgir debido a la demora de propagación y, por consiguiente, minimizar el error.

Algoritmo del Mejor Reloj Maestro

Este algoritmo, llamado *Best Master Clock Algorithm* en inglés, determina que la medición del reloj interno de un dispositivo puede ser calculada en términos de un reloj externo: el reloj Maestro.

Mediante intercambio de mensajes entre nodos, se determina un único reloj Maestro a partir del cual los demás van a sincronizarse. Cada dispositivo perteneciente a la red envía un mensaje con las características de su reloj interno. Al mismo tiempo, recibe la información de los demás, la cual utiliza para comparar propiedades y seleccionar el reloj que considera óptimo para convertirse en el reloj Maestro. La decisión sobre qué reloj seguir es unánime, todos los nodos eligen al mismo reloj Maestro.

Con estos tres pasos, el estándar gPTP logra establecer un reloj común para toda la red acercándole a Ethernet el determinismo que requiere para poder desempeñarse y cumplir con las exigencias del entorno intra-vehicular.

1.3.1.2. IEEE 802.1Qav: Forwarding Queue in Time Sensitive System (FQTSS)

Una vez que los nodos de la red se sincronizaron, puede comenzar la transmisión. Para que las tramas transmitidas arriben a destino cumpliendo con las cotas que garantiza el protocolo, AVB/TSN sigue los principios del estándar IEEE 802.1Qav.

FQTSS define un sistema de conformado de tráfico utilizando prioridades establecidas en las dos clases que maneja el protocolo: clase A y clase B. Para esto utiliza el algoritmo de selección de transmisión Credit Based Shaper, que será descrito más adelante en la sección 1.4.

1.3.1.3. IEEE 802.1Qat: Stream Reservation Protocol (SRP)

El estándar SRP agrega mejoras a Ethernet implementando control de admisión. Esta es la comprobación previa a realizar una conexión para asegurarse que el dispositivo con el que el Talker se comunicará posee los recursos necesarios para realizar una transmisión. Además, incorpora el concepto de *streaming* a la capa de enlace y posee un mecanismo de administración de recursos de un extremo a otro para asegurar Calidad de Servicio (CdS).

SRP registra un stream y reserva los recursos que se requieren para cumplir con la demora y el ancho de banda que especifica cada clase de tráfico (A o B), mediante el envío de mensajes entre los dispositivos de la red.

Para lograr esto, SRP utiliza tres protocolos de señalización:

1. **Multiple MAC Registration Protocol (MMRP) [13]**: controla la propagación de mensajes para el registro de *Talkers* en la red

2. **Multiple VLAN Registration Protocol (MVRP) [14]**: utilizado por los puentes y dispositivos de la red para señalizar su pertenencia a la VLAN desde donde se origina el stream de datos

3. **Multiple Stream Registration Protocol (MSRP) [15]**: permite a los miembros de la red reservar recursos que puedan garantizar la transmisión de flujos de datos con la CdS requerida

Entrando más en detalle sobre MSRP, en este estándar se etiqueta a cada nodo de la red como *Talker* o *Listener* dependiendo si son el origen o el receptor de la información

a transmitirse. Entre ellos se envían mensajes que incluyen los requerimientos de CdS que posee cada dispositivo y existen de dos tipos:

1. *Talker Advertise*: especifica los streams que van a ser transmitidos desde el dispositivo junto con sus requerimientos. A partir de estos mensajes, los puentes en la red reconocen el ancho de banda demandado y replican los mensajes a otros nodos siempre y cuando estos puentes puedan cumplir los requisitos. Si este no es el caso, el puente contesta con un mensaje de tipo *Talker Failed* [16]

2. *Listener Advertise*: estos mensajes surgen desde los potenciales receptores del stream. Los mismos devuelven mensajes de tipo *Listener Ready* o *Listener Failed* dependiendo de si poseen los recursos necesarios y la intención de recibir el stream o no.

A partir de este intercambio de información, se termina estableciendo un dominio SRP con todos los dispositivos y puentes aptos para realizar una comunicación que cumple con las demandas de demora y ancho de banda de AVB/TSN.

1.3.1.4. IEEE 802.1BA: Audio Video Bridging

El propósito principal de este estándar es la determinación de un dominio AVB compuesto exclusivamente por dispositivos aptos para cumplir las demandas de demora y sincronización que demanda el protocolo.

En este dominio, cada nodo puede actuar como *Talker*, la fuente desde donde se origina el flujo de datos, o como *Listener*, el destino de la información. El tráfico debe originarse de un *Talker*, pero puede ser recibido por varios *Listeners*.

Las condiciones para ser parte del dominio AVB son:

1. Ejercer el estándar IEEE 802.1Qav: Forwarding Queue in Time Sensitive System (FQTSS)

2. Permitir el registro, declaración y propagación de los atributos MSRP y MVRP

3. Soporte para Generalized Precision Time Protocol (gPTP)

4. Categorización del tráfico en clase A y clase B

De esta manera, para que un dispositivo pueda formar parte del dominio AVB debe cumplir con los estándares que conllevan el protocolo.

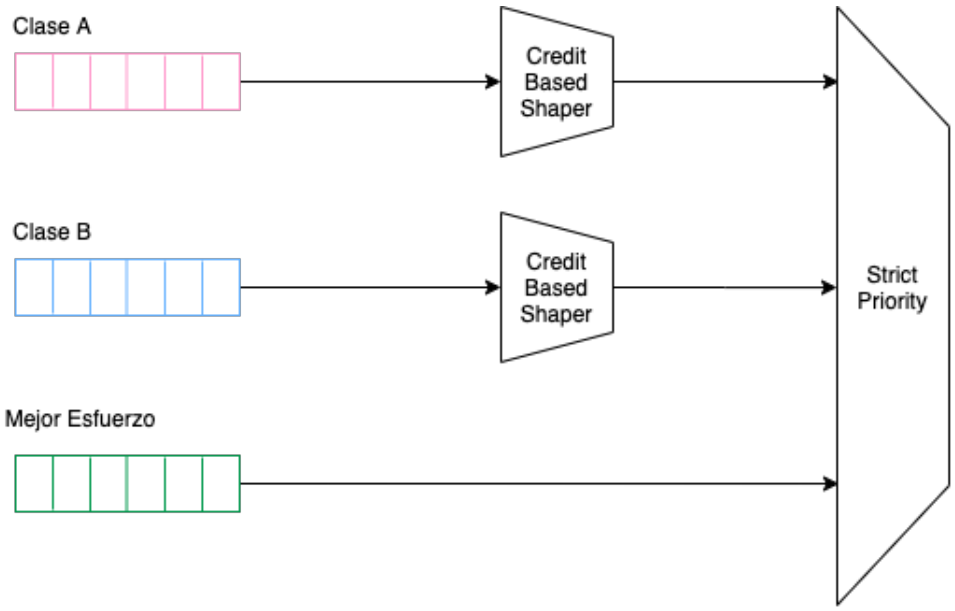


Figura 1.1: Diagrama de las colas de tráfico en un puerto de un switch AVB. CBS trabaja con las colas de clase A y B, mientras que el tráfico de mejor esfuerzo es conformado únicamente por el algoritmo de Strict Priority.

1.4. Algoritmos de Selección de Transmisión

Para concluir este capítulo, presentaremos algunos de los algoritmos de transmisión que existen hoy en día.

En cada puerto de los dispositivos que conforman la red, se aplican algoritmos de selección de transmisión para determinar el orden en el que las tramas van a ingresar o egresar el puerto. Estos algoritmos permiten diferenciar el tráfico según su clase y priorizar el de mayor criticidad. Gracias a esto, Audio Video Bridging (AVB)/Time-Sensitive Networking (TSN) logra cumplir las garantías de demora que le agregan a Ethernet el determinismo necesario para funcionar eficientemente en redes vehiculares.

El egreso de paquetes en switches y dispositivos que operan el protocolo AVB está determinado a través de dos algoritmos avocados a la selección de transmisión. Uno basa su orden de transmisión en la prioridad que tiene asignada cada paquete, transmitiendo primero los de mayor prioridad. El otro regula las transmisiones según un sistema de créditos, mediante el cual una clase sólo puede transmitir si posee una cantidad positiva de créditos. Estos son:

1. Strict Priority

2. Credit Based Shaper

El diagrama de la figura 1.1 refleja las tres colas en las que se divide el tráfico en AVB/TSN . Sólo clase A y B, las clases de mayor prioridad, reciben el conformado de

tráfico provisto por CBS. Por otro lado, el algoritmo de Strict Priority regula tanto el tráfico de clase A y B como el de mejor esfuerzo.

1.4.1. Strict Priority

Controla la transmisión de los flujos según la prioridad que le fue asignado a cada uno. Existen ocho prioridades disponibles y cada una tiene una cola diferente en donde acumula el tráfico correspondiente.

Se basa en un campo de tres bits contenido en la trama, llamado Priority Code Point (PCP) [17], que contiene el nivel de prioridad de la trama. Este campo se origina en el estándar IEEE 802.1Q, el cual agrega 32 bits a la trama de Ethernet para poder garantizar calidad de servicio a nivel de la capa de enlace. Estos bits se dividen en:

1. 16 bits para identificar el protocolo. Se utiliza el valor 0x8100.
2. 3 bits para PCP
3. 1 bit para indicar si la trama puede ser descartada en caso de congestión
4. 12 últimos bits para indicar el VLAN ID

El campo PCP, por lo tanto, permite diferenciar el tráfico según su prioridad a nivel de la capa de enlace.

Para ubicar cada trama en su cola se establece una relación entre el número de cola y la clase de tráfico. De esta forma, si existen solamente tres clases de tráfico pero ocho colas se asignan varias colas a la misma clase. Esto permite que Strict Priority pueda diferenciar entre clase A, clase B y mejor esfuerzo en AVB/TSN a partir del valor PCP.

Para que una trama esté disponible para ser transmitida basta con que se encuentre en su cola correspondiente. Para elegir la próxima trama a transmitir sigue un enfoque voraz (greedy, en inglés) mediante el cual transmite en primer lugar las tramas con el número mayor de prioridad.

Es el algoritmo de transmisión por defecto y debe estar implementado en cada puerto de la red.

1.4.2. Credit Based Shaper

El Credit Based Shaper (CBS) es un algoritmo de selección de transmisión de Espera Equitativa Ponderada (Weighted Fair Queuing, en inglés) basado en un sistema de créditos. Forma parte del estándar IEEE 802.1Qav [18], contenido en el protocolo AVB/TSN.

Las prioridades que regula el algoritmo, en el contexto de AVB/TSN, son tres: clase A, clase B y mejor esfuerzo. El algoritmo asigna una cantidad inicial de créditos a cada clase, la cual disminuye mientras una trama de la misma clase está siendo transmitida (pudiendo tomar valores menores a 0). Cuando la clase no puede transmitir, ya sea porque tramas de otro flujo están siendo transmitidas o porque la cantidad de créditos es negativa, su valor de créditos comienza a incrementar gradualmente. La velocidad en la que los créditos aumentan depende de una pendiente llamada *pendiente de inactividad (idle slope)* y la velocidad con la que disminuyen depende de la *pendiente de envío (send slope)*.

Como menciona Cao *et al.* [19] una trama puede ser transmitida si:

1. No hay tramas pendientes de transmisión de prioridades más altas
2. No hay tramas pendientes de transmisión previas en la misma cola
3. Ninguna trama está siendo transmitida en ese momento

A estas propiedades, CBS agrega que una transmisión solamente puede comenzar si:

1. Su clase de tráfico no posee créditos negativos

En la figura 1.2 se muestra un flujo de transmisión en el cual se refleja el trabajo de conformado de tráfico que realiza CBS. Los pasos son los siguientes:

1. Arriban dos tramas de distintas clase al puerto
2. Se comienza con la transmisión de la trama de mayor prioridad: clase A
3. Durante la transmisión de la misma, los créditos que posee esta clase disminuyen según la pendiente de envió de la misma (α^-)
4. Por otro lado, mientras la trama de clase B aguarda su transmisión su cantidad de créditos aumenta siguiendo la pendiente de inactividad (α^+)
5. Una vez finalizada la trama de clase A, se comienza con la transmisión de clase B. La situación es la inversa a la anterior, los créditos de clase A incrementan mientras que los de clase B disminuyen.
6. Una vez que los créditos alcanzan el valor 0 se mantienen en ese valor hasta que arribe una nueva trama
7. De la misma manera, si al finalizar la transmisión no quedan más tramas por transmitir, la cantidad de créditos baja a 0

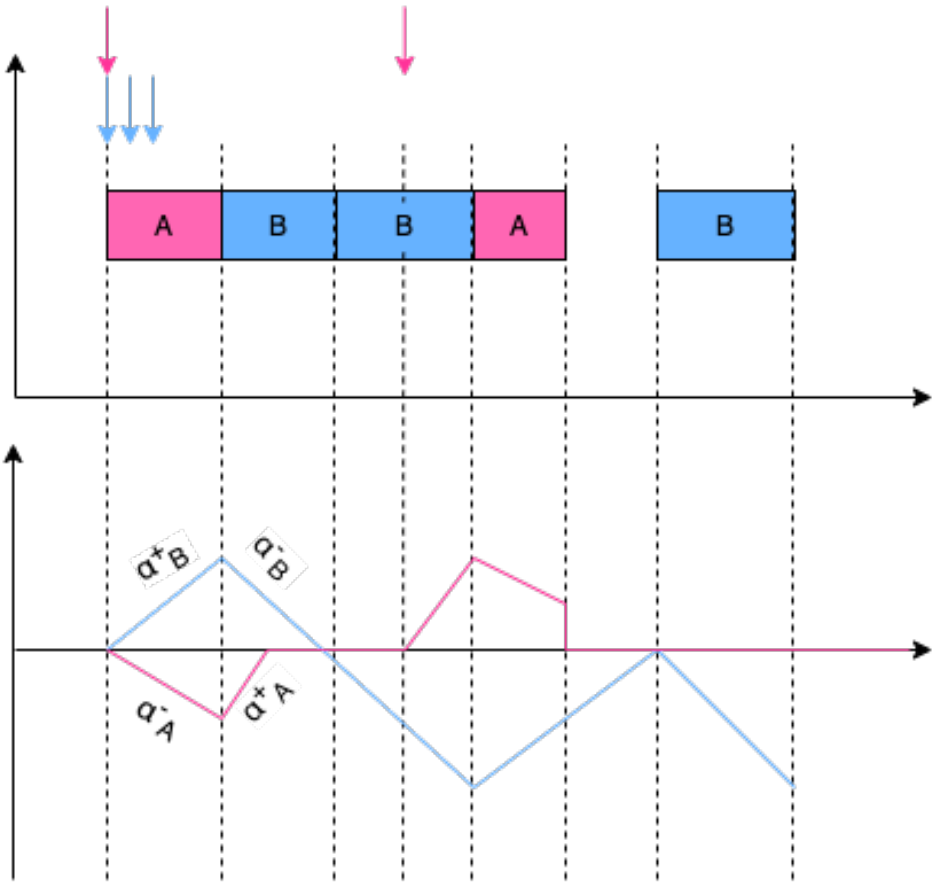


Figura 1.2: Ejemplo de conformado de tráfico realizado por CBS. Mientras una trama es transmitida su cantidad de créditos disminuye según su pendiente de envío (α^-) y mientras aguarda a ser transmitida crece según su pendiente de inactividad (α^+). Una trama es transmitida si posee una cantidad positiva de créditos y si no hay una trama de mayor prioridad pendiente.

8. Por último, una trama no puede comenzar su transmisión hasta que su cantidad de créditos sea mayor o igual a 0, como sucede con la tercera trama de clase B

El CBS, a diferencia de Strict Priority, no sigue un enfoque voraz si no que controla el orden de transmisión según la cantidad de créditos que posee cada flujo de datos a transmitir. Mediante este enfoque, bloquea la transmisión de un cierto flujo de datos una vez que este haya agotado su cantidad de créditos y comienza la transmisión de otro flujo con créditos disponible. De esta manera, garantiza que el tráfico de menor prioridad sea eventualmente transmitido y no sufra de *inanición*.

Impacto de Credit Based Shaper en AVB/TSN

En AVB/TSN la transmisión de información se divide en distintos flujos de datos. Mediante el Stream Reservation Protocol (SRP) se reservan los recursos necesarios para transmitir cada flujo en los dispositivos que forman parte de la red. Esto se realiza mediante un sistema de registro de flujos que permite asegurar que cada uno cuenta con los recursos que necesita para transmitir. Garantizando, así, los límites de demora y ancho de banda que avala AVB/TSN.

En este contexto aparece un parámetro de especificación de tráfico llamado T-Spec (Traffic Specification) que caracteriza la cantidad de ancho de banda que puede consumir un flujo determinado. Para cada flujo, T-Spec determina:

1. El tamaño máximo de trama (MaxFrameSize)

2. La cantidad de tramas a enviar por intervalo de medida para una clase SR (clase A o B) (MaxIntervalFrame)

En cuanto al segundo parámetro, las clases SR son las de mayor prioridad que mejor esfuerzo, es decir, clase A y B. Cada una posee un intervalo específico de medida, es decir el máximo período de transmisión, distinto:

1. Clase A: 125 µs

2. Clase B: 250 µs

Mediante estos parámetros, T-Spec determina un ancho de banda y un límite de demora específicos para cada clase. Credit Based Shaper, entonces, es agregado al protocolo para poder cumplir con los valores asignados a cada clase. La sincronización que realiza CBS previene el agrupamiento de tramas evitando, de esta forma, la sobrecarga de los recursos de la red. Al asegurar un flujo de tráfico uniforme, reduce la posibilidad de que haya pérdida de paquetes debido a congestión en la red.

Por consiguiente, este algoritmo toma un rol protagonista en el cumplimiento de las garantías del protocolo AVB/TSN que no podrían ser satisfechas solamente por el algoritmo de Strict Priority.

Capítulo 2

Simulación

En este capítulo se presentarán las simulaciones realizadas en la plataforma de simulación OMNeT++. El objetivo de las simulaciones fue interiorizarse con los experimentos que luego serían llevados a cabo en el banco real. Estos experimentos contaron con tres topologías que se armarían más adelante en el banco de pruebas físico: topología Estrella, topología Cadena y topología Anillo. Se ejecutaron distintas simulaciones para familiarizarnos con el entorno y encontrar las distintas herramientas y soluciones que el mismo dispone como, también sus limitaciones. Armar pruebas con tres distintas configuraciones generaba desafíos distintos a cada etapa y permitía un mejor conocimiento y descubrimiento del simulador. Además, las simulaciones mostraron cómo los nodos se comunican entre ellos para formar el dominio AVB. El entorno virtual permitió obtener un detallado seguimiento de los mensajes que se transmitieron en el mismo momento de la transmisión, brindando información sobre el origen y el destino del mensaje, el contenido del mismo y en qué segmento de la red se encontraba en un instante determinado mediante registros. Asimismo, el simulador nos otorgó la capacidad de poder poner a prueba la factibilidad de las topologías. Realizar los experimentos de manera virtual nos permitía tener una primera confirmación de que las configuraciones a probar eran posibles para el protocolo y podrían, entonces replicarse en un entorno real.

