UNIVERSIDAD DE MURCIA

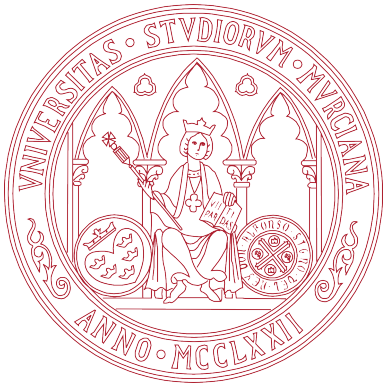
ESCUELA INTERNACIONAL DE DOCTORADO

Evaluation of Video Transmission Systems over Information Centric Content Delivery Networks: towards the ICNaaS

Evaluación de Sistemas de Transmisión de Vídeo sobre Redes de Distribución de Contenidos Centradas en Información: hacia un ICNaaS

D. Jordi Ortiz Murillo

2018



Universidad de Murcia

Facultad de Informática

Evaluation of video transmission systems over information centric content delivery networks: towards the ICNaaS

Evaluación de sistemas de transmisión de vídeo sobre redes de distribución de contenidos centradas en información: hacia un ICNaaS

Tesis Doctoral

Presentada por:
Jordi Ortiz Murillo

Supervisada por:
Prof. Antonio F. Skarmeta Gómez

Murcia, Junio de 2018

Resumen

Las computadoras toman su nombre de su capacidad de administrar, almacenar y operar con datos para resolver problemas de distinta índole. Inicialmente dichos datos eran expuestos mediante paneles de control, posteriormente impresoras y finalmente con pantallas o monitores. Con la aparición de las pantallas y los monitores no se tardó en introducir mecanismos gráficos para expresar datos que a la postre darían paso a la representación de imágenes. Estas imágenes en muchos casos tenían que ser persistentes y almacenadas en soporte no volátil. Debido a los limitados recursos de las computadoras de la época, tanto en capacidad de cómputo como de almacenamiento, optimizaciones en cuanto a la codificación y almacenamiento de imágenes fueron investigadas dando lugar a los codecs (acrónimo de la unión codificador-decodificador) y formatos de imagen.

De forma similar a cómo una secuencia de fotos capturadas a gran velocidad constituyó la aparición del vídeo en el mundo analógico, la aparición de imágenes en la era digital acarreó consigo la aparición del vídeo digital. Análogamente a cómo el tratamiento y almacenamiento de imágenes digitales ha ido evolucionando, el tratamiento y almacenamiento de vídeo digital dio paso a los codecs y formatos de vídeo, los cuales han evolucionado fuertemente ligados a la potencia de los ordenadores así como a las redes de computadoras, pues uno de los principales acometidos de almacenar un vídeo digital es su diseminación y por tanto su transmisión.

Las redes de ordenadores y su mayor exponente, Internet, han evolucionado desde líneas de pocos hasta miles de millones Bits por Segundo (bps), de sistemas punto a punto a medios compartidos y de sistemas terrestres a medios aéreos. Los codecs de vídeo han aumentado sus capacidades de compresión, en algunos casos estableciendo como objetivos mínimos aquellos estándares que definían las conexiones de red. De manera paralela, diversos protocolos han emergido para facilitar la tarea de transmitir vídeo y sobretodo para permitir la reproducción de flujos o transmisión por secuencias (más conocido por su término anglosajón *streaming*) de vídeo.

La red ha visto emerger multitud de protocolos, destacando el artífice de su universalidad y convertido en el estándar de facto, el protocolo de Internet (IP). El

protocolo IP define el nivel de red, siendo en el nivel de transporte donde comienza la heterogeneidad de la red con protocolos como Transmission Control Protocol (TCP), User Datagram Protocol (UDP) o el menos conocido y moderno Stream Control Transmission Protocol (SCTP) entre los estándares y viéndose ampliado en el nivel de aplicación (aglutinando sesión, presentación y aplicación del modelo de capas Open Systems Interconnection (OSI)) con protocolos más específicos como Real-time Transport Protocol (RTP), Session Description Protocol (SDP) o Real-time Streaming Protocol (RTSP), por nombrar algunos de los más influyentes históricamente en streaming de vídeo.

El empleo de IP para todo tipo de escenarios y sus deficiencias, muchas de ellas debido a su diseño inicial el cual por otro lado ha sobrepasado toda expectativa, ha llevado a limitaciones que poco a poco se van tornando insalvables o simplemente demasiado complejas debido a la retro-compatibilidad. Este estancamiento en la evolución de la red u osificación de la misma es combatido por lo que se ha venido denominando el Future Internet (FI) o Internet del Futuro. Dentro de FI nos encontramos propuestas a medio/largo plazo para solventar desde la conectividad en general hasta problemas muy específicos, pasando por nuevos paradigmas que cambian completamente la visión de la red en sí como podría ser el Internet of Things (IoT) (Internet de la cosas). Entre ellas una tecnología que ha atraído la atención de la industria por sus capacidades de facilitar la incorporación de otras propuestas a entornos de producción es Software Defined Networking (SDN) (o redes definidas por software).