2.1. Motivación

Uno de los objetivos principales por los cuales decidimos realizar los siguientes experimentos en forma de simulaciones consistió en poder entender con mayor detalle el entorno con el que se iba a trabajar posteriormente. Fue posible visualizar con detalle los mensajes ***Talker Advertise*** y ***Listener Ready*** del protocolo Stream Reservation Protocol (SRP) [20] que se envían entre los distintos nodos y switches para establecer el dominio AVB. Asimismo, parámetros que luego no resultaron posibles capturar en

el entorno físico, sí pudieron ser observados en el ambiente virtual, como es el caso de la variación de créditos del Credit Based Shaper (CBS) en cada puerto que ejecutara este protocolo. Dado que en el estado del arte, la mayoría de los experimentos sobre este protocolo fueron hechos mediante simulaciones, realizar las mismas da perspectiva de las investigaciones que se realizaron contemporáneamente y cómo estas explotaron el protocolo en un entorno virtual. También el simulador brinda el beneficio de poder contar con más dispositivos para agregar a la red que, si bien son virtuales, permiten modelar una red más extensa que la que es posible en un contexto real con un número limitado de instrumentos disponibles. Por último, simular los experimentos que iban a realizarse posteriormente provocó tener que lidiar con situaciones conflictivas *a priori*, lo que derivó en aprender a solucionarlas cuando sucedieran en el entorno físico de manera más sencilla.

2.2. Entorno

Para realizar simulaciones de red que utilicen el protocolo AVB/TSN fue fundamental contar con un entorno de simulación que permitiera la creación y control de nodos que trasmitieran información de distinto tipo. Además, se aplicaron frameworks (marcos de trabajo) que permitieran que los módulos creados representen componentes de una red vehicular. Para modelar los componentes vehiculares requeridos, se necesitaba, por un lado, que los simuladores dispusieran los protocolos de red en los que se basa AVB/TSN, como lo es Ethernet, y, de la misma manera, todos los protocolos que forman AVB/TSN: IEEE 802.1AS, IEEE 802.1Qat, IEEE 802.1Qav y IEEE 802.1BA. Por estas razones, se decidió utilizar el simulador OMNeT++ [21] y el framework de simulación de redes alámbricas, inálambricas y móviles INET. Asimismo, se agregó la extensión de INET, Communication over Realtime Ethernet 4 INET (CoRE4INET) , que provee implementaciones para protocolos basados en Ethernet, como lo son AVB/TSN y Time-Triggered Ethernet (TTEthernet).

2.2.1. Simuladores

Un simulador de red es una pieza de software que actúa como una red de comunicación. Con el avance del tiempo, las redes se volvieron demasiado complejas para que métodos analíticos tradicionales puedan proveer un entendimiento del comportamiento del sistema. Provocando, así, que los simuladores de red comiencen a tomar relevancia. De los simuladores disponibles hoy en día, tres posibles candidatos para realizar las simulaciones deseadas en este trabajo fueron: Graphical Network Simulator-3 (GNS3) [22],

Mininet y OMNeT++.

2.2.2. GNS3

Uno de los simuladores más populares utilizados en la actualidad, y frecuentemente usado para estudiar para exámenes de certificación profesional como pueden ser específicamente los de la empresa CISCO.
Es de código abierto y está implementado en lenguaje Python. Posee una comunidad en línea formada y extensa de más de 800.000 miembros. Fue uno de los primeros simuladores de redes en aparecer, permitiendo construir redes grandes y complejas de manera virtual para realizar pruebas o experimentos sin necesidad de comprar dispositivos físicos. De los simuladores evaluados para la realización de esta tesis, fue el de más antigüedad, habiendo salido oficialmente en el año 2007.
Sin embargo, posee configuraciones y protocolos orientados a necesidades enfocadas a experimentar con hardware comercial de manera virtual. Las configuraciones disponibles varían según las especificaciones que trae cada vendedor, como puede ser CISCO, Juniper y Arista. No se encontró evidencia de que incluyera frameworks o extensiones que facilitaran la experimentación del protocolo AVB/TSN ni que esto fuera un objetivo característico de este simulador como fue el caso de la extensión para OMNeT++ , CoRE4INET .

2.2.3. Mininet

Mininet es un emulador de red de código abierto que crea una red con nodos virtuales, switches, controladores y enlaces [23]. Los switches están implementados para soportar el protocolo OpenFlow para la experimentación con redes definidas por Software (en inglés Software Defined Networking, SDN).
Provee las ventajas de ser de fácil instalación y de rápido arranque. No es necesario programar estrictamente para utilizarlo, pero si se lo desea provee una API en lenguaje Python para hacerlo.
A pesar de mostrar una usabilidad flexible y de fácil entendimiento, las herramientas que ofrece están orientadas al trabajo con SDNs por lo que no ofrece implementaciones para simular redes que trasmitan información siguiendo los principios de AVB/TSN . Por esta razón, se decidió por probar con otro sistema de simulaciones más acorde a las experimentaciones con AVB/TSN que se realizarían posteriormente en un entorno físico en la presente tesis.

2.2.4. OMNeT++

Para la realización de las simulaciones se optó por OMNeT++ [21], un framework y biblioteca de código público creado principalmente para la construcción de simuladores de red por OpenSim Ltd [24]. El entorno de simulación está en constante desarrollo y mantenimiento y puede encontrarse en el repositorio abierto del proyecto OMNeT++ Discrete Event Simulator [25]. Entre varias herramientas, provee un Entorno de Desarrollo Integrado (en inglés Integrated Development Environment, IDE), basado en la plataforma Eclipse [26] y un entorno de desarrollo gráfico en tiempo de ejecución. Este framework funciona como un esqueleto para modelar redes, pero no posee modelos para protocolos de red, como IP [27] o HTTP [28]. Para realizar simulaciones que utilicen protocolos de red es necesario integrar en el proyecto de OMNeT++ otro framework que simule esos modelos, como es el caso de INET. La plataforma provee una arquitectura de componentes para realizar los modelos. Estos componentes, también llamados módulos, están implementados en el lenguaje de programación C++ y luego se ensamblan en componentes o modelos de mayor tamaño utilizando un lenguaje de alto nivel llamado NED [29].
NED es fundamentalmente utilizado para describir topologías de red. Es un lenguaje flexible que, mediante parámetros, permite controlar la cantidad y el tipo de elementos que componen la red, al igual que su configuración de interconexión. Está caracterizado principalmente por poseer una estructura modular jerárquica, una descripción de topología flexible, uso de parámetros y soporta la partición del modelo para ejecución paralela. Para ejemplificar el lenguaje, se extrae y adapta un fragmento de código de una de las simulaciones de la presente tesis.

```
network small_network
{

  parameters:
    @display("bgb=,,white");

  submodules:
  node1: Node1 {
    @display("p=39,129");
  }

  node2: Node2 {
    @display("p=39,21");
  }

  switch1: Switch1 {
    parameters:
      @display("p=147,70");
    gates:
      ethg[2];
```

```
21      }
22
23      connections:
24
25        node1.ethg <--> Eth100M {  length = 20m; } <--> switch1.ethg[0];
26        node2.ethg <--> Eth100M {  length = 10m; } <--> switch1.ethg[1];
27    }
28
```

Listado de código 2.1: Extracto de código de lenguaje NED en donde se implementa una red simple de dos nodos conectados por un switch.

En el listado de código 2.1, se modula una red pequeña de dos nodos conectados por un switch. En el ejemplo mencionado, podemos observar como por medio de estructuras de datos de lenguaje C++ se define la red en cuestión. En particular, para definir la red es necesario configurar dos tipos de elementos: relativos al simulador (***parameters*** y ***submodules***) y a la topología (***nodes***, ***switches***, ***connections***). Los parámetros determinan qué aspectos serán configurables en la entidad en cuestión y, en caso de que corresponda, se les asigna valores por defecto. En este caso, se configura la posición en la que cada módulo se va a graficar en el entorno visual del simulador. Sin embargo, otros elementos a modificar pueden ser: la clase de tráfico a transmitir, el tamaño de la carga útil (en inglés *payload*) y toda propiedad que determine ese módulo en particular como parametrizable. Los submódulos representan las distintas entidades que forman el módulo, en este caso, los componentes de la red. En este caso se modelan nodos y switches. Cada nodo y cada switch es implementado de una forma en particular en su archivo *.ned*. Por último, las conexiones establecen cómo los distintos submódulos se van a relacionar entre ellos, pudiendo determinar también el tipo y la longitud del enlace. En el simulador OMNeT++ el modelo de red definido se compila en código C++ y se enlaza al ejecutable del simulador. Por este motivo, se puede construir velozmente el modelo internamente cuando comienza la simulación. Además de este tipo de módulos, existe otro tipo de archivos de extensión *.ini* que funcionan para configurar los parámetros de la red definidos en los archivos de tipo NED como ***parameters***, tales como el ancho de banda reservable para la simulación y el algortimo de selección de transmisión de tráfico.

2.2.5. INET

En el contexto de ***OMNeT++*** se utilizó también el modelo de simulación de redes de código abierto llamado INET. INET es mantenido por quienes desarrollan OMNeT++ y se encuentra en constante desarrollo. Este modelo agrega funcionalidades para simular redes alámbricas, inálambricas y móviles. Está construído bajo el concepto de módulos que se comunican mediante el envío de mensajes, como es el caso del protocolo AVB/TSN.

2.2.6. NeSTiNg

Network Simulator for Time-sensitive Networking (NeSTiNg) [30] es un modelo de simulación para el protocolo TSN. Utiliza el INET y lo extiende agregando componentes que soportan TSN. Es un proyecto de código abierto mantenido por un grupo de estudiantes del grupo de Sistemas Distribuidos de IPVS, la Universidad de Stuttgart. Fue presentada en 2018 por uno de sus autores en la reunión interina de septiembre de IEEE 802.1 Working Group. A pesar de que ofrece varias funcionalidades aptas para AVB/TSN , los protocolos que implementa son pertenecen mayoritariamente a TSN y sus funcionalidades giran en torno a satisfacer las necesidades de este último. Dada que la presente tesis se enmarca en un contexto más orientado a AVB que a TSN, las herramientas provistas por esta extensión se iban fuera de alcance de los experimentos que se proponían realizar. Además, poseía menos años de antigüedad que la otra extensión candidata, CoRE4INET , creada en 2014, lo que se refleja en una comunidad más pequeña y un repositorio de código abierto menos extenso y con un menor historial de cambios.

2.2.7. CoRE4INET

Para modelar las redes que utilizaran el protocolo AVB/TSN que se busca examinar en esta tesis, se encontró también otra extensión a INET, llamada CoRE4INET [31]. Esta extensión de código abierto se encuentra en constante mantenimiento y expansión y fue creada por el grupo de investigación CoRE (Communication over Realtime Ethernet) de la universidad de Hamburgo de ciencias aplicadas HAW-Hamburg. Este proyecto provee protocolos de tiempo real para Ethernet como: AVB/TSN, IEEE 802.1Q / IEEE P802.1p VLANs [17] y sistema de prioridades y TTEthernet [32]. Dado el análisis hecho previamente sobre el estado del arte en simulaciones de redes de comunicación, la conclusión es que con su extensa disponibilidad de protocolos relacionados con redes vehiculares, en particular los requeridos por AVB, este framework brinda las soluciones necesarias para simular redes en el contexto de este trabajo de investigación y por lo tanto fue el elegido para realizar las simulaciones que se describen a continuación.

2.3. Experimentos

La propuesta de realizar los experimentos en un entorno de simulación tenía el objetivo de entender el contexto con el que se iba a trabajar en el banco de pruebas físico. El objetivo de los experimentos era verificar que el banco de pruebas fuera capaz de ser un entorno en donde AVB/TSN pudiera ejecutarse correctamente. De esta forma, se eligieron topologías que provocaran situaciones desafiantes para el protocolo, generando

momentos de alta congestión en donde los algoritmos de selección de tráfico se vieran forzados a elegir qué clases transmitir con mayor prioridad que otras. Se modelaron las tres topologías principales que se estarían probando más adelante:

1. Estrella
2. Cadena
3. Anillo

Se comenzó con la topología Estrella, dado que, desde el punto de vista del control de tráfico que implementan los protocolos, esta era una configuración que debería presentar el menor caso de eventos inesperados. Asimismo, permitía la transmisión de los tres tipos de tráfico que interesaban: clase A, clase B y Mejor Esfuerzo. Luego, se optó por la topología Cadena, ya que generaba varios puntos de congestión para los 3 tipos de tráfico en los puertos de los switches que se conectaban entre ellos. Puesto que la conexión entre un switch y otro era mediante un solo puerto, ese puerto se veía saturado por el tráfico generando así una situación útil para la investigación. Por último, se decidió por la topología Anillo, fundamentalmente porque esta configuración presenta una forma de ciclo por lo que generaba interés observar si el protocolo iba a poder sobrellevar esta situación correctamente.

Para comenzar con las simulaciones, se optó por realizar una configuración simple para introducirse al uso del simulador, comprender sus funcionalidades básicas y entender los parámetros a usar para realizar las transmisiones necesarias. La figura 2.1 muestra la primer topología: dos nodos conectados mediante un switch. Cada nodo implementa internamente los protocolos necesarios para una transmisión de tráfico apta para AVB/TSN , como son el Protocolo SRP y el algoritmo de selección de tráfico Credit Based Shaper. La transmisión de información fue de tráfico de clase A, la clase con mayor prioridad dentro del contexto de AVB/TSN.
Esta topología elemental nos permitió visualizar por primera vez una simulación en el contexto de OMNeT++ , comprendiendo cómo se envían los mensajes y la información sobre los mismos que puede obtenerse desde el simulador. También nos dio conocimiento sobre cómo configurar los nodos para que envíen el tipo de tráfico en particular que deseábamos analizar y cuál es el comportamiento que deberíamos esperar de las simulaciones más complejas que vendrían a continuación.

En cuanto a la figura 2.2, se muestra una topología con forma de estrella en la que seis nodos se encuentran conectados a un switch. El tráfico se divide en tres tipos: clase A, clase B y Mejor esfuerzo. Los tres nodos que figuran del lado izquierdo del switch son los nodos ***Talkers*** que generan la información a transmitirse y comienzan

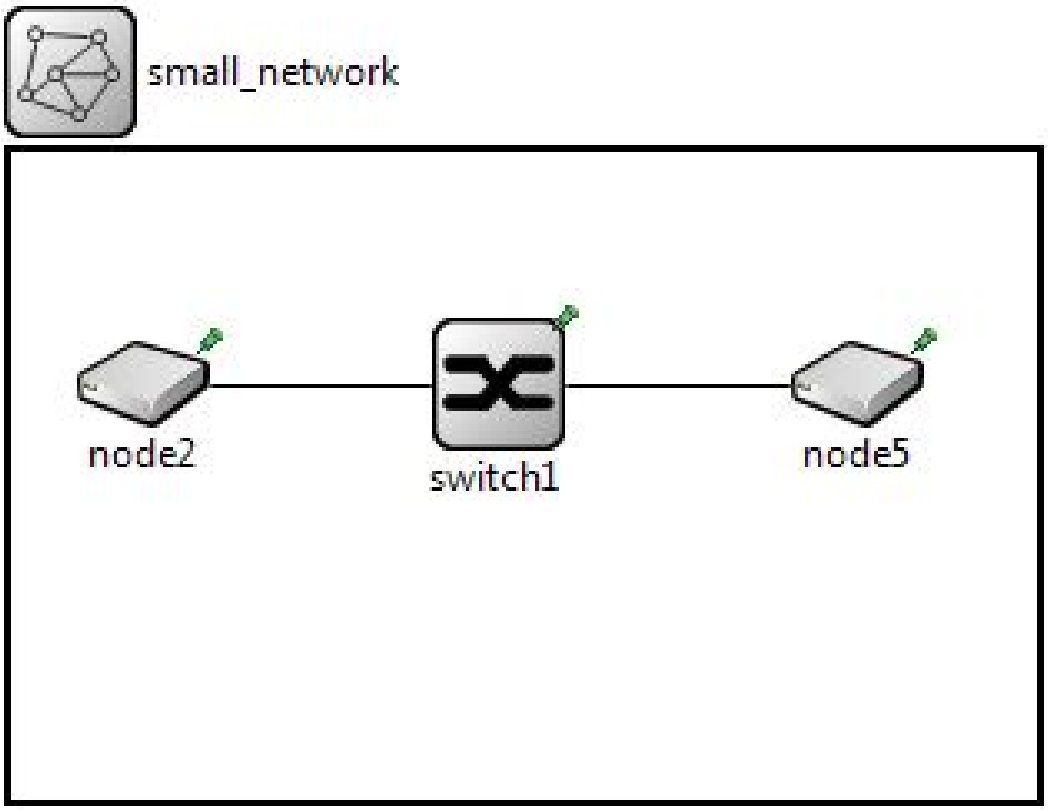


Figura 2.1: Captura de pantalla de una topología simple en simulador OMNeT++. Se comenzó con un escenario de poca complejidad para introducirse al simulador y entender sus funcionalidades básicas. Se procedió a transmitir tráfico de clase A como paso inicial para entrar en contexto con el protocolo AVB/TSN.

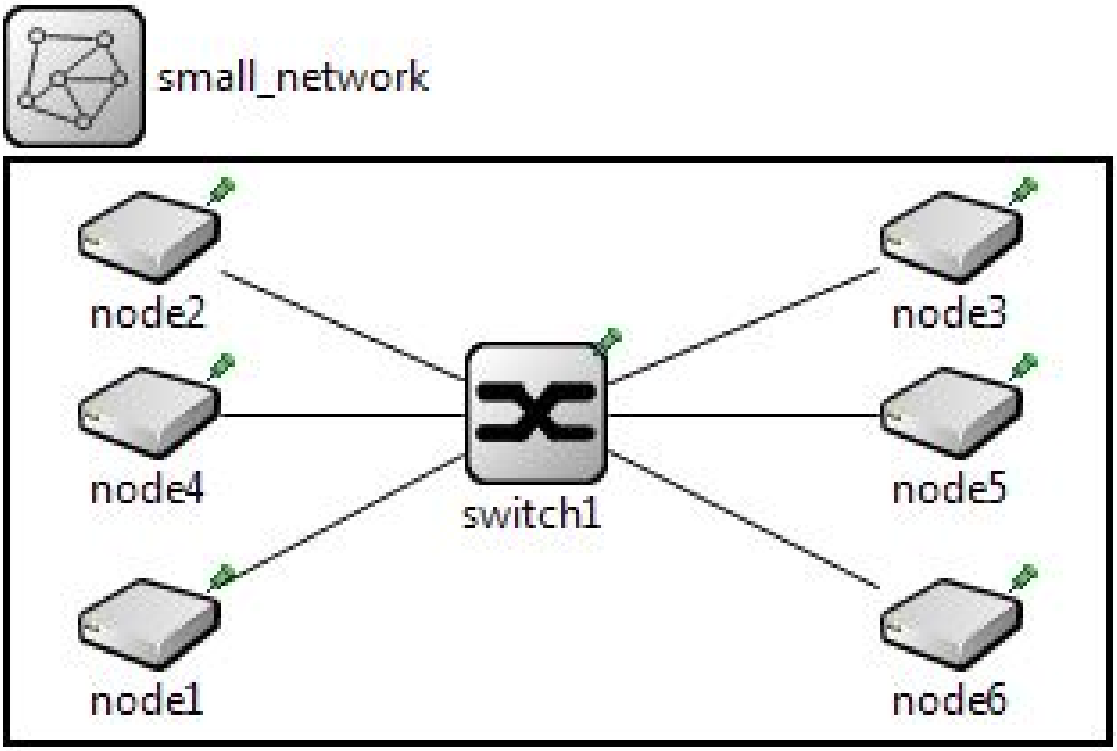


Figura 2.2: Captura de pantalla de la topología Estrella en simulador OMNeT++. Este escenario brinda la posibilidad de tener tres tipos de tráfico distinto transmitiendo al mismo tiempo. El switch, de esta forma, se enfrenta a la tarea de tener que encolar y enrutar dos tipos de tráfico más que en el caso anterior.

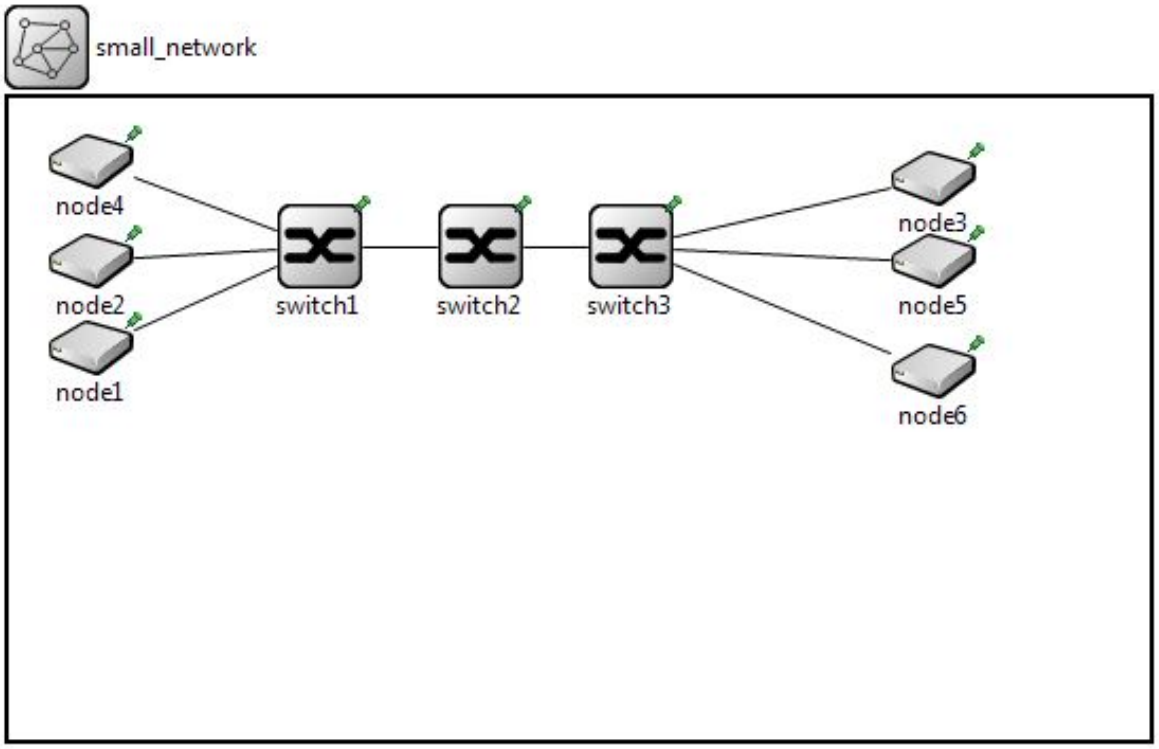


Figura 2.3: Captura de pantalla de la topología Cadena en simulador OMNeT++. Este caso introduce dos switches más en el entorno virtual, lo que permite explorar la conexión de switches entre ellos y el desempeño del protocolo al agregar más puntos de congestión.

con el envío de datos. Los tres nodos que se muestran del lado derecho del switch son los ***Listeners*** que se encargan de recibir la información transmitida por el resto. Cada par de nodos ***Talker - Listener*** manejó un tipo distinto de tráfico. De esta forma, el nodo 4 transmitía tráfico clase A que era recibido por el nodo 5, el nodo 2 tráfico clase B que se enviaba al nodo 3 y, por último, el nodo 1 comenzaba con la transmisión de tráfico de Mejor Esfuerzo que terminaba en el nodo 6. Para distinguir los distintos tipos de tráfico, el tráfico de tipo AVB/TSN se dividió en dos streams distintos: stream 1 que pertenece a tráfico de clase B y stream 2 que pertenece a clase A, y el tráfico de Mejor Esfuerzo fue etiquetado como PcP 7 siguiendo el protocolo IEEE 802.1Q.

La figura 2.3 presenta la topología de cadena en el entorno virtual del simulador. Esta configuración agrega dos switches más a la red, lo que implicó el aprendizaje de entender cómo conectar switches entre ellos y más puntos de congestión para desafiar al protocolo. Nuevamente, se utilizaron tres tipos de tráfico distintos transmitidos por tres pares de nodos diferentes. El switch 2 en este caso es el más interesante, puesto que funciona como un punto de congestión para todo tipo de tráfico, al tener un solo puerto de entrada y un solo puerto de salida. En consecuencia, funciona como el cuello

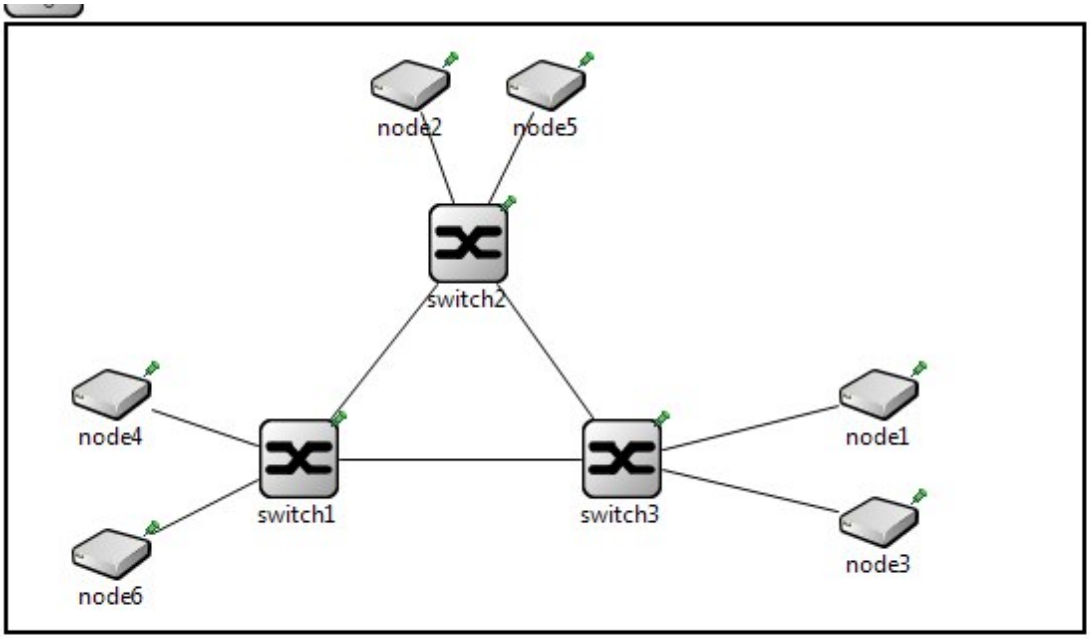


Figura 2.4: Captura de pantalla de la topología Anillo completa en simulador OMNeT++. Esta configuración presenta una estructura cíclica lo que ocasiona obstáculos en el protocolo. Se encontró que no fue posible establecer una simulación exitosa con esta estructura, dado que se generaban errores de sobreescritura de mensajes y la misma era abortada.

---

de botella de la red.

La topología con forma de anillo se muestra en la figura 2.4, en su completitud. Esta configuración suma el desafío de tener una estructura cíclica, agregando así, un obstáculo más para el protocolo. Los resultados no fueron positivos para este tipo de configuración, ya que se generaban errores de sobreescritura de ciertos mensajes y esto provocaba que la simulación fuera detenida y abortada. El problema se da durante el intercambio de mensajes del protocolo SRP. Cuando se envía el mensaje de Talker Advertise del switch 2, se envía al switch 1 y al switch 3 dado que ambos estan conectados al primer switch mencionado. Al mismo tiempo, tanto el switch 1 como el switch 3 al recibir este mensaje de Talker Advertise lo envían en modo broadcast (difusión amplia, en español) por todos sus puertos. De esta manera, cuando el switch 3 vuelve a recibir el mensaje de Talker Advertise intenta actualizarlo de nuevo causando el error de sobreescritura que aborta la simulación. Una posible causa de este error es el hecho de que no hay implementación del protocolo de Spanning Tree en CoRE4INET que controle la tormenta de mensajes de tipo broadcast que se termina generando. Si se aplicara Spanning Tree los bucles que genera esta topología se podrían prevenir. El algoritmo transforma una red física (o en

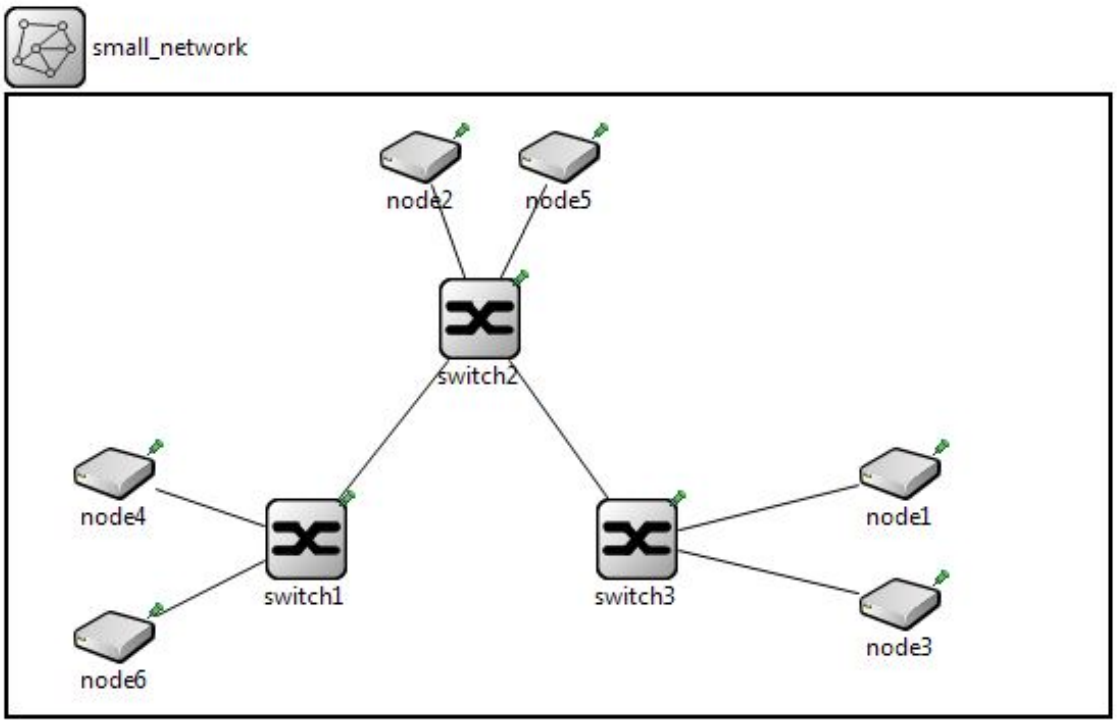


Figura 2.5: Captura de pantalla de la topología Anillo con una conexión menos en simulador OMNeT++. Fue necesario desconectar dos switches para romper con la estructura cíclica, puesto que el framework no la pudo soportar. Una vez realizada la desconexión, la simulación pudo ser completada.

---

este caso virtual) cíclica en una red lógica en forma de árbol y, luego, desactiva los enlaces de conexión que determina como redundantes. En consecuencia, al desactivar enlaces que generan bucles el mensaje de Talker Advertise no sería sobrescrito y la simulación podría funcionar correctamente.

Por consiguiente, en la figura 2.5, se muestra la topología Anillo pero con una desconexión. Se desconectaron dos de los switches para romper con la estructura cíclica que ocasionaba problemas en el simulador y, una vez realizado este cambio, la simulación pudo ser ejecutada.

2.4. Análisis de resultados

Una vez realizadas las simulaciones, fue posible ver en detalle los distintos mensajes que se envían de un dispositivo a otro, al igual que visualizar comportamientos esperados del protocolo de manera clara y minuciosa.

Nos interesaba poder obtener evidencia de que el protocolo AVB/TSN estaba siendo implementado para, de esta forma, obtener claridad sobre su funcionamiento. Por lo

tanto, buscábamos resultados que reflejen el uso de los protocolos que incluye, resultados en donde se pudiera visibilizar el funcionamiento de algoritmos como CBS y el protocolo SRP. También requeríamos entender que las garantías que el protocolo ofrece, tal como el límite en la latencia para tráfico de alta prioridad como el de clase A y B se cumplieran en este contexto virtual. Dadas las herramientas que CoRE4INET ofrece y la complejidad de las mismas, hubo ciertos resultados que no fueron posibles encontrar en el límite de tiempo con el que contábamos. Este fue el caso de la medición de la latencia de la transmisión de un paquete de un extremo a otro. Sin embargo, a continuación se muestran gráficos y elementos que nos proveyó el simulador que nos permitió comprender de mejor forma el funcionamiento de AVB/TSN .

En la figura 2.6 puede visualizarse el monto de créditos transmitidos en el puerto 2 del switch 1 en la topología Estrella. Es interesante notar cómo se encuentran presentes tanta la pendiente de inactividad como la pendiente de envío en el gráfico. Cada vez que decrece o crece la curva, esta mantiene la misma pendiente para cada caso: pendiente de inactividad para crecer y pendiente de envío para decrecer. Además el punto mínimo de la curva es siempre el mismo, lo que refleja que la transmisión dura hasta que los créditos son agotados. En caso contrario, el punto máximo no se mantiene fijo lo que indica que no se llega a acumular la cantidad máxima de créditos de la clase graficada antes de iniciar su transmisión. Esto demuestra entonces que el puerto no llega a saturarse, el tráfico de este tipo de clase para este puerto particular es transmitido con poco retraso una vez que llega al puerto. Sin embargo, se ve que cada vez la acumulación de créditos es mayor, por lo que el retraso aumenta con el pasar del tiempo. Se visualiza que se transmite hasta gastar todos los créditos, luego se espera hasta llegar de vuelta al monto 0 de créditos y ahi se sigue transmitiendo lo que faltaba en una curva de menor amplitud. En otras situaciones, se acumula créditos dado que no puede transmitir a pesar de tener cantidad de créditos mayor a cero, una vez que se libera transmite, llega al mínimo, para y tiene que volver a esperar a llegar a 0 para transmitir lo que le quedaba. De esta forma, en este gráfico se verifican las cualidades que caracterizan CBS, dándole validez a la implementación del protocolo brindada por el framework.
Por otro lado, el simulador también proveyó información detallada sobre los mensajes que se transmitían de un nodo a otro.

Se pueden observar los detalles de un mensaje en particular durante una de las simulaciones de la topología Cadena en la Figura 2.7. El mensaje es una trama de tipo AVB perteneciente a la clase *AVBFrame* que brinda CoRE4INET en su implementación del protocolo. En el momento de la captura, la trama se encuentra en la cola número 6 del puerto 1 creada por el algoritmo de selección de tráfico del switch 2. Se brinda información también sobre el módulo desde donde se envió la trama, en el campo *senderModule*, y el

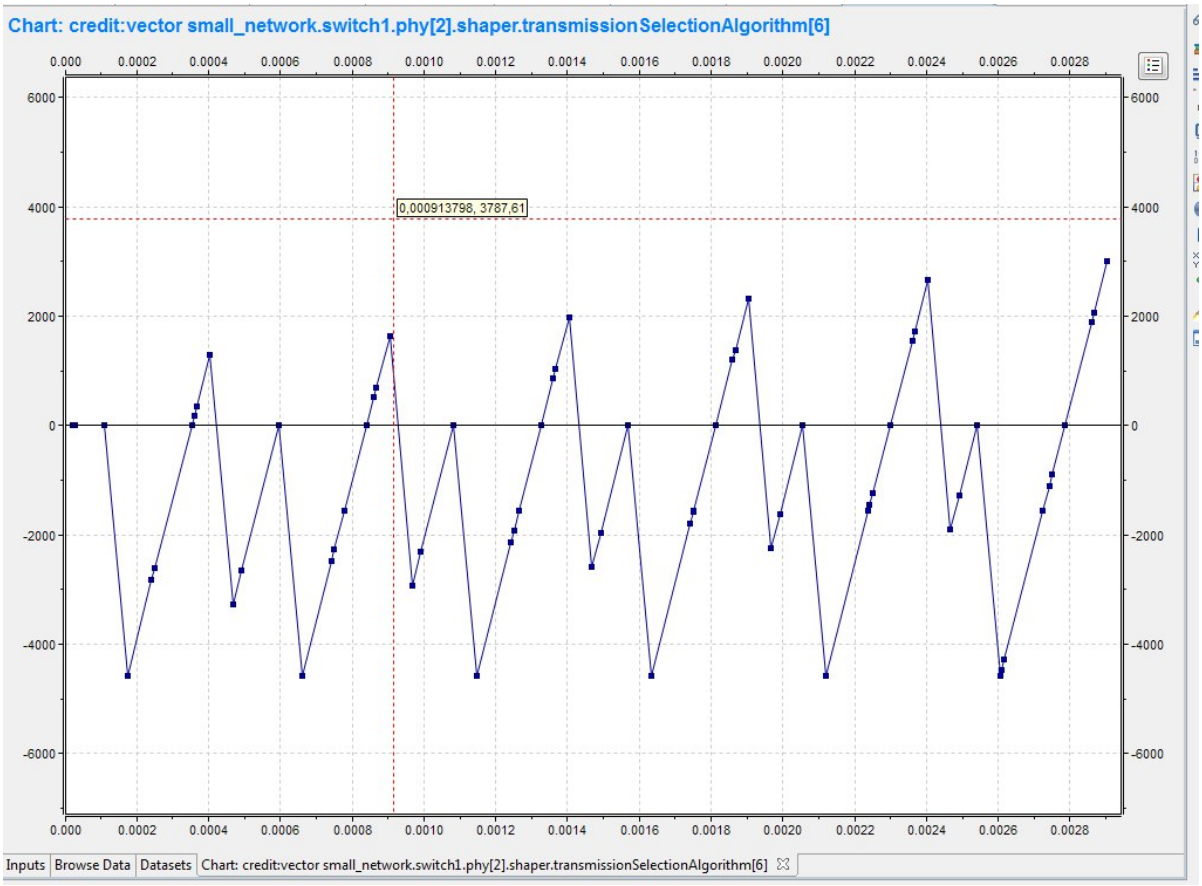


Figura 2.6: Resultado de la simulación correspondiente a la topología Estrella. Se muestra la variación en la cantidad de créditos en base al tiempo transcurrido en la clase B de tráfico transmitida en el puerto 2 del switch 1. Una transmisión dura desde el punto mínimo al punto máximo de créditos. Una vez que se transmite toda la información, la cantidad de créditos incrementa hasta alcanzar el valor 0. Cuando incrementa hasta un máximo local mayor a 0 es porque no puede transmitir, dado que el puerto está ocupado con otra tarea. Puede observarse que la cantidad máxima de créditos va aumentando con el paso del tiempo, lo que refleja un aumento en la demora en el puerto para transmitir la clase de tráfico correspondiente. Además se puede ver que la cantidad de créditos gastados por transmisión es siempre la misma, por lo que siempre se envía la misma cantidad de datos.