Esta Tesis se centrará en el análisis, aplicación y definición de nuevos sistemas de transmisión de vídeo con el objetivo de mejorar el uso efectivo de la red, involucrando a la misma como parte del proceso. Las propuestas que formarán parte de esta tesis encaran la optimización de la transmisión de vídeo sobre la red desde tres metodologías distintas, una continuista, otra transgresora y finalmente una conciliadora con respecto a su posible impacto en su adopción sobre la actual red. La metodología continuista está basada en un protocolo ya estandarizado por el Internet Engineering Task Force (IETF) como es SCTP. Por su parte, la apuesta transgresora pasa por analizar cómo afecta la transmisión de vídeo en redes de las denominadas FI y cómo nuevos entornos más restrictivos como son las redes IoT pueden afectar al transporte de vídeo sobre las mismas. Finalmente, la apuesta conciliadora pasa por aprovechar las posibilidades que ofrece SDN para ofrecer servicios de transmisión de vídeo avanzados, mejorando las alternativas de forma transparente y no rupturista.

La propuesta de empleo de SCTP es específica para la transmisión de vídeo escalable dadas las propiedades inherentes del mismo. La idea consiste en aprovechar el concepto de flujos de una asociación SCTP junto con la definición de capas y sus dependencias del

vídeo escalable para permitir distintos tipos de protección para las mismas, dependiendo de la relevancia de las capas para la decodificación del vídeo. En particular se propone el uso de Scalable Video Coding (H.264/SVC) en el que su capa base es completamente retro-compatible con uno de los codecs de vídeo más extendidos Advanced Video Coding (H.264/AVC). La capa base de H.264/SVC es independiente de cualquier otra capa para su decodificación y las capas de mejora suponen como su nombre indica, mejoras en la calidad percibida en tres sentidos, tamaño, tasa de refresco de imágenes (Frames per second (FPS)) o definición. Por lo tanto, es lógico ofrecer mayor nivel de protección a las capas de las que otras capas dependen, siendo el mayor exponente la capa base. Algunos de los mecanismos evaluados en la tesis no han sido estandarizados todavía, pero ofrecen características muy interesantes en general para la transmisión de contenido multimedia. SCTP posee características muy deseables como son el multi-homing o su capacidad de emplear múltiples direcciones de red para evitar pérdidas de conexión aprovechando las capacidades multi-interfaz que muchos dispositivos poseen. Además algunas de las propuestas existentes pretenden emplear esta capacidad para conseguir la emisión/recepción paralela por distintas interfaces consiguiendo así un aumento en el ancho de banda disponible. La evaluación del sistema es realizada mediante simulaciones.

A continuación se resaltan algunas de las aportaciones en el ámbito de la transmisión de vídeo escalable con SCTP realizadas:

- Posiblemente fuese la primera asociación de transporte de vídeo con SCTP y más en particular, el mapeo de las capas H.264/SVC con streams de SCTP.

- Combinación de distintos niveles de protección en la capa de transporte dependiendo del contenido del nivel de aplicación. En particular, protegiendo la capa base de H.264/SVC permitiendo pérdidas en capas de mejora que no son imprescindibles para la obtención de un servicio mínimo.

- Posiblemente esta haya sido una de las primeras aplicaciones prácticas para la propuesta Concurrent MultiPath Transfer for Stream Control Transmission Protocol (CMT-SCTP), frente a la típica evaluación con tasa de transferencia constante.

- Implementación de un alimentador de paquetes RTP de tamaño variable con marca de tiempo para el simulador NS-2 con el fin de obtener resultados más representativos en el ámbito de la transmisión de vídeo, además de un sistema multi-traza para poder modelar flujos multi-capa como H.264/SVC. Una traza hace referencia a un fichero con información relativa a cada uno de los paquetes de vídeo que serían transportados

en un entorno real con datos como el tamaño o el instante en que debe ser enviado durante la reproducción.

- Extracción de pistas de hint de ficheros mp4 a trazas, dónde cada capa de H.264/SVC es exportada a una traza distinta simplificando el posterior proceso de simulación.

- Resultados expuestos a la comunidad científica en los artículos 'SCTP as scalable video coding transport' [1] y 'Video Adaptation based on SVC File Format [2]' entre otros.