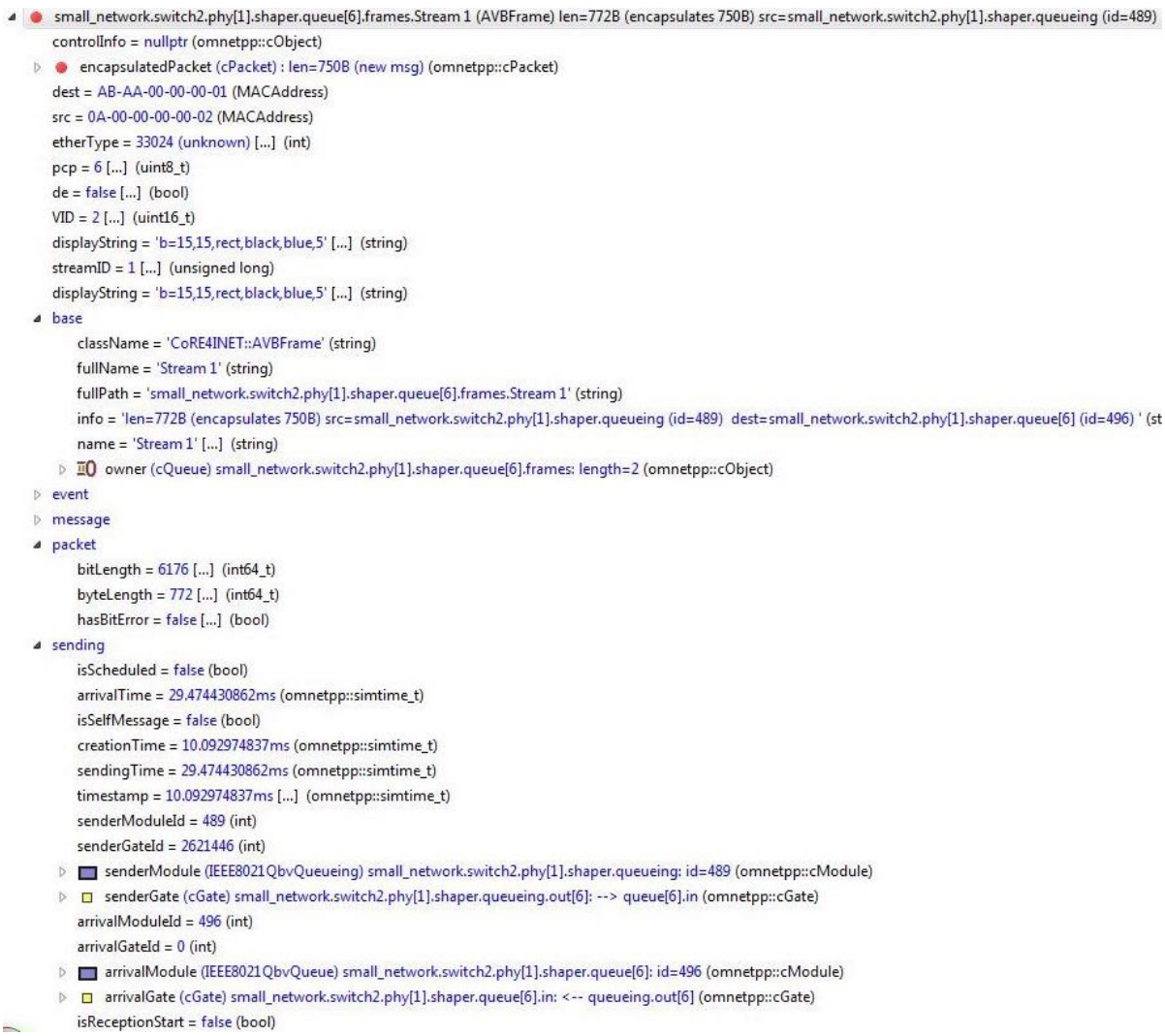


Figura 2.7: Captura de pantalla sobre los detalles que provee el simulador sobre un mensaje transmitido durante una de las simulaciones. El mensaje es una trama de tipo AVB de clase B y fue recibido por el switch 2 en la topología Cadena. Entre otros datos, puede observarse la dirección MAC de origen como la de destino, la prioridad PCP, el vlan ID, el número de Stream al que pertenece el mensaje (en este caso 1). Esta trama en el momento de la captura se encuentra en la cola número 6 del algoritmo de selección de tráfico del puerto 1 del switch.

módulo al que llegó, en el campo *arrivalModule*. Estos detalles nos permiten entender que la trama es insertada en la cola 6 por el algoritmo de encolado que provee el algoritmo de selección de este puerto en particular. Además, se especifica el tiempo de creación de la trama (*creationTime*) y el momento en el que llegó a este módulo en particular (*arrivalTime*), lo que da la posibilidad de realizar cálculos y obtener conclusiones sobre la latencia de una trama en la simulación. Otros datos disponibles son las direcciones MAC de origen y destino, el tamaño del paquete, el id de la vlan a la que fue asignado, el número de flujo (*stream*) al que pertenece y la prioridad PCP que tiene. Esta información que provee el simulador es de gran utilidad si se desea realizar un seguimiento de cada paquete que se transmite en la red virtual, permitiendo establecer estadísticas y cálculos con la misma. Asimismo, brinda visibilidad sobre los valores que están tomando los parámetros de los elementos de la red y, de esta forma, ayuda a encontrar más fácilmente la causa de posibles complicaciones que esté teniendo la red en un debido momento.

2.5. Limitaciones

Aunque CoRE4INET nos permitió visualizar las pruebas que se harían posteriormente, el protocolo presenta una serie de limitaciones que hicieron que se presenten complicaciones no esperadas a la hora de utilizar el simulador. Por un lado, la instalación de OMNeT++ es dificultosa, teniendo particulares inconvenientes en sistemas operativos diferentes a Windows. Esto generó grandes demoras para configurar el entorno previo a las simulaciones.

Por otro lado, la comunidad que utiliza CoRE4INET no es muy extensa, por lo que no fue sencillo obtener documentación que ayudaran a la instalación y entendimiento del framework. Además, como puede verse en la figura 2.4 no fue posible modelar la topología Anillo en su totalidad, dado que el simulador no logró soportar su estructura cíclica. Se daban errores de sobreescritura sobre los mensajes que se utilizan para determinar el dominio SRP, por lo que la simulación fallaba y se abortaba. De esta forma, al no poder establecer el dominio SRP la transmisión de los flujos de datos no pudo iniciarse. Como se mencionó previamente, CoRE4INET no poseía una implementación del protocolo Spanning Tree, sin embargo, la extensión de OMNeT++ , INET, contenía un módulo que lo representaba. Por consiguiente, las clases creadas para representar entidades de una red apta para AVB/TSN no hicieron uso de un algoritmo que ya estaba implementado por el framework en el que CoRE4INET se basa, lo que denota la complejidad que trae hacer uso de un sistema de capas. La necesidad de utilizar frameworks distintos e inconexos para poder contar con todos los protocolos requeridos para realizar una simulación genera confusión y complica la usabilidad del sistema final.

Asimismo, el framework no era muy abierto en los elementos de la red que categorizaba como parametrizables. Si se deseaba, por ejemplo, configurar ciertos parámetros del algoritmo de CBS no había otra opción que alterar el código fuente del mismo.
Por último, idealmente, habría sido positivo exportar la información de los mensajes a archivos de extensión *pcap* , pero encontramos que para realizarlo había que modificar el código fuente de los módulos para que hereden de una clase en particular, lo que complejizaba bastante el proceso y por cuestiones de tiempos se optó por no realizarlo. Esto es una limitación dado que si hubiera sido más sencillo podríamos haber realizado gráficos y estadísticas basándonos en los detalles de los paquetes, como la latencia de un nodo a otro.

2.6. Conclusiones

La posibilidad de realizar simulaciones brinda una serie de ventajas que incluye la visualización del envío de paquetes, la posibilidad de depurar para observar detalles puntuales de la transmisión, la configuración sencilla de parámetros, y la habilidad para crear redes de gran tamaño sin contar con el hardware necesario.

Sin embargo, las limitaciones que posee el simulador nos muestran que existe espacio para la innovación en este campo. Por un lado, podría expandirse la portabilidad del mismo puesto que su instalación no fue exitosa en ningún sistema operativo más que Windows. Además, la herramienta de dibujo gráfica podría contar con una interfaz más clara para el usuario y con una manera de graficar intuitiva y parametrizable. Por último, existe oportunidades de mejora en cuanto a la implementación de los protocolos para poder soportar topologías circulares como por ejemplo la anillo.

Capítulo 3

Banco de Pruebas Intravehicular

El objetivo principal de este proyecto es la construcción de un banco de pruebas formado por placas computadoras de propósito general y software de código abierto capaz de ejecutar el protocolo AVB/TSN . Nuestro trabajo es la continuación y consolidación de un labor previo en el cual solamente se llevaron a cabo tareas preliminares, como la puesta en funcionamiento de una topología elemental. El aporte de la presente Tesis consistió en extender el banco, agregando topologías de mayor complejidad y realizando experimentaciones que ataquen elementos más profundos del protocolo. El ambiente de pruebas brinda una herramienta que permite explotar AVB/TSN y, al mismo tiempo, funciona como un instrumento de fácil acceso y replicación que facilita la investigación de Ethernet Automotriz en el ambiente académico. La robustez y aplicabilidad del protocolo se ve demostrada por medio de resultados empíricos y no, como se encontró que es lo usual en el estado del arte, meramente por simulaciones o modelos teóricos. La configuración del entorno de pruebas, las metodologías utilizadas y los resultados obtenidos se describen en este capítulo.

3.1. Descripción de Hardware

El protocolo vehicular AVB/TSN requiere configuraciones de hardware particulares que permitan el funcionamiento correcto de los estándares necesarios:

1. General Precision Time Protocol (gPTP) [12] para la correcta sincronización de todos los elementos que conforman la red

2. Credit Based Shaper (CBS) [33]

Por un lado, el protocolo gPTP asegura una precisión de 1 µs para 7 saltos en la red. Lograr este nivel de exactitud requiere herramientas capaces de asignarle a los paquetes de

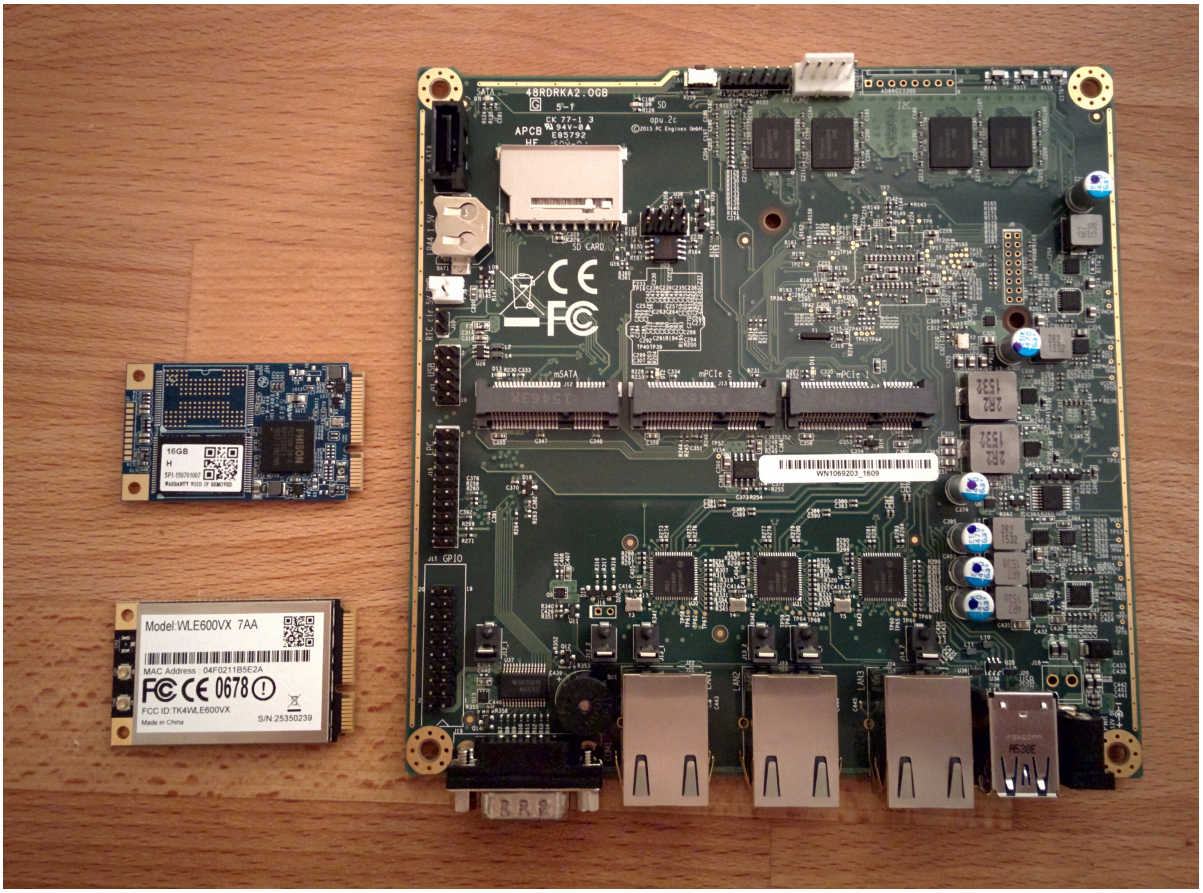


Figura 3.1: Vista de un dispositivo Apu2C4 [36]

sincronización una marca temporal con una exactitud de nanosegundos [34]. Asimismo, la tarjeta de red que utilicen los dispositivos debe tener la capacidad de poder discriminar entre prioridades de tráfico altas o bajas, poseyendo distintas colas para cada nivel de prioridad. Sumado a esto, para que AVB/TSN funcione correctamente, el algoritmo CBS debe poder aplicarse en hardware como el algoritmo final de selección de tráfico a transmitir.

Entró en consideración el uso de la placa computadora Raspberry Pi dada su extensa comunidad y renombre, pero fue descartada al no poseer el hardware necesario para sellarle el timestamp (marca de tiempo, en español) a los paquetes [35].

Por otro lado, la tarjeta de red Intel I210 1 Gbit/s cumple con todas estos requerimientos y, por lo tanto, se buscaron dispositivos que contaran con la misma.

Las placas computadoras Apu2C4 y Apu2D4 de PcEngine [37] que se muestran en la figura 3.1 poseen tres de estas tarjetas de red junto con un microprocesador *AMD G-series GX-412TC* que trae consigo una arquitectura de 64 bits, a diferencia de otras placas computadoras que mayoritariamente manejan arquitectura de tipo ARM.

El switch Netgear GS716Tv3 representado en la figura 3.2, es un dispositivo que

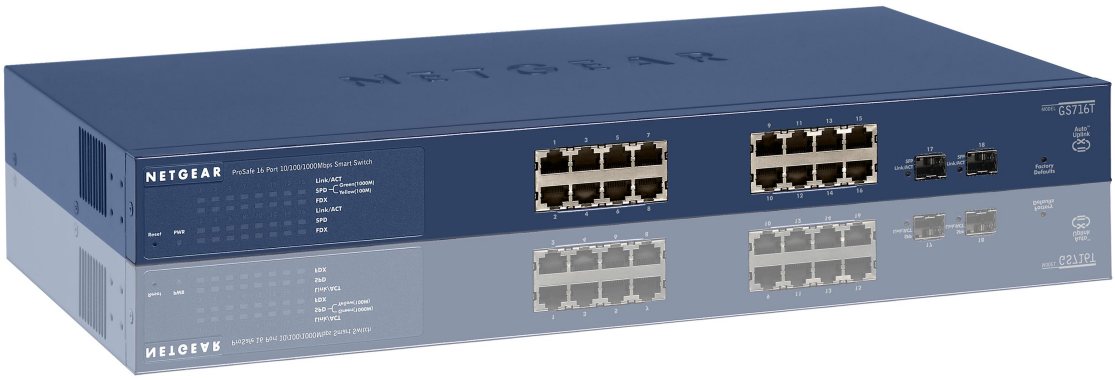


Figura 3.2: Vista del switch Netgear GS716Tv3 [38]

permite el tráfico de tipo AVB y posee 16 puertos que permiten operar con múltiples flujos de tráfico AVB/TSN al mismo tiempo.

3.2. Descripción de Software

Nuestro requerimiento era contar con una implementación de AVB/TSN capaz de correr en los dispositivos a usar y generar las transmisiones que se pretendían probar en las topologías. Dado que es un sistema que consiste de varios estándares, se buscaba una implementación que contara con documentación extensa y una comunidad capaz de responder en caso de dudas o problemas que pudieran surgir de nuestro lado.

Existen varias implementaciones de código abierto que fueron consideradas para este proyecto:

1. **Avnu Alliance [9]**: se trata de una asociación entre empresas de manufactura y electrónica orientadas al automovilismo que colaboran juntas para establecer y certificar la interoperablidad de los estándares de AVB/TSN . Buscan simplificar la sincronización de la red generando un nuevo ecosistema certificado. Su objetivo es resolver problemas técnicos y de desempeño, tales como la sincronización en tiempo real, tiempos precisos, reserva de ancho de banda y traffic shaping (conformado

de tráfico, en español), al mismo tiempo que apelan a reducir el costo total de propiedad minimizando el tiempo de despliegue y mantenimiento. Su foco está puesto en tecnologías automotoras, sistemas audio y video profesional y mercados de electrónica industrial y comercial. Su proyecto [39] de código abierto implementa todos los estándares requeridos por AVB/TSN .

2. **Automotive Grade Linux [40]:** es un proyecto originado por la fundación Linux [41] cuya motivación es construir un sistema operativo abierto basado en Linux para aplicaciones automotoras. Con el apoyo de parte de Mercedes-Benz, Toyota, Ford [42], entre otras el proyecto resultaba interesante y poseía una comunidad activa y extensa. Sin embargo, tanta la gran magnitud del proyecto como las tecnologías con las que trabaja que se escapaban del alcance del banco de pruebas que se pretendía armar en este contexto.

3. **AudioScience [43]:** es una compañía fundada en 1996 con el objetivo de proveer soluciones específicas de diseño y manufactura de audio. En 2012, colaboraron con Avnu Alliance para construir un sistema AVB centralizado en audio a partir del repositorio de código abierto de Avnu Alliance, OpenAvnu [39].

Después de haber analizado las diferentes opciones se optó por el repositorio provisto por Avnu Alliance, OpenAvnu. Esto se debe a que posee una extensa y detallada documentación [44] y, también, implementa el estándar completo de AVB/TSN :

1. Por un lado, dispone de IEEE802.1AS (gPTP) que se encarga de que cada nodo perteneciente a la red sincronice su reloj con los demás y establezcan un dominio gPTP.

2. Luego, se conforma el dominio SRP con los miembros de la red que estén aptos para transmitir tráfico de tipo AVB/TSN . Esto se logra con el uso de los protocolos Multiple Mac Registration Protocol (MMRP), Multiple Stream Registration Protocol (MSRP) y Multiple VLAN Registration Protocol (MVRP) basados en Multiple Registration Protocol (MRP) [45].

3. La intersección de los dominios gPTP y Stream Reservation Protocol (SRP) conforman el dominio AVB (IEEE 802.1BA) en donde los nodos que funcionan como *Talkers* anuncian los flujos que disponen para transmitir y los *Listeners* se acoplan a los flujos anunciados.

4. Una vez establecido el canal de transmisión entre un *Talker* y un *Listener*, el protocolo IEEE 802.1Qav se ocupará de garantizar una transmisión con un retardo máximo y un ancho de banda específico según la prioridad del flujo.

En conclusión, el repositorio OpenAvnu maneja todos los estándares necesarios para AVB [6] y, de esta manera, fue el candidato seleccionado para la construcción del proyecto en el que se basa la presente tesis.

3.3. Herramientas Adicionales

Para cada evaluación de desempeño, se utilizaron herramientas adicionales que permitieron crear las condiciones apropiadas para el experimento, realizar configuraciones necesarias y analizar los paquetes que se enviaron en cada transmisión.

Ethtool

Ethtool es una herramienta de Linux que permite modificar las configuraciones de una tarjeta de red [46]. A pesar de que utilizamos dispositivos cuyas tarjetas de red permiten transmisiones de 1 Gbit/s, las mismas fueron limitadas a 100 Mbit/s. Se decidió en este valor, dado que buscábamos evaluar el desempeño de los nodos en una red congestionada y este cambio nos permitía saturar la red más rápidamente.

Tcpdump

`tcpdump` [47] permite capturar paquetes que se envían o reciben por una interfaz específica. Además permite asignar un timestamp a la captura, lo que fue esencial para el posterior análisis de las transmisiones. Específicamente se utilizó la versión 4.9.2 de `tcpdump` ya que da la posibilidad de sellar un timestamp no solo desde el software si no también desde el hardware. Un timestamp hecho en software puede verse afectado por ruido introducido por la pila de protocolos de la red (*network protocol stack*, en inglés). Esto se debe a que el timestamp es sellado en la capa de Aplicación y, antes de salir del dispositivo, el mensaje debe pasar por toda la pila de protocolos. Por otro lado, cuando se sella en hardware, el mensaje es sellado justo antes de salir del dispositivo y apenas llega a la tarjeta de red del destino, minimizando así tiempo extra que podría perjudicar la estimación final de demora de un dispositivo a otro. Como en nuestro caso observábamos en detalle de milisegundos los resultados, sellar en hardware era fundamental.

Wireshark

Esta herramienta [48] es uno de los capturadores de paquetes y analizador de protocolos más utilizados en la actualidad. Su funcionalidad es similar a la de `tcpdump`, pero

provee una interfaz gráfica que permite observar de manera clara el contenido de los paquetes. Fue usado para analizar los paquetes contenidos en los archivos *pcap* obtenidos con `tcpdump`. Esto nos permitía observar los mensajes que cada estándar del protocolo enviaba y entender cómo los mismos se componían para posteriormente obtener datos relevantes de ellos.

Python

Python [49] es un lenguaje de programación interpretado, dinámico, de alto nivel y multiparadigma.

Fue de gran utilidad para cumplir dos propósitos fundamentales del proyecto. Por un lado, nos permitió implementar programas de manera sencilla para transmitir tráfico mediante el protocolo Ethernet sin hacer uso de AVB/TSN . Para simular el tráfico mediante Ethernet, hicimos uso de la biblioteca *socket* [50] para crear sockets de tipo *raw* y transmitir tramas directamente en la capa de enlace. A través de esta biblioteca pudimos simular tráfico con la misma tasa de bits y tamaño de trama que el transmitido mediante AVB/TSN . Contar con este otro flujo de datos, fue fundamental para comparar los desempeños entre ambos protocolos. Y, también, se crearon programas en Python que nos permitieron enviar tráfico de mejor esfuerzo en broadcast para congestionar la red.

Por el otro lado, nos facilitó el análisis de los datos obtenidos durante las transmisiones, mediante la extensa variedad de bibliotecas que posee el lenguaje. Aplicamos funcionalidades provistas por la biblioteca *Pcapy* [51] para leer los paquetes obtenidos por `tcpdump`, filtrar los datos que nos interesaban y computar las métricas que deseábamos obtener. Asimismo, utilizar bibliotecas como *Matplotlib.pyplot* [52], *Seaborn* [53] y *Pandas* [54] nos permitió visualizar y comprender los resultados de manera clara.

Phc2sys

Phc2sys [55] es una herramienta de Linux capaz de sincronizar el reloj interno de un dispositivo con el reloj de su hardware. Resultó esencial para sincronizar a los dispositivos con el Reloj Maestro de gPTP seleccionado mediante el Best Master Clock Algorithm (BMCA) [56]. Al poder sincronizar el reloj del sistema con el reloj de hardware, los paquetes capturados por `tcpdump` podían ser sellados con el timestamp acorde al Reloj Maestro. Esto posibilitó la comparación de timestamps de los paquetes al salir de un dispositivo y entrar a otro para, así, calcular las métricas de demora y jitter. Dado que los dispositivos de la red estaban sincronizados, la diferencia entre timestamps realmente

reflejaba el tiempo que tardó un paquete en viajar de un nodo a otro.

3.4. Experimentos Topológicos

Esta sección describe los experimentos topológicos que se realizaron en el banco de pruebas intravehicular. A partir de una configuración inicial, se prosiguió con la extensión del mismo y la apropiada calibración mediante distintos experimentos cuyos objetivos eran comprobar que las garantías que ofrece AVB/TSN fueran cumplidas en distintas configuraciones topológicas.

Para validar el funcionamiento de AVB/TSN en el banco de pruebas, se comparó el desempeño de este protocolo con el de Ethernet bajo las mismas configuraciones topológicas. El desempeño de ambas tecnologías fue evaluado a través de las siguientes métricas: demora de un extremo a otro, pérdida de tramas y fluctuación de retardo (en inglés *jitter*). Dado que el protocolo Ethernet fue diseñado bajo el paradigma de máximo esfuerzo, es decir no ofrece garantías de calidad de servicio, es plausible no garantice el valor de las métricas como sí lo hace AVB/TSN . Para enviar información implementando el protocolo AVB/TSN se hizo uso del código presente en el repositorio dispuesto por Avnu Alliance, OpenAvnu. Por otro lado, se implementó un programa escrito en lenguaje Python para realizar la transmisión del archivo mediante Ethernet. Además, se creó otro programa en Python para enviar tráfico de Mejor Esfuerzo adicional que funcione como tráfico de fondo para saturar la red lo más posible. Los resultados presentados a lo largo de esta sección verifican el correcto funcionamiento del banco de pruebas.

3.4.1. Configuraciones de tráfico

La inyección de tráfico sintético para congestionar la red fue llevada a cabo por un programa desarrollado en Python, cuyas características igualaban las del tráfico a transmitirse por AVB/TSN (imitando el tamaño de la trama de clase A y su tasa de bits). El objetivo de transmitir tramas de Ethernet sin AVB/TSN en simultáneo a las de prioridades AVB/TSN era poder comparar el desempeño de ambos protocolos y, con esto, poder comprender la mejora que trae el segundo.

Se sumaron las siguientes configuraciones:

1. El ancho de banda tanto de las tarjetas de red de los dispositivos como de los puertos del switch fue limitado a 100 Mbit/s para saturar la red de manera más rápidamente

2. El protocolo gPTP fue ejecutado tanto en los dispositivos que enviaban tráfico AVB/TSN como los que transmitían Ethernet simple para establecer un reloj

común entre todos los dispositivos y, de acuerdo a este, poder calcular eficientemente la demora de las tramas

3. Los buffers del núcleo del sistema operativo (en inglés, kernel) no fueron modificados

4. El tráfico de fondo que fue utilizado para saturar la red consistió de datagramas UDP con 1500 B en su payload

5. Este tráfico era transmitido desde la computadora de control hacia alguno de los switches que luego se encargaba de difundirlo por la red

6. En cuanto al flujo de información enviada por Ethernet, el tamaño de la trama fue de 430 B y el ancho de banda de aproximadamente 16,40 Mbit/s.

7. El tráfico de fondo utilizado para saturar la red fue transmitido en forma de *broadcast* a partir de un programa de Python. Esto provoca que el switch tenga que lidiar con tres tipos de tráfico distintos (clase A, clase B y mejor esfuerzo) y, además, con el tráfico de fondo, convirtiéndose en un punto de congestión.

8. Se encontró que el núcleo del sistema operativo de los dispositivos sellaba los paquetes con el timestamp acorde al reloj del software. Esta limitación impedía que se pudiera calcular la demora con respecto al timestamp de los demás dispositivos. Para solucionar esta limitación, se utilizó el programa *phc2sys* [55] que permite al hardware sellar a los paquetes con su reloj, que es el que se sincroniza mediante el Best Master Clock Algorithm (BMCA).

En cuanto el tráfico a enviar, se envió un archivo de video de tamaño 113,8 MB con un ancho de banda total de 100 Mbit/s. El archivo se envió desde tres *Talkers* hacia tres *Listeners*.

Existieron tres flujos, entonces, y un flujo en broadcast para saturar la red:

1. Un flujo con prioridad clase A, en el cual la transmisión completa del archivo de video llevó 302746 tramas

2. Otro con prioridad clase B, en cuyo caso la transmisión del video se dividió en 201831 tramas

3. Una transmisión mediante Ethernet sin AVB/TSN con un tamaño de trama de 430 B y 302746 tramas

4. Para la carga adicional de la red, se inyectó tráfico mediante una computadora a un switch específico con destino "FF:FF:FF:FF:FF:FF", para que el mismo envíe en broadcast tramas de tamaño 1500 B a cada dispositivo conectado

La diferencia en cantidad de tramas a pesar de ser el mismo video el que se transmite se debe a que AVB/TSN divide la información a enviar en tramas de menor tamaño para la prioridad A que para la prioridad B. Como el flujo de Ethernet simple intentaba imitar en lo posible el tráfico de clase A se utilizó un tamaño de trama similar y se envió misma cantidad. Para el tráfico de saturación, dado su objetivo de intentar colapsar la red, se eligió un tamaño de trama que más se acercara a la unidad máxima de transferencia (Maximum Transmission Unit - MTU, en inglés) de la capa de enlace.

3.4.2. Topologías Configuradas

Las topologías implementadas para realizar los experimentos contaron con seis dispositivos Apu2C4 y Apu2D4. Estas placas computadoras poseían un sistema operativo Ubuntu 16.04 GNU/Linux y tenían instalada la versión del repositorio OpenAvnu [39] correspondiente a la identificada bajo el código `6788f2f`. Los dispositivos se conectaron al switch mediante un cable Ethernet full duplex. La administración y el manejo de los Apus fue ejecutada por medio de Secure Shell (SSH). El acceso a los dispositivos era principalmente necesario para comenzar y finalizar las transmisiones, y para llevar a cabo las capturas de paquetes para su posterior análisis.

La configuración del escenario también requirió incluir un dispositivo extra, desde el cual se inyectaría tráfico para saturar la capacidad de los enlaces. Esta necesidad fue cubierta por medio de una computadora de escritorio conectada a uno de los swtiches. El tráfico inyectado desde esta computadora era específicamente tráfico mejor esfuerzo.

Topología Estrella

La figura 3.3 ilustra la topología de estrella implementada, la cual cuenta con seis dispositivos, todos conectados a un único switch.

Una vez formada la estructura, se comenzó con la transmisión de los tres flujos de tráfico distintos en simultáneo a lo que se le sumó el broadcast de tráfico de fondo para saturar la red.

La configuración crea un punto central para evaluar el desempeño del switch ante la presencia de congestión. Nuestro interés se centró en determinar si esta topología con un único punto de falla era capaz de funcionar en escenarios de saturación de tráfico de mejor esfuerzo.

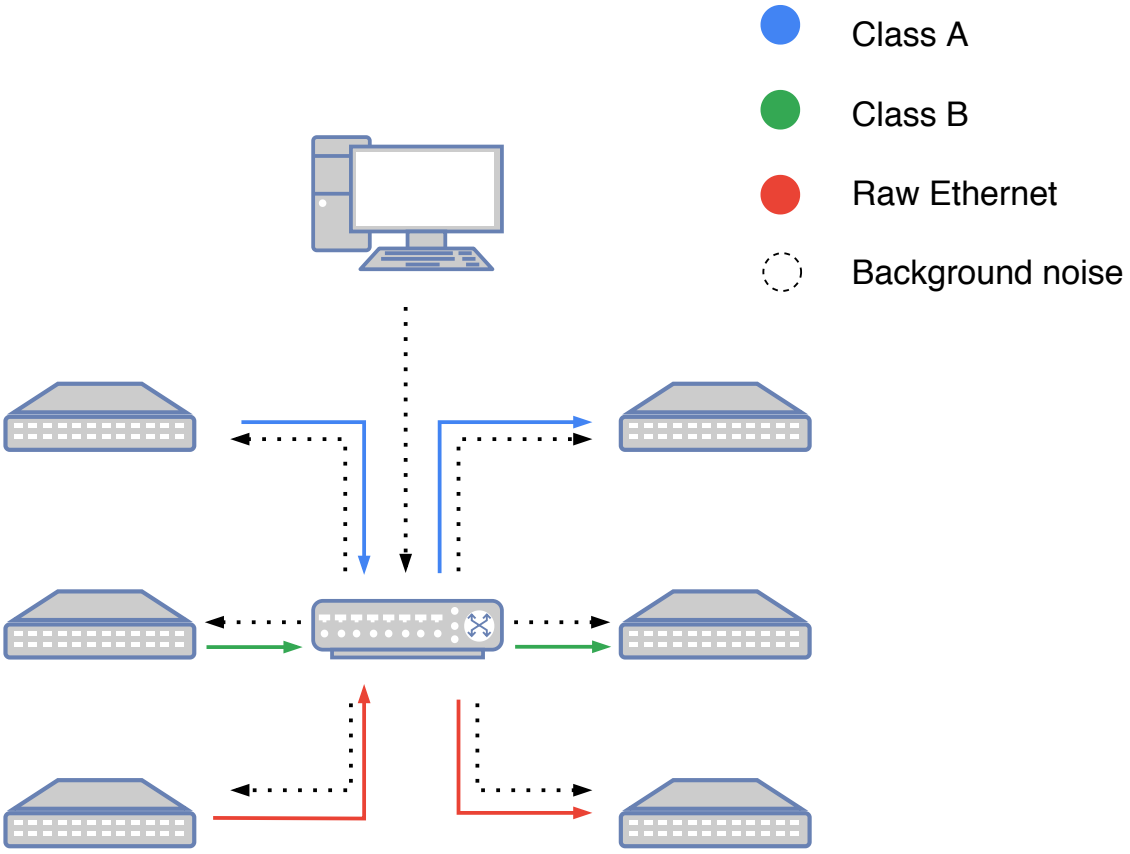


Figura 3.3: Topología Estrella. Esta disposición consiste de seis dispositivos conectados a un switch central. El desafío principal es la generación de un punto de congestión por donde todo el tráfico tiene que viajar.

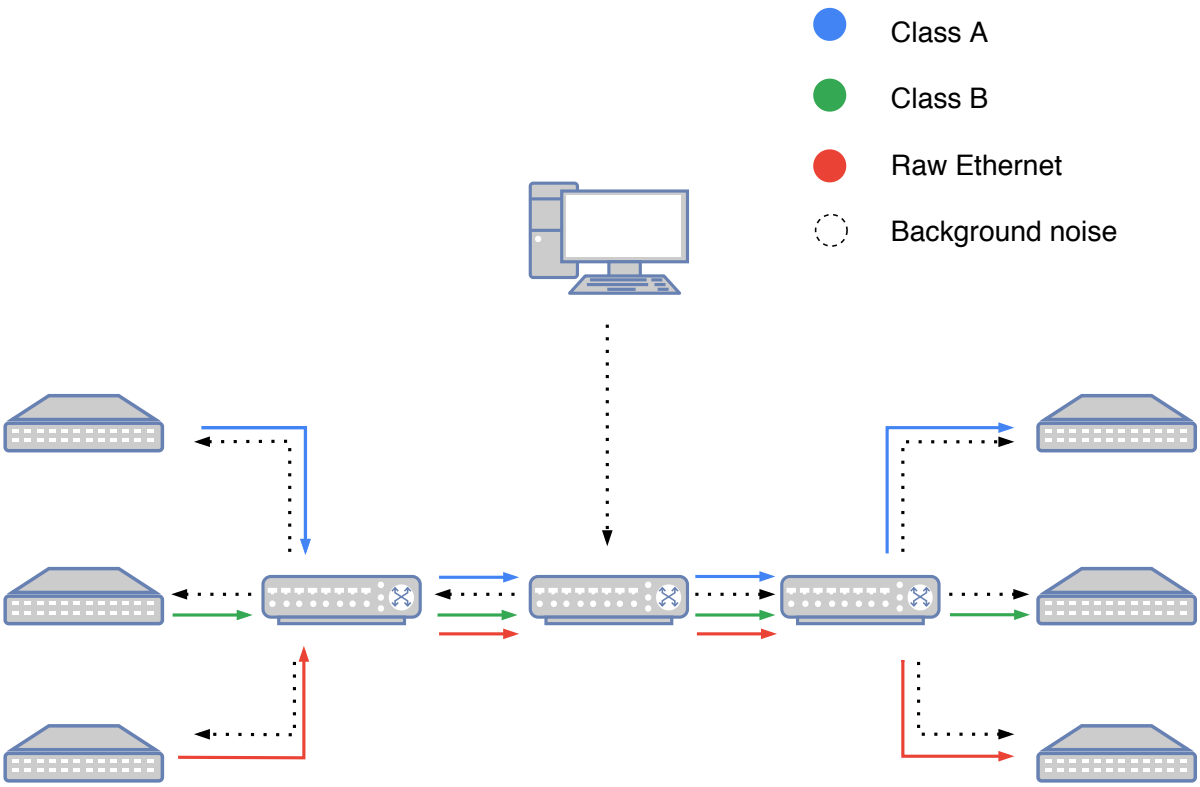


Figura 3.4: Topología Cadena. Esta disposición consiste de seis dispositivos y tres switches. Se agregan dos saltos más por donde debe fluir el tráfico. Además, el switch ubicado en el medio crea un punto de congestión en donde los tres tipos de tráfico sumado al tráfico de saturación de la red deben ingresar por un mismo puerto y salir por otro único puerto.

Topología Cadena

En la Figura 3.4, se agregan dos switches más con respecto a la topología anterior. Dos están conectados a tres dispositivos cada uno, mientras que el tercero se ubica entre los demás switches para llevar la información entre ellos. Una computadora conectada al tercer switch inyecta el tráfico de saturación a la red.

La motivación de esta configuración fue trabajar con más switches, creando más puntos de congestión y aumentando la cantidad de saltos por los que tiene que pasar el tráfico desde un *Talker* a un *Listener*.

Topología Anillo

Como puede observarse en la Figura 3.5, para crear esta topología se conectaron los tres switches entre ellos en forma de anillo y se unieron dos dispositivos a cada uno.

Esta disposición genera que el tráfico fluya por dos switches para llegar a destino.

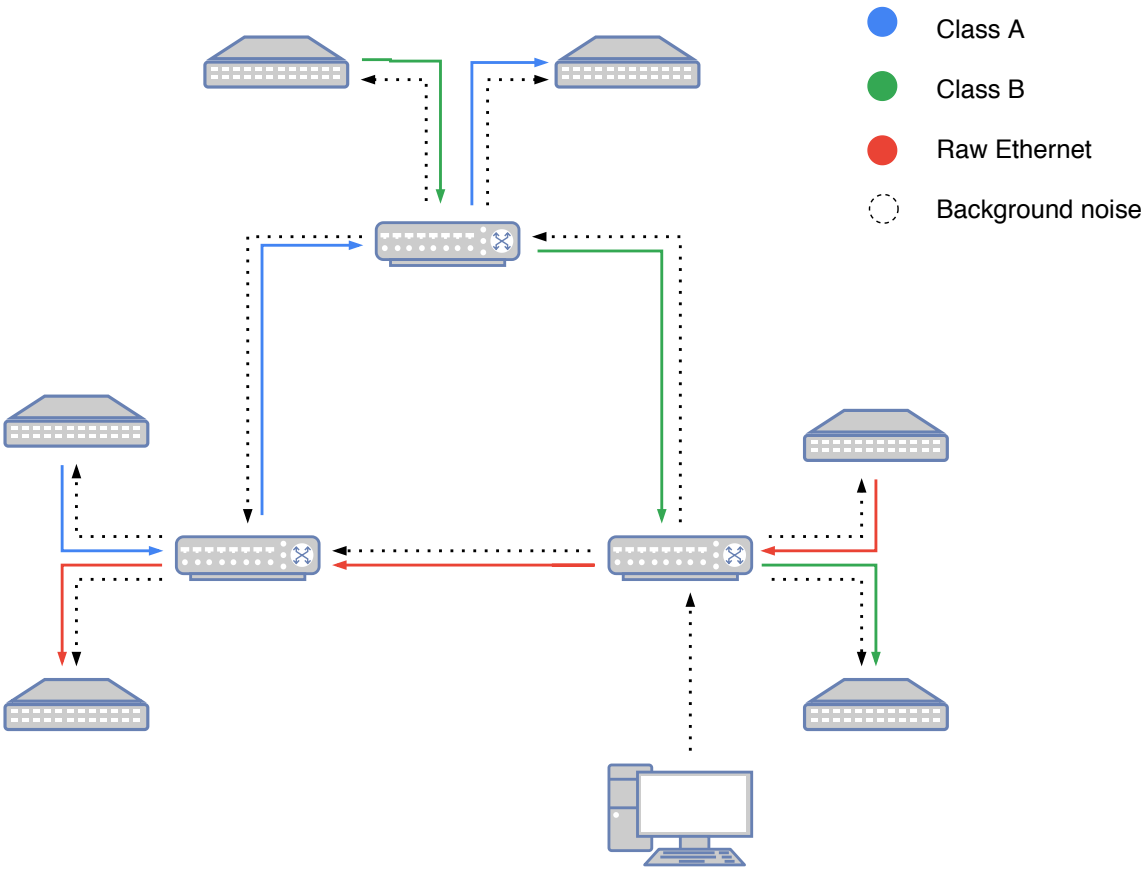


Figura 3.5: Topología Anillo. En esta configuración se conectan los tres switches entre ellos y además se le agregan dos dispositivos a cada uno. Las conexiones se hicieron tales que el tráfico tuviera que fluir por dos switches para llegar a destino. El interés principal de esta topología consiste en su estructura cíclica.