La opción rupturista viene de la mano de Heterogeneity Inclusion and Mobility Adaptation through Locator ID Separation (HIMALIS) y Content-Centric Networking (CCN), dos arquitecturas de FI que rompen con la asunción de que una dirección de red debe estar asociada con una localización, tal y como asumen las redes basadas en Internet Protocol (IP). Mientras que HIMALIS ofrece una infraestructura de red completa con un sistema de rutas basado en direcciones de red, CCN aboga por la metodología Information Centric Networking (ICN) o redes centradas en información con un sistema de rutas basado en el empleo de Uniform Resource Locators (URLs) en la que es el contenido en sí el objetivo del sistema de direcciones y no su localización. Evaluamos por tanto el uso de estas dos arquitecturas para la transmisión de vídeo sobre HyperText Transfer Protocol (HTTP) adoptando por tanto una de las últimas tendencias más extendidas en cuanto a la transmisión de vídeo. Adicionalmente la tesis evalua las posibilidades de dichos sistemas para la transmisión de vídeo producido o consumido por 'cosas' en el ámbito de IoT. En este caso la metodología empleada es la del despliegue de las arquitecturas sobre laboratorios globalmente distribuidos, aportando por tanto la credibilidad de los resultados al ser evaluados directamente sobre Internet y obteniendo repetibilidad mediante la adopción de sistemas de gestión de experimentación.

Acto seguido se enumeran las más relevantes aportaciones e hitos de esta tesis en su propuesta rupturista de transmisión de vídeo en FI.

- Evaluación de sistemas de transmisión de vídeo sobre HTTP en redes del futuro.

- Comparación de dos sistemas contrapuestos como son HIMALIS y CCN.

- Aproximación al efecto de redes IoT en dichos sistemas.

- Despliegue y evaluación en entorno altamente distribuido y públicos.

- Experimentación instrumentalizada con el foco puesto en la repetibilidad.

- Resultados expuestos en capítulo de libro '6. Information-Centric Network for Future Internet Video Delivery.' del libro 'User-centric and Information-centric Networking and Services: Access Networks and Emerging Trends.' [3].

Finalmente, esta tesis propone un sistema capaz de sustituir los actuales sistemas de distribución de contenido Content Delivery Network (CDN) de forma transparente, integrando algunos de sus elementos actuales y aportando características deseables a dichos sistemas como son la segmentación por proveedor o CDN como servicio (por sus siglas en inglés *aaS* - as a Service) de contenidos mediante el uso de SDN. Además una aproximación de tipo ICN es adoptada, de ahí que nos refiramos en la tesis a Information Centric Network as a Service (ICNaaS), ofreciendo optimizaciones específicas en la red en cuanto a la localización del contenido basada en la meta-información del mismo, en particular mostrando el caso de uso de video escalable sobre HTTP. Para ello se define una arquitectura de capas sobre el controlador SDN y se definen las interfaces a emplear por los distintas entidades involucradas en la provisión del servicio. Además se diferencian dos redes de control, a la ya existente red de control SDN, se añade una red de control ICN específica para los elementos de red implicados en la provisión del servicio, y dos redes de administración, a la ya existente red de administración para la SDN se añade la red de administración ICN encargada de la gestión de las instancias ICN.

La propuesta es evaluada en dos despliegues reales sobre la red de la Universidad de Murcia, inicialmente con instancias virtuales de los elementos de red y finalmente con equipamiento compatible. A su vez, la evaluación del sistema emplea un sistema de gestión de experimentación fruto de la experiencia obtenida durante la evaluación de FI para garantizar la repetibilidad.

Las aportaciones realizadas como parte de la propuesta conciliadora basada en SDN en contrapartida a la propuesta transgresora basada en HIMALIS y CCN se enumeran a continuación:

- Propuesta de un sistema de distribución de contenidos y caches como servicio, ICNaaS.

- Propuesta de un sistema de transporte de HTTP basado en el uso de SDN y transparente a los participantes (end-points).

- Guiado del tráfico en base a la URL y a la instancia ICN a la que pertenece. Lo que implica la inspección de datos de nivel de aplicación fuera de las capacidades nativas ofrecidas por los elementos de red que conforman una SDN basada en OpenFlow Protocol (OpenFlow).

- Propuesta de un sistema de capas para aplicaciones SDN aplicado a la propuesta de ICNaaS para transmisión de vídeo escalable H.264/SVC sobre HTTP.
- Propuesta de un sistema de precarga adelantada o prefetching transparente para HTTP. Aplicación del sistema al caso de video H.264/SVC transportado con Dynamic Adaptive Streaming over HTTP (DASH)
- Implementación y evaluación de la propuesta en laboratorio, siendo además los dos primeros despliegues SDN de la Universidad de Murcia.

Para cada alternativa se presentan conclusiones y vías futuras. De entre ellas la más interesante a corto/medio plazo teniendo en cuenta los últimos acontecimientos en la comunidad investigadora sería la inclusión de la propuesta de ICN como servicio en entornos de virtualización de funciones de red o Network Function Virtualisation (NFV). Además, aunque el caso de uso para el cual ICNaaS fue diseñado está orientado a la transmisión de vídeo, el sistema es lo bastante genérico para ser empleado en otros entornos que emplean HTTP como medio de transporte.

Agradecimientos

Esta página espero que sirva de agradecimiento a todos los que de una forma u otra habéis hecho posible esta tesis.

Como no podía ser de otra manera quiero empezar agradeciendo a mi mujer, Mari Ángeles, por su apoyo, amor y sobretodo paciencia estos interminables años, por tus ¡Aprovecha!, porque lo que tenía que ser una carrera de obstáculos he conseguido que se convirtiese en un maratón. A mis tres niños, Rodrigo, Ángeles y Neus por darle sentido a todo y en particular a este libro, os debo muchas horas de juego en algunos de los años más importantes de vuestras vidas.

A mis padres, que siempre habéis visto en mi las partes buenas y se os habéis ocupado de enseñarme desde niño que a ningún sitio se llega sin esfuerzo. Por obligarme cuando no tenía ganas y por apoyarme cuando no tenía fuerzas. A mi hermanita, porque con un buen ejemplo siempre es más fácil llegar lejos.

A Antonio, director de esta Tesis, por abrirme las puertas del trabajo de mi vida y haberme hecho mejorar como investigador y como persona, a veces con la mano izquierda y a veces con la derecha que suele ser la que más necesito. Gracias por los 'Jordi, céntrate!' que tan bien me vienen y que espero seguir recibiendo durante muchos años. Gracias por hacer de nuestro hormiguero una pequeña familia.

A todos los profesores de la Facultad de Informática que habéis forjado y seguís forjando la materia prima que sirvió para llegar hasta aquí, en especial a Eduardo por tus buenos consejos. A todos mis compañeros de departamento a los que muchos considero mis amigos, gracias por las palabras de ánimo y por meteros conmigo cuando tocaba, habéis sido parte importante de mi motivación. Por supuesto, no puedo olvidar a mis muchos compañeros de estudios y de trabajo. A Alejandro Rosúa fue un placer tenerte como compañero y es un privilegio tenerte como amigo. A Elena y los Alejandros, por las visitas, por las comidas, por las risas. A Pedro J y nuestros viajes a Aveiro. A Pedro Martínez por dejarme trastear en Gaia y por los buenos ratos que en ella hemos pasado. A todos los que desde DibuLibu y sus alrededores habéis tenido tiempo para ayudarme y aconsejarme. A mis amigos en ATICA, por vuestra disponibilidad y porque siempre hubiese sido más fácil decir que no. A José Luis Álcoba, gracias por haberme enseñado que un ordenador no sólo sirve para jugar y al resto de mis compañeros de la cadena Fiesta.

Finalmente, a la Universität Klagenfurt, a Hermann Hellwagnner y a Michael Ransburg por haberme acogido y haber abierto mi mente a otras formas de pensar. A Hilda por haberme acogido en su casa permitiéndome experimentar la verdadera vida Austriaca.

University of Murcia

Faculty of Computer Science

Evaluation of video transmission systems over information centric content delivery networks: towards the ICNaaS

PhD Thesis

Author:
Jordi Ortiz Murillo

Thesis Advisors:
Prof. Antonio F. Skarmeta Gómez

Murcia, September 2018

Abstract

The term computer, as defined by the Oxford dictionary, referenced initially to a 'person' who carried out calculations or computations. Later, the word acquired the meaning by which we all understand it now as a 'machine' that carries out calculations.

The data fed to a computer as well as the outputs produced by the calculations were initially represented by means of patch panels, followed by printers and finally with screens. The screen adoption as the mechanism to communicate with humans rapidly lead to the usage of graphic charts to represent data which in turn resulted in the representation of picture. Those pictures were usually stored in a persistent storage media. Due to the scarce computer resources in that time period, in terms of computing power and storage, optimizations to represent graphics and in particular pictures were researched. As a consequence what we all know as picture codecs (from coder-decoder) and formats.

Similarly to how a high speed captured sequence of photos fostered the appearance of analog video, the advances in picture processing in computers forged the digital video era. In parallel with picture codec and format evolution, video codecs and formats have been emerging and evolving. The video coding evolution has been heavily linked with computer network as well as computing power capabilities. It has to be taken into account that one of the primary objectives of storing and processing digital video is its transmission to another peer.