El punto interesante de esta configuración fue observar si el protocolo es efectivo en estructuras con ciclos. Esta configuración se había previamente probado en las simulaciones realizadas por medio de OMNeT++ y no había dado resultados satisfactorios. De esta manera, el interés se centró en estudiar cómo se iba a comportar el protocolo en un contexto real con las mismas características.

3.4.3. Resultados Experimentales

A continuación se presentarán los resultados obtenidos en los experimentos realizados con las diferentes configuraciones. Cambiar las topologías nos permite agregar más saltos y obstáculos al protocolo y probar si la demora, jitter y pérdida de paquetes se mantienen iguales. Diferentes topologías implican diferente demora de un extremo a otro, ya que el número de saltos se modifica. Nuestro objetivo fue verificar si AVB/TSN mantiene su garantía de demora máxima de (2 ms para tráfico clase A y 50 ms para clase B) con distinto número de saltos. Además, se realizaron varias corridas modificando el porcentaje de tráfico de saturación inyectado para analizar cómo actuaba el protocolo según la congestión de la red.
Mediante el programa de recolección de paquetes `tcpdump` [47], se capturaron los mensajes que se transmitían durante los experimentos. Luego, las muestras obtenidas fueron almacenadas en archivos de extensión ***pcap***. El programa se ejecutaba tanto en los Talkers como en los Listeners para almacenar tanto los paquetes enviados como los recibidos. Una vez finalizada la transmisión, se calculaba la diferencia entre el timestamp de una trama al llegar a destino con su timestamp del momento en el que egresó de su origen. Asimismo, al tener un registro de las tramas enviadas por cada talker, podíamos detectar las perdidas al no encontrarlas en el archivo ***pcap*** del destino.

Además, se implementaron programas en Python que recibían los archivos de tipo *pcap* y analizaban la información de los paquetes filtrando sólo los datos que considerábamos relevantes para las métricas que queríamos obtener. Esta información se escribía en archivos separados por comas (Comma Separated Values, CSV) cuyos datos eran posteriormente analizados mediante otro programa de Python. Este análisis consistía en realizar los cálculos pertinentes para obtener la demora, jitter y pérdida de paquetes. Para la manipulación de datos recolectados se utilizó la biblioteca `pandas` para Python y, para realizar gráficos que reflejaran los resultados obtenidos se usaron las bibliotecas `matplotlib` y `seaborn`.

Paquete transmitido

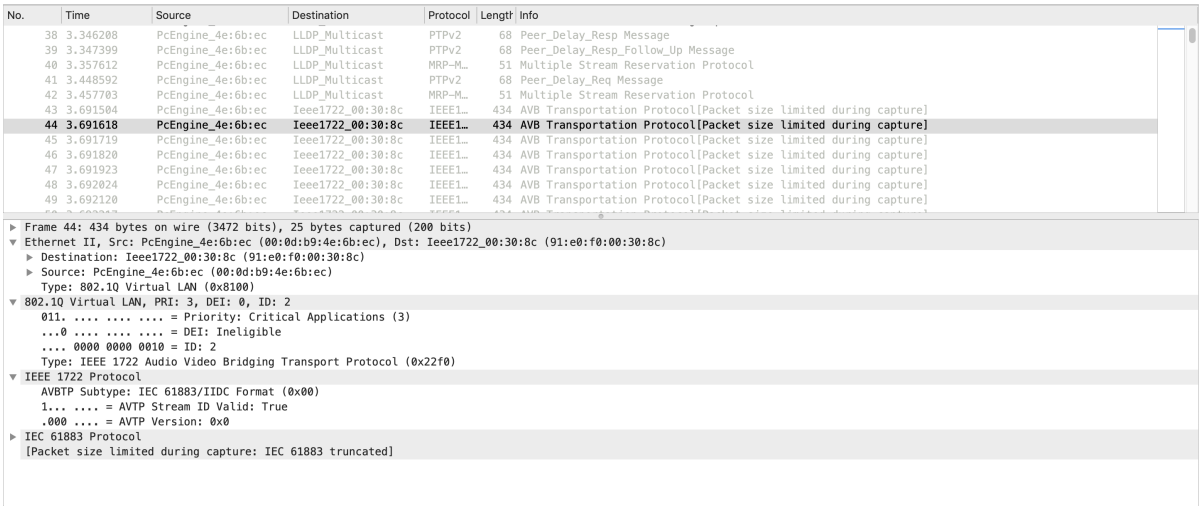


Figura 3.6: Captura de pantalla de el análisis en Wireshark de una trama de clase A transmitida en uno de los experimentos. Pueden visualizarse varios campos que representan estándares del protocolo, como la dirección de destino conseguida por MAAP, la prioridad PCP según el protocolo IEEE 802.1Q y el ID de la VLAN a la que corresponde.

En la figura 3.6 se muestra un paquete capturado por `tcpdump`. Esta captura nos permite observar los distintos mensajes que se envían para garantizar que los nodos de la red sean aptos para una transferencia que cumpla las garantías del protocolo. El tamaño de captura fue de 25 B para disminuir el tamaño de los archivos.

En esta captura se encuentran tanto los mensajes de control y sincronización del protocolo como los mensajes de flujo clase A que se enviaron. La figura muestra los mensajes *Peer Delay Response*, *Peer Delay Follow Up* y *Peer Delay Request* que se utilizaron para sincronizar los dispositivos según el protocolo gPTP . Además, se ven los mensajes que conforman el protocolo Stream Reservation Protocol (SRP), como los de tipo Multiple Stream Registration Protocol (MSRP) que funcionan para analizar si los dispositivos involucrados en la red cumplen las condiciones necesarias para una transmisión de tipo AVB/TSN .

Por último, se muestran los detalles de un paquete que transmite un flujo AVB/TSN de tipo clase A de información. Se indica que la trama pertence al protocolo IEEE1722 Audio Video Bridging Transport Protocol (AVBTP, también llamado AVTP). Es posible notar el header 802.1Q, que include el VLAN identifier (VID), Drop eligible indicator (DEI) y Priority code point (PCP). Esta trama forma parte de una VLAN con número de identificación 2, posee una prioridad PCP 3, destinada a aplicaciones críticas, correspondiente con la clase de tráfico A a la que pertenece y su bit DEI la categoriza como *Ineligible*, lo que implica que no debe ser omitida en caso de alta congestión en la red. La dirección MAC desde donde se envía la trama pertenece al dispositivo que

funcionó como Talker en ese flujo de información. Puede verse que el destino del mensaje es hacia `91:e0:f0:00:30:8c`, que es la dirección asignada dinámicamente por el MAC address Acquisition Protocol (MAAP). MAAP especifica un mecanismo para asignar direcciones Multicast y Unicast MAC de manera dinámica en un rango específico de direcciones [58]. Por último, también pueden visualizarse campos correspondientes al protocolo *IEC 61883* el cual define un formato de datos para audio contenido en flujos IEEE 1722 [59].

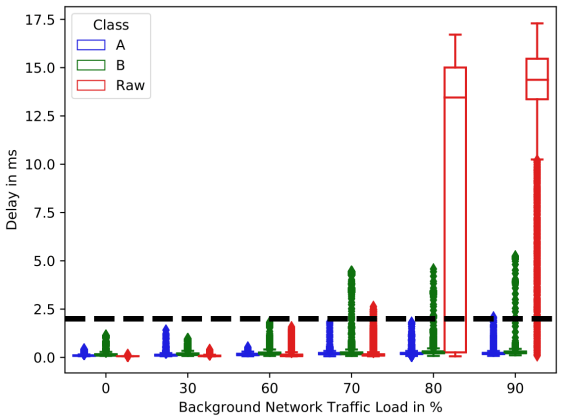
Topología Estrella

La figura 3.7a muestra un diagrama de cajas de la demora a medida que se incrementa el porcentaje de utilización de los enlaces. En este diagrama se presenta la demora de un extremo a otro para cada tipo de tráfico. Los bordes de cada caja representan el primer y el tercer cuartil de los resultados, la línea central muestra la mediana y los valores máximos y mínimos son los bigotes que sobresalen la caja. Además, se incluyen los valores atípicos en forma de puntos que sobresalen los máximos y mínimos de la caja. En el eje x se muestran los distintos porcentajes de tráfico de saturación que se inyectó en las múltiples corridas de los experimentos sobre esta topología. La línea punteada negra refleja la demora máxima de 2 ms asegurada por AVB/TSN para tráfico de clase A.

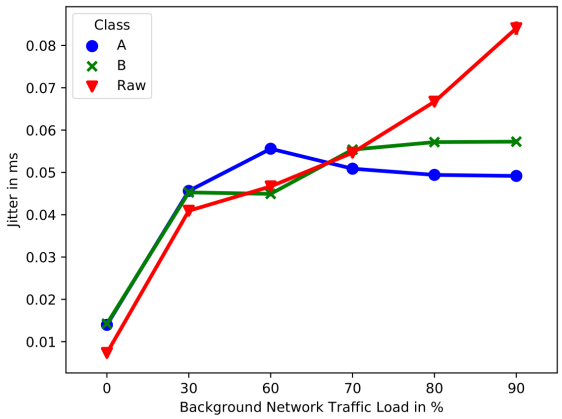
Puede notarse que la demora del tráfico de tipo AVB/TSN se mantiene constante a pesar de la congestión de la red. Las garantías de un máximo de demora de 2 ms para tráfico de clase A y 50 ms para tráfico de clase B se mantuvieron en los casos reflejados en el diagrama.

Por otro lado, las transmisiones hechas utilizando Ethernet (señalizado como Raw en la figura) incrementan significativamente su demora a 80 % de tráfico de saturación. Ethernet no realiza ningún tipo de reserva de recursos, por lo tanto, tiene que competir con el tráfico de saturación por el ancho de banda disponible. En el caso de AVB/TSN , como el estándar SRP se encarga de reservar los recursos necesarios para asegurar la transmisión efectiva del tráfico, establece y reserva un canal para que el mismo pueda fluir sin problemas. Por consiguiente, se reserva una cantidad específica de ancho de banda para tráfico AVB/TSN que no será compartida con otro tipo.

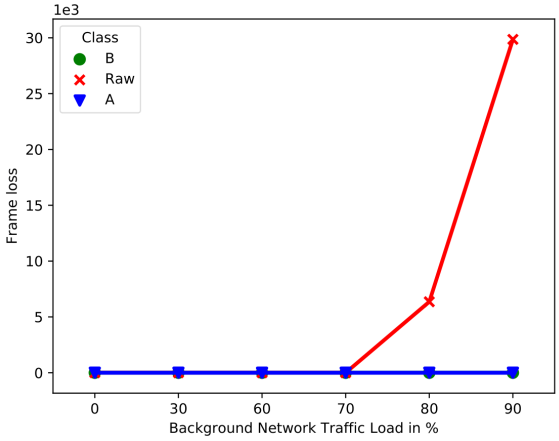
Otro detalle a notar, es la diferencia de demora entre clase A y clase B. El tráfico de clase B presenta una demora mayor que clase A en la mayoría de los casos, lo que es consistente con la prioridad menor que se le asigna. El algoritmo CBS reconoce la prioridad mayor de clase A y asigna los créditos de la forma correspondiente para que la clase A sea trasmitida primero.



(a) Diagrama de caja. En este diagrama se presenta la demora de un extremo a otro para cada tipo de tráfico. La línea punteada negra refleja la demora máxima asegurada por AVB/TSN para tráfico de clase A.

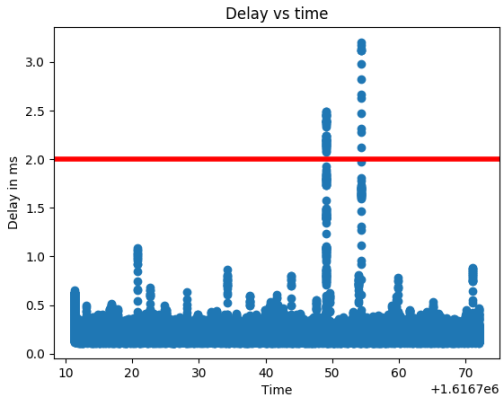


(b) Variación de jitter en milisegundos de tráfico AVB/TSN y tráfico Ethernet (Raw) para distintos porcentajes de tráfico de saturación inyectado. Los resultados fueron tomados con un intervalo de confianza de 95 %.

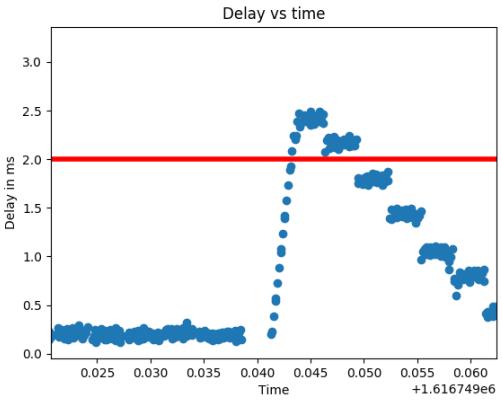


(c) Pérdida de paquetes durante las transmisiones para tráfico de clase A, B y Ethernet (Raw) para distintos porcentajes de tráfico de saturación inyectado.

Figura 3.7: Diagramas para la demora, jitter y pérdida de paquetes para la topología Estrella. En los tres escenarios se visualiza que el tráfico AVB/TSN cumple con las garantías que ofrece, mientras que Ethernet sufre alteraciones significativas a nivel alto de congestión de la red.



(a) Diagrama de demora vs tiempo de una transmisión de tráfico clase A en la topología Estrella con 85 % de tráfico de saturación.



(b) Vista en detalle de uno de los picos del diagrama de demora vs tiempo de la figura 3.8a.

Figura 3.8: Estas imágenes muestran la demora de cada trama durante el tiempo de duración de la transmisión. Se señala con una línea roja el límite de 2 ms garantizado para el tráfico de esta clase por AVB/TSN . Puede observarse que ciertas tramas no cumplieron con esta garantía y tuvieron una demora de un extremo a otro mayor. La figura 3.8b grafica cómo la demora de cada trama incrementa rápidamente hasta que llega a un valor máximo y luego decrece lentamente y en escalones.

Si bien mayoritariamente el tráfico de clase A cumple con su garantía de demora menor a 2 ms, existieron casos en donde este límite no fue cumplido. Uno de estos se refleja en la figura 3.8a, en donde se muestra un diagrama de Demora vs Tiempo de una transmisión de tráfico de clase A realizada en la topología Estrella con un 85 % de tráfico de saturación.

El eje x marca el tiempo transcurrido en la transmisión y el eje y la demora de cada trama transmitida durante ese tiempo. Se señala con una línea roja el límite de 2 ms garantizado para el tráfico de esta clase por AVB/TSN .

Se observa en la figura que en ciertos momentos de la transmisión existen picos de demora en donde las tramas tardan más del límite garantizado por AVB/TSN .

En la figura 3.8b se muestra una vista detallada y acercada de uno de los picos que sobrepasan el límite. Como puede observarse, la demora de cada trama incrementa rápidamente hasta que llega a un valor máximo y luego decrece lentamente y en escalones.

Si bien, por una cuestión de tiempos, no nos fue posible ahondar más en la investigación de este caso, sería interesante entender las causas del mismo como trabajo futuro. El banco de pruebas realizado en esta tesis presenta un escenario en donde se podrían realizar estas futuras investigaciones, ya que al ser de código abierto, los parámetros del protocolo son fácilmente modificables.

En cuanto a la segunda métrica que nos interesaba analizar, el jitter, la figura 3.7b

muestra los resultados que obtuvimos con esta topología. Se visualiza que el desempeño de ambos protocolos es similar hasta alcanzar un 70 % de tráfico de saturación. A partir de ese valor, Ethernet comienza a incrementar de manera creciente, mientras que el tráfico clase A y B alcanzan una meseta.

Esta figura muestra valores esperables según las garantías que ofrece el protocolo AVB/TSN , asegurando consistencia y estabilidad para clases de prioridad alta. De esta manera, como Ethernet no realiza la reserva de recursos necesaria para cumplir estas garantías, sus resultados no son los mismos.

La última métrica a analizar para esta topología es la pérdida de paquetes. En la figura 3.7c se observa la cantidad de paquetes perdidos para cada transmisión realizada con distinto porcentaje de tráfico de saturación. Se muestran tres tipos de tráfico: clase A, clase B y mejor esfuerzo transmitido mediante Ethernet.

Tanto el tráfico clase A como el B no sufren ninguna pérdida durante las transmisiones. Este comportamiento está en concordancia con las garantías de calidad de servicio que promete el protocolo AVB/TSN . El algoritmo SRP verifica que los dispositivos involucrados en la comunicación sean capaces de transmitir las tramas necesarias y poseen los recursos suficientes para garantizar la calidad de servicio. Una vez detectados los nodos de la red que poseen estas características, crea el dominio SRP compuesto solamente por los nodos aptos para cumplir con las transmisiones a realizar. El protocolo reserva una cola de alta prioridad por donde viajará el tráfico y el algoritmo CBS reconoce que parte del ancho de banda está reservado exclusivamente para el flujo de datos de AVB/TSN . En consecuencia, tanto para clase A como clase B todas las tramas son transmitidas desde un extremo a otro.

Por otro lado, Ethernet no ofrece estas garantías por default y, de esta manera, algunas tramas son descartadas una vez que el tráfico de saturación aumenta lo suficiente. Este comportamiento se debe a que Ethernet usa la misma cola de tráfico de mejor esfuerzo que el tráfico de saturación y, entonces, cuando la misma comienza a llenarse, el switch debe descartar tramas que exceden la capacidad. Al mismo tiempo, cada trama en la cola debe esperar una cantidad de tiempo linealmente creciente. Dado que en AVB/TSN , la situación no es la misma, en una red congestionada sólo el tráfico de mejor esfuerzo es descartado.

En conclusión, el switch logra priorizar las clases de tráfico con más alta prioridad y distribuir el ancho de banda de forma favorable para los flujos de datos de AVB/TSN . A la par, mediante la gestión de créditos que maneja el algoritmo CBS evita que el tráfico de mejor esfuerzo sufra de inanición, asegurándose que el mismo sea eventualmente transmitido. Esto se logra gracias al uso de los estándares que componen AVB/TSN : como lo son gPTP, SRP y CBS.

Topología Cadena

Cuando se agregaron nuevos switches a la estructura, nos enfrentamos con errores de sincronización con los que antes no habíamos lidiado. Entre los nodos de la red no era posible establecer un único Reloj Maestro a partir del cual todos pudieran sincronizar sus relojes internos. Algunos dispositivos elegían uno de los switches como Reloj Maestro, mientras que otros elegían otro. Esto provocaba que los relojes de uno de los grupos de dispositivos no pudieran sincronizarse con los del otro grupo. El conflicto se debió a que los switches no se estaban sincronizando entre ellos, por lo que cada switch se elegía a sí mismo como Reloj Maestro.