Computer networks and its main representative, the Internet, have grown from single point few Bits per Second (bps) wide land lines to multi-point millions bps wireless connections. Video codecs have been increasing their compression capabilities, sometimes taking as reference objectives the minimum rate defined by network standards. At the same time, several protocols intending to facilitate video transmission have appeared also focusing on allowing the consumption of video streams, therefore the appearance of the term '*video streaming*'.

The IP stands out among all of the protocols that have been emerging as the responsible for the Internet itself as we all know it, becoming the 'de facto' standard for the network layer. The transport layer is where Internet's heterogeneity starts with TCP, UDP or the

less popular SCTP among the standardized options. The heterogeneity is even wider in the application layer (which includes session, presentation and application layers from the OSI heap) with more specific and application oriented protocols like RTP, SDP or RTSP, just enumerating some of the historically relevant protocols related to video streaming.

The usage of IP for any scenario and its drawbacks, highly tied to its initial design which has in the end exceeded any expectations, has imposed limitations which have turned unavoidable or simply too complex due to the need of retro-compatibility. The halt in network evolution or ossification is faced by what has been named as the FI. What are known as FI mid and long term proposals to solve from generic connectivity problems up to very narrow and specific problems also facing new paradigms like the IoT where the usage of the network is transformed as well as the actors involved in the communication process. Among the FI proposals, the SDN has attracted industry's attention due to its capabilities to facilitate other proposals deployment into production.

This Thesis focuses on the analysis, applicability and definition of new video transmission systems intended to enhance the effective network usage, involving the network itself as part of the solution. The proposals included in this work approach network optimization for video transmission from three different strategies, conservative, clean-slate and evolutionary in terms of its possible impact in case of its adoption taking into account the actual network architecture. The conservative approach is based on an already IETF standardized protocol, SCTP. The clean-slate approach focuses on analyzing how video transmission would affect the new FI networks and how new and more restrictive environment, such as IoT, could affect video transmission on top of them. Finally the evolutionary approach takes advantage of SDN characteristics to offer advanced video streaming services, enhancing actual alternatives transparently and without breaking with actual networking concepts.

The proposal based on SCTP is scalable video coding specific due to its design characteristics. The main idea is to take profit of the stream definition within an SCTP association in coordination with the layer definition and its dependencies from scalable video codecs to provide with different protection levels depending on the layer relevance for video decoding process. The usage of H.264/SVC is proposed since its base layer is completely backward-compatible with one of the more extended video codecs, H.264/AVC. H.264/SVC's base layer is independently decodable from any other layer and the enhancement layers serve as enhancements in three different dimensions, size, frame rate or definition. Hence, it is logical to offer higher protection to those layers of which other layers depend, having as the main representative the base layer. Some mechanisms evaluated in this Thesis haven't still been yet standardized, but they offer very interesting

capabilities for the transmission of multimedia content. SCTP has already interesting characteristics such as multi-homing or its capability to employ multiple network addresses to avoid connectivity loss by employing multi-interface characteristics that nowadays most devices have. In addition some of the proposals employ this capability to achieve parallel data transmission over multiple interfaces thus achieving a bandwidth enhancement. The evaluation is performed through simulations.

A list of contributions in scalable video transmission with SCTP are listed below:

- This was probably the first approach that associated video transport with SCTP and in particular the mapping of H.264/SVC layers with SCTP streams.
- Merging the different protection levels offered at the transport layer depending on the content at the application level. In particular, protecting the H.264/SVC base layer allowing losses in enhancement layer which are not indispensable for achieving a minimum service.
- Definitely this was one of the first proposals to employ CMT-SCTP in a practical application instead of the typical bulk rate simulation.
- Development of a RTP packet feeder with variable packet size with timestamping for the NS-2 simulator to obtain meaningful results in field of video transmission, in addition a multi-trace system was also implemented to model multi-layer flows such as H.264/SVC. A trace makes reference to a file with information relative to each of the video packet as it would be transported in a real scenario with related data such as size or timestamp in which it should be sent over the network (usually the decoding order and not the presentation order).
- Hint track extraction from mp4 files to traces, where each H.264/SVC layer is exported to one trace file therefore easing the later simulation process.
- Results delivered to the research community in the papers 'SCTP as scalable video coding transport' [1]' and 'Video Adaptation based on SVC File Format [2]' among others.

The clean-slate approach is supported by HIMALIS and CCN, two FI architectures that break with the premise that each network address must be associated with a location, like in IP networks. While the HIMALIS architecture offers a complete network infrastructure with a routing system based on network addresses, CCN follows an ICN methodology with a routing system leveraging on URL being the content the objective and not its location.