Al investigar con profundidad las características y configuraciones de los switches, detectamos que uno de sus parámetros, el *Neighbor Propagation Delay*, no era lo suficientemente grande como para aceptar mensajes de otro switch. Este parámetro define un límite de tiempo a partir del cual si un mensaje recibido llega con una demora mayor, es ignorado. Dado que la demora de un mensaje proveniente de otro switch sobrepasaba el umbral, al llegar al puerto no se lo consideraba apto para participar del dominio gPTP establecido por el protocolo 802.1AS y era descartado. Este protocolo es uno de los estándares que componen AVB/TSN y se encarga de asegurar que los miembros de la red cumplan los requerimientos de sincronización necesarios para aplicaciones tiempo sensibles. Por consiguiente, el switch no podía sincronizarse con los demás.

Para solucionar este inconveniente, aumentamos el límite máximo del umbral de demora de los switches a un valor que permita el ingreso de otros switches al dominio AVB. Debido a que al aumento no era significativo, las garantías de demora de AVB/TSN no se vieron afectadas. Una vez hecho este cambio, los switches comenzaron a aceptar los mensajes de los demás y pudieron seleccionar un único Reloj Maestro y sincronizarse con el mismo.

En la figura 3.9a, se ilustra el diagrama de cajas correspondiente a la demora de los tres tipos de tráfico transmitidos para distintos porcentajes de tráfico de saturación. El tráfico de mejor esfuerzo transmitido mediante Ethernet supera la barrera de los 2 ms a un porcentaje menor de congestión que en la topología anterior. Esto se debe a que hay una mayor cantidad de saltos entre un extremo y otro, lo que es una desventaja para el tráfico de mejor esfuerzo. Dado que hay tres switches, existen tres puntos de congestión donde el tráfico se acumula y, debido a los algoritmos CBS y SRP, AVB/TSN se asegura de tener prioridad al pasar por estos puntos. En consecuencia, tanto el tráfico que viaja mediante Ethernet como el tráfico de saturación, al tener menor prioridad, deben esperar para ser transmitidos por ese enlace, aumentando así su demora.

En el diagrama mostrado en la figura 3.9b, se grafica el jitter de los tres tipos de tráfico

según el porcentaje de congestión de la red. En este caso, el jitter de la transmisión por Ethernet supera el de clase A y B apenas cuando se inyecta 30 % de tráfico de saturación a la red. Esto se debe a la misma razón que en el caso anterior, dado que hay más saltos entre *talker* y *listener* y, de esta manera, más puntos de congestión, Ethernet pierde consistencia.

Siguiendo lo mencionado previamente y, observando los resultados reflejados en la figura 3.9c, sólo existe pérdida de paquetes en la transmisión mediante Ethernet. La pérdida comienza con 60 % de congestión de la misma forma que sucedió con el incremento en la demora y el jitter. El tráfico AVB/TSN no sufre pérdidas dado que SRP protege las tramas transmitidas mediante este protocolo.

Topología Anillo

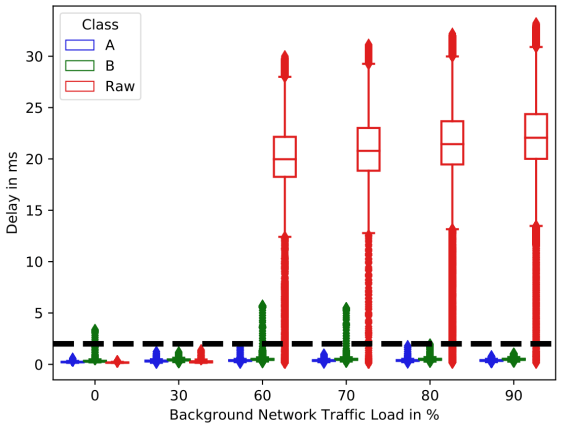
Los resultados obtenidos con esta topología muestran similitudes con los conseguidos en la topología Estrella.

Esto puede verse en el diagrama de cajas que representa la demora para cada tipo de tráfico de la figura 3.10a. El tráfico de clase A y B cumple las garantías de demora establecidas por el protocolo, mientras que Ethernet comienza a incrementar notablemente a partir de un 80 % de tráfico de saturación.

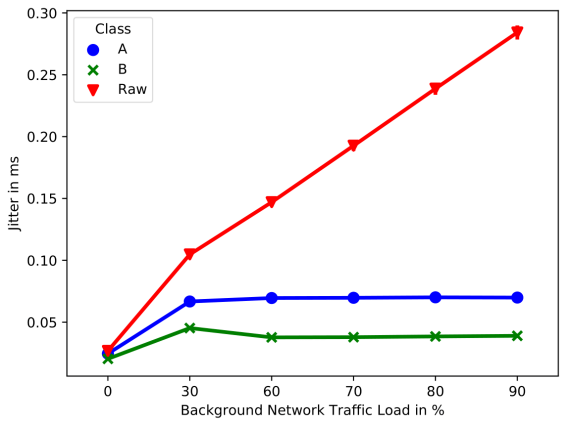
En cuanto al jitter, la figura 3.10b muestra valores que no se condicen con las garantías que podrían esperarse del tráfico AVB/TSN . Por un lado, para valores menores a 80 % de congestión, Ethernet muestra mayor consistencia que el tráfico de clase A y B. Por otro lado, los valores de jitter de clase A son mayores que los de clase B cuando la primera es de mayor prioridad.

Estas diferencias pueden ser consecuencias de varios factores. Por un lado, puede ser que los ciclos que posee esta topología hayan generado más trabajo para los algoritmos que componen AVB/TSN causando que pierdan consistencia. Asimismo, el protocolo AVB/TSN debe correr un conjunto de estándares previo a realizar la transmisión lo que puede causar que inicialmente este tipo de tráfico tenga un desempeño peor que Ethernet que no debe realizar reservas de recursos inicialmente si no que comienza a transmitir instantáneamente. Es importante considerar, también, que los resultados obtenidos son valores realmente bajos de jitter por lo que las diferencias en términos generales no llegan a ser significativas.

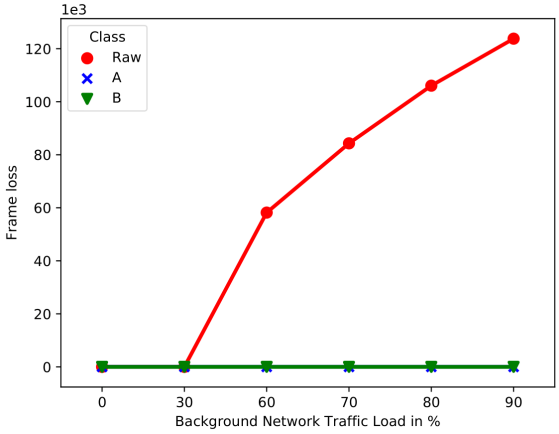
No obstante, a medida que se aumenta la congestión, el tráfico AVB/TSN alcanza una meseta, mientras que Ethernet incrementa crecientemente.



(a) Diagrama de caja para la topología Cadena. En este diagrama se presenta la demora de un extremo a otro para cada tipo de tráfico. La línea punteada negra refleja la demora máxima asegurada por AVB/TSN para tráfico de clase A.

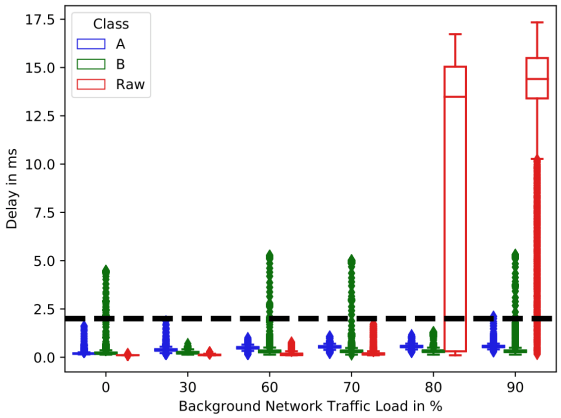


(b) Variación de jitter en milisegundos de tráfico AVB/TSN y tráfico Ethernet (Raw) para distintos porcentajes de tráfico de saturación inyectado. Los resultados fueron tomados con un intervalo de confianza de 95 %.

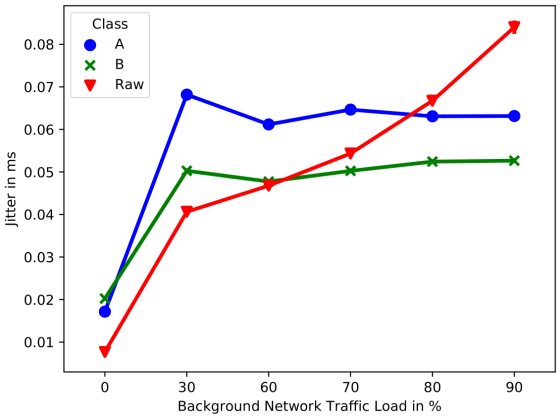


(c) Pérdida de paquetes durante las transmisiones para tráfico de clase A, B y Ethernet (Raw) para distintos porcentajes de tráfico de saturación inyectado.

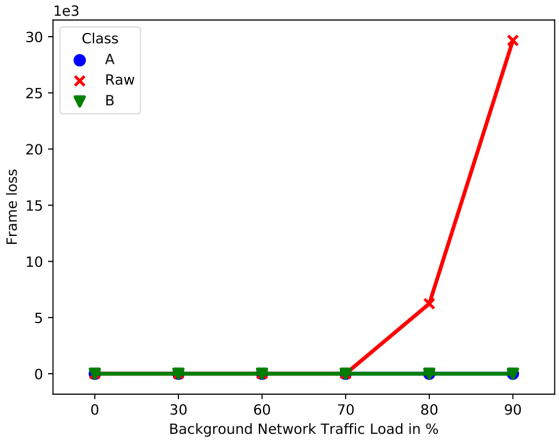
Figura 3.9: Diagramas para la demora, jitter y pérdida de paquetes para la topología Cadena. Para cualquier porcentaje de congestión de la red tanto el tráfico clase A como el clase B mantienen las garantías establecidas por el protocolo, mientras que a partir un porcentaje de 60 % de saturación, Ethernet (Raw) muestra aumentos significativos.



(a) Diagrama de caja para la topología Anillo. En este diagrama se presenta la demora de un extremo a otro para cada tipo de tráfico. La línea punteada negra refleja la demora máxima asegurada por AVB/TSN para tráfico de clase A.



(b) Variación de jitter en milisegundos de tráfico AVB/TSN y tráfico Ethernet (Raw) para distintos porcentajes de tráfico de saturación inyectado. Los resultados fueron tomados con un intervalo de confianza de 95 %.



(c) Pérdida de paquetes durante las transmisiones para tráfico de clase A, B y Ethernet (Raw) para distintos porcentajes de tráfico de saturación inyectado.

Figura 3.10: Diagramas para la demora, jitter y pérdida de paquetes para la topología Anillo. Para cualquier porcentaje de congestión de la red tanto el tráfico clase A como el clase B mantienen las garantías de demora establecidas por el protocolo, mientras que a partir de un cierto porcentaje de saturación, Ethernet (Raw) muestra aumentos significativos.

3.5. Experimentos con Credit Based Shaper

Esta sección profundiza la investigación de AVB/ TSN realizando experimentos sobre uno de los estándares que conforma este protocolo: *IEEE 802.1Qav*. El objetivo de esta investigación fue individualizar el comportamiento de una de las piezas esenciales de AVB/TSN y contrastarlo con resultados experimentales.

Una de las características principales de este estándar es el algoritmo de admisión de tráfico denominado Credit Based Shaper (CBS). Este algoritmo tiene como propósito regular la utilización de los enlaces, otorgando cuotas máximas de ancho de banda a diferentes categorías de tráfico. En particular, el CBS regula las asignaciones por medio de créditos otorgadas a cada una de las clases de tráfico definidas. De esta forma, busca asegurar la calidad de servicio necesaria para el tráfico de alta prioridad y, al mismo tiempo, no dejar que las clases de menor prioridad sufran de inanición. Tal como se expresa en el estándar oficial del protocolo *IEEE Std 802.1Q* [60], esto se logra deteniendo la transmisión del tráfico de una clase una vez que haya consumido todos sus créditos disponibles. En ese momento, se cede la transmisión de tráfico a otra clase.

Durante la sección se explorará el desempeño del algoritmo Credit Based Shaper (CBS) en el banco de pruebas siguiendo un modelo matemático planteado en el trabajo de investigación planteado por Cao *et al.* [19].

También la sección presentará cómo varía el tiempo de respuesta al modificar distintos parámetros de la clase de mayor prioridad, utilizando el artículo realizado por Cao *et al.* [19] como base para poder acotar el Peor Tiempo de Respuesta (PTR).

Esta experimentación permitirá evaluar la robustez del banco de pruebas, como también demostrará su flexibilidad examinando en profundidad varios aspectos de AVB/ TSN. Asimismo, se evidenciará su capacidad de funcionar como una herramienta para evaluar la veracidad de modelos matemáticos en un contexto de tiempo real.

3.5.1. Modelo matemático

A continuación presentaremos los resultados teóricos formulados por Cao *et al.* [19], los cuales luego contrastaremos con ensayos experimentales en el banco de pruebas.

Si bien en la publicación se demuestra la factibilidad del modelo matemático planteado, se hace mediante simulaciones y deducciones matemáticas. La motivación principal de esta investigación es demostrar la aplicabilidad y robustez del modelo por medio de resultados empíricos y no por simulaciones ni modelos teóricos, utilizando el banco de pruebas implementado en la presente tesis.

El artículo explora una nueva forma de analizar el peor tiempo de respuesta (PTR) para sistemas basados en prioridades. Basándose en la metodología denominada Eligible

Intervals (Intervalos elegibles, en español) se determinan intervalos de medición para una clase específica de tráfico mientras está siendo interferida tanto por tráfico de clase más alta o de prioridad más baja. Esta metodología de selección de intervalos, permite hallar una cota para el peor tiempo de respuesta para un flujo de información cuyas tramas se verán conformadas por el CBS. Esta cota se mantiene para situaciones en la que el flujo de datos puede sufrir interferencias causadas por la transmisión de tráfico de mayor prioridad, de menor o de ambas al mismo tiempo.

Las deducciones matemáticas que se realizan a lo largo del artículo llegan a la siguiente fórmula de peor tiempo de respuesta con interferencia:

$$WR_{int}(m) = WR_{no-int}(m) + C_L^{max}.(1 + \frac{\alpha_H^+}{\alpha_H^-}) + C_H^{max}$$

$WR_{int}(m)$ y $WR_{no-int}(m)$ representan el peor tiempo de respuesta con y sin interferencia, respectivamente , de una trama m de un flujo de prioridad media (M). Por otro lado, C_L^{max} y C_H^{max} se refieren al máximo tamaño de trama para la clase de menor prioridad (L) y la de mayor prioridad (H). Por último, α_H^+ es la pendiente de inactividad y α_H^- la pendiente de envío.

La propuesta de trabajo fue validar la cota que establece la ecuación en un entorno real, haciendo uso del banco de pruebas construido. De esta manera, realizamos transmisiones variando dos de los parámetros de la fórmula:

1. el máximo tamaño de trama de la clase H (C_H^{max})

2. la pendiente de inactividad de la clase H (α_H^+)

Los experimentos se realizaron observando la demora de tráfico de clase B, puesto que el mismo puede ser interferido tanto por tráfico de prioridad más alta, como es el de clase A, como de prioridad más baja como es el de mejor esfuerzo.

El análisis del artículo fue realizado limitando el alcance al tiempo que una trama permanece en el buffer del puerto de salida del switch al que ingresó. Para adaptar este alcance a nuestro contexto real, determinamos medir la demora de una trama entre que ingresa y egresa de un switch. A este tiempo lo consideramos como el tiempo de encolado.

3.5.2. Medición de tiempo de encolado

Comenzamos estudiando la fórmula de PTR ($WR_{int}(m)$) planteada en el modelo de Cao *et al.* [19]. Para medir el PTR, es necesario obtener el tiempo de encolado del switch. En otras palabras, se busca obtener la demora entre que una trama ingresa al switch y

egresa del mismo. En esta subsección se explica el método que utilizamos para medir el tiempo de encolado necesario para probar los modelos matemáticos definidos por Cao.

En los experimentos anteriores, creamos archivos *pcap* de registro con los paquetes que egresaban e ingresaban a un dispositivo. Estos archivos fueron generados utilizando el programa `tcpdump`, el cual para cada paquete capturado agrega el timestamp en el que el paquete egresó o ingresó al puerto. Posteriormente, calculábamos la diferencia entre ambos timestamps para obtener la demora de extremo a extremo.

Tanto en el talker como en el listener, era posible registrar el timestamp de las tramas dado que los dispositivos APU pueden correr un sistema operativo Ubuntu. Esto les brinda flexibilidad y permite, entre otras funcionalidades, ejecutar `tcpdump` desde sus consolas. El switch, por otro lado, era un instrumento que tenía un sistema operativo propio e inaccesible más que por la interfaz visual que ofrece. Al no poder correr `tcpdump` en el mismo, hubo que buscar otras alternativas.

De esta forma, decidimos optar por un mecanismo del switch llamado Puerto Espejo (Port Mirroring, en inglés). Este mecanismo consiste en enviar una copia de los paquetes recibidos en un puerto a una conexión en otro puerto del switch. A través de este método conectamos dos dispositivos APU a los puertos que funcionaban como espejo y en los mismos ejecutamos `tcpdump`. Esto permite registrar tanto el instante en el que la trama ingresa al switch como en el que egresa del mismo. La diferencia entre estos dos timestamps lo consideramos en nuestros cálculos como el tiempo de encolado.

Para comprobar que esta forma de medir fuera válida para cumplir con los objetivos planteados, realizamos comparaciones entre las medidas realizadas con el puerto espejo y las medidas de un extremo a otro. La configuración mostrada en la figura 3.11 fue realizada para evaluar la validez del método de medición. Para visualizar el funcionamiento de CBS, se conectó un switch a otro y se generó un punto de congestión en donde los tres tipos de tráfico tuvieran que competir por ser transmitidos. El puerto conectado al segundo switch, de esta manera, es el punto de mayor aglomeración de la red. Se conectan tres dispositivos a cada switch para poder transmitir tres tipos de tráfico: clase A, clase B y mejor esfuerzo.

La figura 3.12a muestra la suma de las mediciones del tiempo de encolado de ambos switches. Estos valores están señalizados como *Switches*. Por otro lado, marcado como *End to end*, se grafica la demora de un extremo a otro de las mismas tramas.