Therefore this work evaluates the usage of these two architectures for video transmission on top of HTTP thus following the last tendencies in this field. Additionally this Thesis evaluates the possibilities offered by such systems in terms of transmitting video produced or consumed by 'things' in the IoT context. In this case the methodology followed is the deployment of the architectures over globally distributed laboratories, therefore adding credibility to the obtained results since the transmission is done over the real Internet while achieving the desired repeatability by means of adopting experimentation management systems.

An enumeration of the more relevant contributions on the clean-slate approach of this work follows:

- Evaluation of HTTP based video streaming systems in FI networks.
- Comparison of two differentiated FI systems like HIMALIS and CCN.
- IoT effect approach on such systems.
- Deployment and evaluation in a highly distributed and public environment.
- Automated experimentation focused on achieving repeatability.
- Results disseminated in book chapter '6. Information-Centric Network for Future Internet Video Delivery.' del libro 'User-centric and Information-centric Networking and Services: Access Networks and Emerging Trends.' [3].

Finally, a system envisioned to transparently replace current CDNs services is proposed, integrating some of their current elements and adding desirable characteristics to such systems like provider based segmentation or CDN as a Service leveraging on SDN. Furthermore, ICN philosophy is adopted, therefore the term ICNaaS is used, offering specific in-network optimizations in terms of content location based on its own meta-information, in particular showing the scalable video streaming over HTTP use case. To that end, a layered architecture sitting on top of the SDN controller is defined in addition to the interfaces to be used by the variety of entities involved in the service provision. Two control networks are identified, to the already existing SDN control plane, a specific ICN control plane is specified, as well as two administration networks, to the already existing SDN management network, an specific ICN management networks is added.

The proposal is evaluated in two different real deployments on top of University of Murcia network, initially with virtual network elements and finally with OpenFlow capable

hardware. In addition, the system evaluation leverages on an experimentation management system result of the experience gained during the FI evaluation to obtain repeatability.

Outcomes from the evolutionary approach based on SDN are listed below:

- Proposal of a cache and content distribution service as a service, ICNaaS.
- Proposal of a transport system for HTTP based in SDN and transparent to the end-points.
- Traffic steering based in URL and the ICN instance to which it belongs. Which on the other hand implies the analysis of application layer data in the network elements, exceeding the capabilities of the network elements belonging to the SDN based in OpenFlow.
- Proposal of a layering system for SDN applications applied to the ICNaaS proposal for scalable video streaming (H.264/SVC) on top of HTTP.
- Proposal of a transparent HTTP prefetching system. Employ the system to the H.264/SVC streaming over DASH case.
- Development and evaluation of the proposal in the laboratory, which in turn have been the two first SDN deployments in the University of Murcia.

For each alternative conclusions and future work is presented. Among them, the more interesting in the short-term taking into account the current events in the research community is the inclusion of ICNaaS in NFV environments. Although the ICNaaS has been designed having in mind video streaming, the system is generic enough to be applied to other environments in which are transported on top of HTTP.

Acknowledgements

I hope this page serves to show my gratefulness to those that one way or another had made possible this thesis.

There is no other way to start but by thanking my wife, Mari Ángeles, for her support, love and above all her patience all these unending years, for those 'Take your chance!', because what should have been an obstacle course has become a full marathon. To my three little kids, Rodrigo, Angeles and Neus since you give meaning to everything and particularly to this book, I owe you too much play time in some of the more important years of your life.

To my parents, you have always seen the best parts in me and have taught me that everything comes at a cost. For forcing me when I was lazy and supporting me when I was too tired. To my sister, its always easier to get further if one has a good reference.

To Antonio, this thesis adviser, for letting me to get into the job of my life and making me advance as a researcher and as a human being, sometimes smoothly and sometimes sharply as I usually be in need of. Thank you for those 'Jordi, Focus!', that are so useful to me and that I hope I'll keep getting for some years. Thank you for making our ant's nest a small family.

To all the Faculty of Computer Science's teachers for forging the raw material that served to reach this point, specially to Eduardo for his good advice. To all my colleagues in the department most of whom I consider my friends, thank you for your support and for pushing me when needed, you have been present in my motivation. I can not forget my studies and work mates. To Alejandro Rosúa, it was my pleasure to have you as colleague and my privilege to have you as a friend. To Elena and the Alejandros, for visiting me in the lab, lunch time and the laughs. To Pedro J and our Aveiro trips. Pedro Martínez for letting me play in the Gaia playground and the good times we spent there together. To all of you in Dibulibu and around that have had the time to help and give advice. To my friends in ATICA, for your availability and because it would have been easier to say no. To José Luis Alcoba, thank you for teaching me that a computer is not only a play machine and to the rest of my colleagues in Fiesta.

Finally, to Klagenfurt Universität, to Herman Hellwagner and Michael Ransburg for letting me join them and having opened my mind to other ways of thinking. To Hilda for opening her home and letting me enjoy the real Austrian life.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

This chapter gives a brief introduction to the key concepts, technologies and their evolution that will contextualize the rest of this Thesis. A brief introduction to computer video coding evolution is followed by an introduction to video transmission on top of computer networks, also known as video streaming. The Future Internet keyword is also introduced pointing out some of the architectures and concepts that will be employed thorough the Thesis, followed by the notion of video streaming in Future Internet (FI). Finally, the objective, contributions and structure of the Thesis are introduced followed by a list of related publications.

1.1 Contextualization

Computers take their name of their ability to manage and apply operations over data to solve a wide variety of problems. The resulting computations were usually to be output and offered to the user. The data that was printed to paper or shown with lights on a control panel on the early days was conveniently sent to screens soon afterwards. The output to screen as a consequence of computer graphics research and the popularization of home computers immediately introduced the representation of data in still images such as data charts. The desire to store those still images in a durable device so that they could be recovered afterwards appeared naturally soon after. Computer resources were scarce and expensive therefore a way to save them was needed and that is how picture coding and image formats came into scene.

Similarly to how photos evolved to film as a succession of very fast captured moments, computer video appeared also as a sequence of still images. Initially taking advantage from the compression methods offered by their contemporary still image counterparts and

eventually taking profit of the similarities in the picture sequences to introduce specific video compression techniques, video codecs came into play.

The main and solely objective of computer networks was to transmit information. Text at a first step but soon after pictures and finally video as the final consequence. The transmission of pictures and video was to a high degree the precursor of picture and video coding techniques since although the resources in computers were scarce, resources in computer networks were even more limited.

Video transmission over networks, also known as video streaming, is one of the more resource consuming applications for computer networks. While picture files were, in general, transmitted as a whole, video can also be transmitted as a whole, nevertheless the term streaming refer to the real-time playback of video content reducing the time to wait for the video playing to the minimum possible.

The new challenge that offered video streaming, also audio streaming was a challenge but with a considerable lower amount of information per file, introduced new techniques oriented to enhancing user experience while reducing network utilization. There has been an evolution in transmission techniques triggered by network evolution from datagram based streaming on the early Integrated Services Digital Network (ISDN) days to the HyperText Transfer Protocol (HTTP) based transmission of today's fiber environment.

Video codecs have been affected by these techniques also, usually in terms of the minimum bit-rate available to be used for data transmission (such as the conformance of MPEG-2 to sustain a bit-stream capable to be transported on top of ISDN) but also with new use cases such as in-network adaptation (like scalable video codecs that allow in-network bit-rate adaptation at a very low amount of processing power) and stereoscopic view and what is to come in the future.

Nowadays there is a new trend in computer networks that claim to break the 'ossification' of the Internet as we know it. The Future Internet (FI) is known as a general term for new architectures that will enable the Internet of the future. Some of these architectures are clean-slate, others are evolution from nowadays networks and others are considered as enablers for new functionalities that haven't been envisioned and are yet to come.

Among the architectures presented as part of the FI, three are key concepts in the successive parts of this Thesis. As an enhancement of the content distribution systems actually deployed such as the Content Delivery Network (CDN), the Information Centric Networking (ICN) philosophy (mainly represented by the Content-Centric Networking (CCN) architecture) focus on the content being transmitted and not on the networks edges of the communication any more, that philosophy implies the separation from identifier and

locator that subsequent network architectures propose, such as Heterogeneity Inclusion and Mobility Adaptation through Locator ID Separation (HIMALIS) or CCN. Finally, the inclusion of a low level network architecture that enables new protocols and clean-slate approaches to be deployed with hardware support and the centralization of network control decisions, represented by the Software Defined Networking (SDN) approach, that is envisioned as necessary to represent viable solutions that could reach in a short period the production level and therefore be deployed in real environments. Another key pattern already spreading and foreseeable as one of the more influencing characteristic for the near future is Internet of Things (IoT).

The appearance and possible adoption of the FI does not exclude video as the key and more resource consuming content to be transmitted in the near future, at least until our brains can be fed another way which is yet to come. In that sense applying and evaluating video transmission techniques to FI is a key research as well as proposing new mechanisms to save network resources is the goal to be achieved in the short term.

1.2 Objectives of the Thesis

This Thesis has the main objective of evolving the network layer by keeping in mind the application layer and in particular the transmission of video as its key contribution.