Para verificar que el método de medición elegido diera resultados confiables, tomamos la siguiente hipótesis:

> *La suma del tiempo promedio de las tramas dentro de los switches no debe superar el tiempo para ser transferidas de un extremo a otro.*

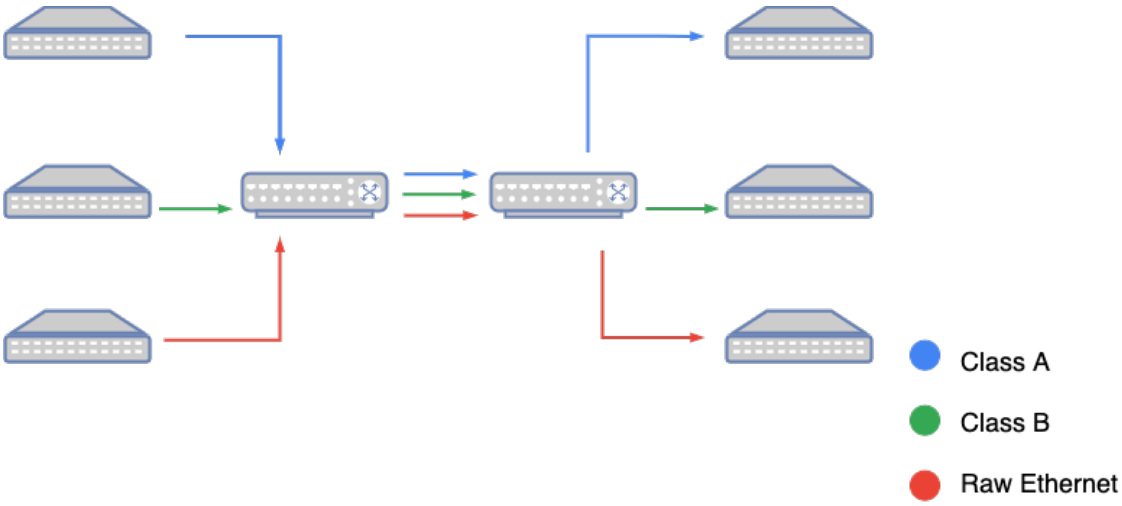
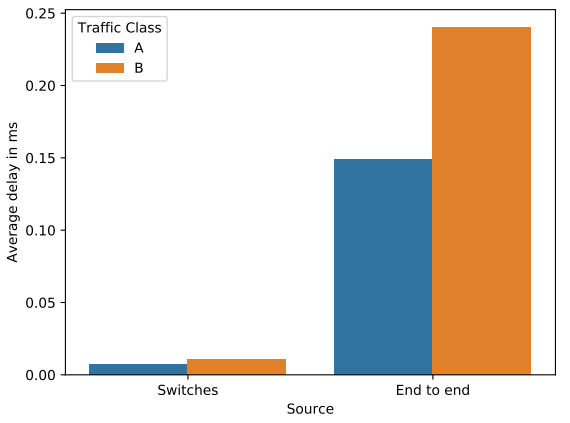
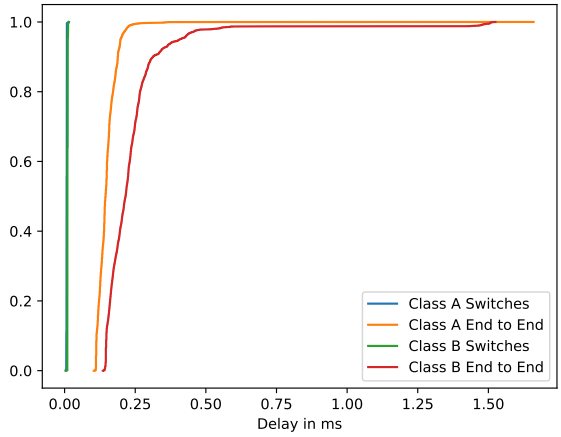


Figura 3.11: Topología realizada para validar el método escogido para obtener el tiempo de encolado dentro del switch. El puerto conectado al segundo switch, de esta manera, es el punto de mayor aglomeración de la red. Se conectan tres dispositivos a cada switch para poder transmitir tres tipos de tráfico: clase A, clase B y mejor esfuerzo.



(a) Demora promedio de una trama dentro del switch comparada con la demora de un extremo a otro.



(b) Gráfico ECDF de la demora de una trama de clase A y una de clase B dentro del switch y desde un extremo a otro.

Figura 3.12: Comparación entre la demora capturada con el método del puerto espejo vs la demora calculada desde un extremo a otro. Mediante los mismos, se concluye que el método del puerto espejo da resultados más certeros que sellando la trama a cada extremo ya que reduce el tiempo extra del viaje de la trama desde el switch al dispositivo.

Si esta hipótesis no se cumplía, nuestro método no era válido.

Ejecutamos varias transmisiones y tomamos las mediciones de ambos casos. Luego calculamos el promedio de estas mediciones y comparamos resultados. Como puede visualizarse en las figuras 3.12a, la demora de un extremo a otro fue mayor que la suma de la demora de la tramas dentro de los switches.

En la figura 3.12b se grafica la distribución empírica de la demora en milisegundos de flujos de tráfico de clase A y B medida desde un extremo a otro y adentro del switch. Se visualiza que para todos los valores obtenidos, que representan la demora de cada trama transmitida, los resultados de las mediciones realizadas dentro del switch son menores que desde un extremo a otro.

En consecuencia, como tanto para el promedio como para el valor de cada trama la demora fue mayor extremo a extremo, consideramos el método de medición como una forma válida de medir el tiempo de encolado y proseguimos a realizar los experimentos posteriores en base a esto.

3.5.3. Contraste del modelo analítico en el entorno real

El objetivo de los experimentos realizados fue validar en el banco de pruebas construido la cota que establece la ecuación:

$$WR_{int}(m) = WR_{no-int}(m) + C_L^{max}.(1 + \frac{\alpha_H^+}{\alpha_H^-}) + C_H^{max}$$

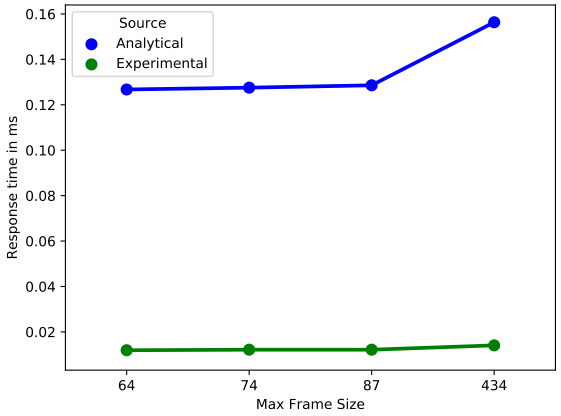
Para esto, se decidió realizar transmisiones variando dos parámetros de la ecuación:

1. el máximo tamaño de trama de la clase H (C_H^{max})
2. la pendiente de inactividad de la clase H (α_H^+)

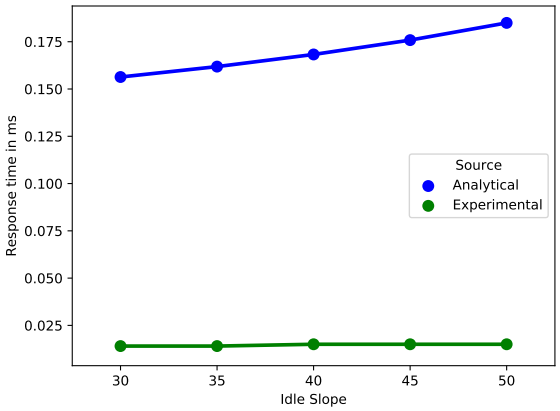
El algoritmo CBS no interrumpe la transmisión de una trama cuando la misma está siendo transmitida, si no que aguarda a que esta finalice para darle la prioridad a otro tipo de tráfico. De esta manera, el tamaño máximo de las tramas de tráfico de prioridades distintas a M determinarán el tiempo en que M no podrá transmitir.

A su vez, la pendiente de inactividad determina la velocidad en la que aumentan los créditos de una clase. Dado que el tráfico puede ser transmitido solamente cuando tiene crédito positivo, la velocidad en la que acumula créditos es fundamental para la demora que va a sufrir una trama de otro tipo de tráfico. La pendiente de inactividad es determinada por el ancho de banda que se destina para una clase de tráfico en particular.

Para modificar la pendiente de inactividad, el switch Netgear GS716Tv3 nos brinda una interfaz para seleccionar el ancho de banda destinado para cada clase de tráfico A



(a) Tiempo de respuesta en milisegundos de tráfico de clase B. En las cuatro transmisiones distintas se varía el tamaño máximo de trama para el tráfico de clase A que se transmite en simultáneo.



(b) Tiempo de respuesta en milisegundos de tráfico de clase B. En las cuatro transmisiones distintas se varía la pendiente de inactividad para el tráfico de clase A que fue transmitido en simultáneo.

Figura 3.13: Comparación entre el tiempo de respuesta de transmisiones experimentales de tráfico de clase B con el peor tiempo de respuesta según la fórmula del artículo de Cao *et al.*. Estas transmisiones pueden ser interferidas tanto por tráfico de clase A como de mejor esfuerzo. Se varían dos parámetros del tráfico de clase A: el tamaño máximo de las tramas y la pendiente de inactividad. Los resultados analíticos acotan los experimentales demostrando efectivamente que la fórmula devuelve el peor tiempo de respuesta posible.

---

y B. Por otro lado, para variar el tamaño máximo de trama, transmitimos archivos de audio de distintos tamaños.

La figura 3.13a muestra el tiempo de respuesta para transmisiones de tráfico de clase B. En simultáneo se transmitió tráfico de prioridad más alta, clase A, y de más baja, mejor esfuerzo. En cada transmisión se modificó el valor máximo de trama para el tráfico de prioridad más alta. Los valores experimentales se muestran en verde y son los obtenidos midiendo el tiempo de encolado de las tramas al ser transmitidas por el switch. Los valores en azul son el resultado de la ecuación formulada en el artículo, reemplazando los parámetros de la misma por los valores que se utilizaron durante la transmisión. Para todos los casos los resultados analíticos presentan una cota para los resultados experimentales.

De la misma manera, en la figura 3.13b se reflejan los tiempos de respuesta de tráfico de clase B transmitido junto a clase A y mejor esfuerzo. En este caso, se varió el valor de la pendiente de inactividad del tráfico de mayor prioridad. Los resultados experimentales se asemejan entre ellos, manteniendo el tiempo de respuesta en valores menores a 0,025 ms. En los resultados analíticos, en cambio, el tiempo de respuesta incrementa proporcionalmente con el aumento en pendiente de inactividad. En este caso, los resultados

experimentales vuelven a estar acotados por los analíticos.

Por consiguiente, no encontramos en estos experimentos evidencia que pueda contradecir los resultados obtenidos en el artículo escrito por Cao *et al.*. Esta investigación, por lo tanto, refuerza los puntos demostrados en el artículo, pero esta vez no desde un entorno virtual y simulado, si no desde una perspectiva física, con dispositivos y configuraciones reales.

3.6. Conclusiones

Se comprobó que en las distintas topologías implementadas, AVB/TSN mantiene las garantías de demora que asegura para las clases de mayor prioridad. Por el otro lado, el tráfico de mejor esfuerzo transmitido por Ethernet sufre cambios abruptos de demora y sufre pérdida de paquetes cuando hay un alto nivel de congestión en la red. Al obtener estos resultados en el banco de pruebas implementado se comprueba que el mismo es apto para funcionar como plataforma para futuras investigaciones sobre el protocolo.

Asimismo, posibilitó la evaluación del modelo teórico presentado en el trabajo de investigación de Cao *et al.*.

En conclusión, el banco de pruebas provee un entorno real, de código abierto y de sencilla implementación que flexibiliza el desarrollo de nuevas soluciones y alternativas, facilitando las iteraciones de prueba de las mismas.

Capítulo 4

Conclusiones

En este trabajo de investigación, se halló que si bien CoRE4INET es el simulador más completo para la experimentación que se llevó a cabo, no existe un flujo armónico entre el simulador y las experimentaciones en el entorno físico. Esto se evidencia, principalmente, en el desempeño de las pruebas realizadas en la topología anillo, las cuales no funcionaron de manera satisfactoria en el simulador pero sí en el entorno real. De esta forma, existen todavía oportunidades de mejora en el software de simulación utilizado para obtener mayor integración entre los resultados simulados y los reales.

La experiencia con el banco de pruebas demostró que es posible ensamblar un entorno de investigación de código abierto para Ethernet Automotriz utilizando dispositivos de fácil alcance y bajo costo. Mediante experimentos con diferentes topologías, se logró calibrar el mismo, obteniendo resultados acordes a las garantías que se esperan de un protocolo para tráfico de alta criticidad como AVB/TSN.

Asimismo, se profundizó el entendimiento del protocolo a través del banco de pruebas analizando el algoritmo principal de uno de los estándares que lo conforman: Credit Based Shaper. Probando también, que el banco disponibiliza un entorno de pruebas en donde es posible validar de manera física modelos analíticos.

En conclusión, se comprueba que el diseño e implementación de un banco de pruebas para probar AVB/TSN en un contexto institucional es factible. Se abre, así, el camino a nuevas investigaciones que puedan llevar a la mejora y expansión del protocolo. Poseer un entorno de experimentación de código abierto y accesible flexibiliza el desarrollo de nuevas alternativas, facilitando las iteraciones de prueba para crear soluciones más robustas y confiables.

Bibliografía

[1] T. Andrea Xu, F. Adamsky, I. Turcanu, R. Soua, C. Köbel, T. Engel y A. Baiocchi, «Poster: Performance Evaluation of an Open-Source Audio-Video Bridging/Time-Sensitive Networking Testbed for Automotive Ethernet», dic. de 2018, págs. 1-2.

[2] J. Cao, P. J. L. Cuijpers, R. J. Bril y J. J. Lukkien, «Tight worst-case response-time analysis for ethernet AVB using eligible intervals», en *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, 2016, págs. 1-8.

[3] A. C. C. News, *ADAS Advanced Driver Assistance Systems – Definition AUTO Connected Car*. dirección: `https://www.alsa-project.org/wiki/Main_Pagehttps://www.autoconnectedcar.com/adas-advanced-driver-assistance-sytems-definition-auto-connected-car/`.

[4] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi y L. Kilmartin, «Intra-vehicle networks: A review», *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, n.º 2, págs. 534-545, 2015.

[5] A. Ali, «BroadR-Reach Technology: Enabling one pair Ethernet», Broadcom, inf. téc., 2012.

[6] T.-S. N. T. Group, *Time-Sensitive Networking Task Group*. dirección: `http://www.ieee802.org/1/pages/tsn.html`.

[7] S. International, *Time-Triggered Ethernet*, 2016. dirección: `https://doi.org/10.4271/AS6802`.

[8] W. Zeng, M. A. S. Khalid y S. Chowdhury, «In-Vehicle Networks Outlook: Achievements and Challenges», *IEEE Communications Surveys Tutorials*, vol. 18, n.º 3, págs. 1552-1571, 2016.

[9] A. Alliance, *Avnu*. dirección: `https://avnu.org/`.

[10] T.-S. N. T. Group, *Time-Sensitive Networking Task Group*, Time-Sensitive Networking Task Group, 2019. dirección: `http://www.ieee802.org/1/pages/tsn.html`.

[11] D. R. O. D. E. Corporation, *OSI IS-IS Intra-domain Routing Protocol*, 1990.

[12] «IEEE Std. 802.1 AS-2011: IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks», 2011.

[13] «IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks», *IEEE Std. 802.1Q-2014, Clause 10.9*, pág. 195, 2014.

[14] «IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks», *IEEE Std. 802.1Q-2014, Clause 11*, págs. 206-213, 2014.

[15] «IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks», *IEEE Std. 802.1Q-2014, Clause 35.1*, págs. 1399-1402, 2014.

[16] «IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks», *IEEE Std. 802.1Q-2014, Clause 35*, págs. 1398-1425, 2014.

[17] «IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks», *IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011)*, págs. 1-1832, 2014.

[18] «IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams», *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, págs. C1-72, 2010.

[19] J. Cao, P. Cuijpers, R. Bril y J. Lukkien, «Independent WCRT analysis for individual priority classes in Ethernet AVB», *Real-Time Systems*, oct. de 2018.

[20] C. de estándares de IEEE 802, *IEEE 802.1: 802.1Qat - Stream Reservation Protocol*. dirección: `http://www.ieee802.org/1/pages/802.1at.html`.

[21] O. Ltd, *What is OMNeT++?* Dirección: `https://omnetpp.org/intro/`.

[22] L. Galaxy Technologies, *What is GNS3?* Dirección: `https://gns3.com/software`.

[23] M. Team, *Mininet - An Instant Virtual Network on your Laptop (or other PC)*. dirección: `http://mininet.org/`.

[24] O. Ltd, *OpenSim Ltd.* dirección: `https://github.com/omnetpp`.

[25] ——, *OMNeT++ Discrete Event Simulator*. dirección: `https://github.com/omnetpp/omnetpp`.

[26] I. Eclipse Foundation. dirección: `https://www.eclipse.org/ide/`.

[27] «Internet Protocol», inf. téc. 791, sep. de 1981, 51 págs. dirección: `https://rfc-editor.org/rfc/rfc791.txt`.

[28] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach y T. Berners-Lee, *RFC2616: Hypertext Transfer Protocol – HTTP/1.1*, USA, 1999.

[29] N. language overview. dirección: `http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm`.

[30] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer y K. Rothermel, «NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++», mar. de 2019.

[31] C. C. over Realtime Ethernet) research group, *CoRE4INET*. dirección: `https://github.com/CoRE-RG/CoRE4INET`.

[32] TTTech, «Time-Triggered Ethernet - A Powerful Network Solution for Multiple Purpose», TTTech, inf. téc. dirección: `https://doi.org/10.4271/AS6802`.

[33] «IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks», *IEEE Std. 802.1Q-2014, Clause 34*, págs. 1391-1398, 2014.

[34] «IEEE Standard for Local and metropolitan area networks–Audio Video Bridging (AVB) Systems», *802.1BA-2011*, págs. 1-31, 2011.

[35] D. de OpenAvnu, *Raspberrypi with AVB*. dirección: `https://github.com/AVnu/OpenAvnu/issues/656`.

[36] M. T. B. Lindvall, *Apu2C4 Assembly*. dirección: `https://thnee.se/articles/pcengines-apu2c4-assembly/` (visitado 01-03-2020).

[37] P. E. GmbH., *PC Engines*. dirección: `https://www.pcengines.ch/`.

[38] Yeppon, *Netgear 16Port Gigabit Smart Switch Gs716tv3*. dirección: https://www.yeppon.it/p-netgear-16port-gigabit-smart-141735/.

[39] O. Alliance, *OpenAvnu*. dirección: https://github.com/AVnu/OpenAvnu.

[40] L. Foundation, *What is Automotive Grade Linux?* Dirección: https://www.automotivelinux.org/.

[41] ——, *The Linux Foundation*. dirección: https://www.linuxfoundation.org/.

[42] ——, *Members*. dirección: https://www.automotivelinux.org/about/members.

[43] A. Inc., *About AudioScience*. dirección: http://www.audioscience.com/internet/company/about.htm.

[44] E. Mann, L. Pearson, A. Elder, C. Hall, C. Gunther, J. Koftinoff, A. Butterworth, D. Underwood, G. Venkatesan y B. Thomsen, «AVB Software Interfaces», inf. téc., 2015.

[45] «IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks», *IEEE Std. 802.1Q-2014, Clause 10*, págs. 163-205, 2014.

[46] T. L. P. Interface, *Ethtool Manual*. dirección: http://man7.org/linux/man-pages/man8/ethtool.8.html.

[47] L. M. Garcia, *Tcpdump*. dirección: http://www.tcpdump.org/.

[48] Wireshark, *Wireshark - About*. dirección: https://www.wireshark.org/.

[49] P. S. Foundation, *About Python*. dirección: https://www.python.org/about/.

[50] ——, *socket — Low-level networking interface*. dirección: https://docs.python.org/3/library/socket.html.

[51] P. community, *pcapy - Pypi*. dirección: https://pypi.org/project/pcapy/.

[52] E. F. M. D. John Hunter Darren Dale y the Matplotlib development team, *matplotlib.pyplot*. dirección: https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.html.

[53] J. M. Waskom, *seaborn: statistical data visualization*. dirección: https://seaborn.pydata.org/.

[54] NumFOCUS, *pandas*. dirección: https://pandas.pydata.org/.

[55] K. Ichikawa, «Precision Time Protocol on Linux», Fujitsu, inf. téc., 2014.

[56] «1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems», págs. 83-107, 2008.

[57] GStreamer, *What is gstreamer?* Dirección: https://gstreamer.freedesktop.org/documentation/application-development/introduction/gstreamer.html.

[58] «1722-2016 - IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks», *IEEE Std. 1722-2016*, 2016.

[59] XMOS, «AVB Endpoint Design Guide», pág. 8, 2014.

[60] «IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks», *IEEE Std. 802.1Q-2014, Annex L*, págs. 1732-1748, 2014.