To that end, video streaming has been faced employing three very different approaches from the network point of view. Stream Control Transmission Protocol (SCTP) represents a conservative or legacy approach in which the Internet Protocol (IP) ossification is maintained and enhancements specific to the application level are addressed by adopting a 'nouveau' but already standardized protocol like SCTP and the adoption of Scalable coding (Scalable Video Coding (H.264/SVC)), which in turn is also standardized. Next, a clean-slate approach has been evaluated, focusing in the FI and how different architectures deal with the current all-HTTP approach for video transmission. Finally, an evolutionary approach is designed and evaluated in which a technology enabler like SDN is employed to transparently add new features to the content (and in particular video) distribution but focusing on taking profit of the existing evolved content distribution infrastructure.

To that extend, the Thesis has explored transmission mechanisms available but not yet employed for video streaming such as SCTP and its extensions. Has taken into account FI technologies, such as CCN and HIMALIS, and deployments, like IoT, for video transmission. And has proposed backwards compatible and nowadays deployable streaming techniques to evolve the actual content delivery paradigm leveraging on the

SDN architecture.

In addition to the typical simulation processes employed for evaluation of protocols, this Thesis has made intensive usage of real experimentation, leveraging on international testbeds as well as local infrastructure with the focus on validation and repeatability of the results obtained.

1.2.1 H.264/SVC on SCTP

The H.264/SVC offers some particularities that can be exploited to enhance end-to-end video delivery. The dependency between the coded video layers imply precedence in terms of importance to obtain the desired quality level also known as operation point. In that sense layers on which the operation point does not depend shouldn't be transmitted on the network but what is more important, those on which the operation point depends should be prioritized and/or protected during network transmission.

The SCTP protocol is general purpose although it was designed and standardized having telephony signaling in mind. Despite its origin, the protocol has some characteristics that make it really interesting for other uses and in particular for video delivery. Some amendments have been proposed to the original protocol, of which some have been already accepted such as the Potentially Failed SCTP or Stream Control Transmission Protocol Potentially Failed (SCTP-PF) [4] which allows loss recovery on demand at different levels or the Concurrent Multipath Transfer or Concurrent MultiPath Transfer for Stream Control Transmission Protocol (CMT-SCTP) [5] that tries to take advantage of the native multi-homing capabilities of the protocol to exceed the available bandwidth offered by just one interface overcoming the eternal ip binding problem. In addition, the multi-homing characteristic has probed already to be useful for mobility scenarios.

Applying SCTP and its extensions to H.264/SVC offers a full range of possibilities. First of all, each H.264/SVC layer can be transmitted on an independent SCTP stream which means that each of them has its own flow control. The SCTP-PR [6] (Partial Reliability extension) allows the loss of messages on demand which is an interesting feature for data that has inherently the expiration concept such as video. It is unnecessary to re-transmit a video chunk that should have been already played in the client. Finally, exploiting the multi-homing and multi-interface of nowadays computers (and why not, mobile phones) seems to be a good idea, therefore SCTP-CMT is employed to transmit over multiple network interfaces obtaining enhanced bandwidth which potentially allows using higher bit-rates. Recently approved extensions, such as SCTP-PF, allow the protection reduction for the enhancement layers ensuring that the base layer is always received, thus providing a

minimum quality service and no frame is completely loss, enhancement layers on the other hand arrive with a best effort approach.

The study compares the proposed solution with typical transmission methods used for video delivery which rely on Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

This study [1] was performed as a side work of the SCALNET project [7].

1.2.2 FI and video streaming for IoT

With the appearance of IoT and the FI and the generalized adoption of HTTP for video streaming, new horizons appeared in research. In terms of video in general, there is the question of what type of video streams could be considered if possible for the kind of devices and networks to be found in the IoT world. In terms of transmission, inspecting the impact of such an stream to be encapsulated into a HTTP compatible stream and in particular with the Dynamic Adaptive Streaming over HTTP (DASH) standard was a natural next-step. Finally, applying other FI approaches to the aforementioned video elements was an interesting movement.

Some IoT compatible video parameters are to be defined and justified taking into account the type of nodes and networks, so that the video could potentially be natively encoded, decoded and transmitted. How that would be done is out of the scope of this Thesis and the expertise of the writer but in any case, that stream will arrive to a border where full featured networks are to be used. At that point, the use of DASH is envisioned.

At the same time, it is clear that IoT deployment numbers will obsolete the IP world certifying its ossification and the need to break with it to some extent. The ICN paradigm solves the problem of content distribution and this Thesis evaluates a geographically widespread deployment for transmitting the video streams.

Other FI architectures such as the HIMALIS architecture propose a complete architecture offering a clean-slate proposal in terms of identification scheme breaking with IP and focusing on heterogeneity and mobility. The former characteristic is the one this Thesis will take profit of to evaluate not only IoT video transmission but also to provide with the hints of what IoT over CCN over HIMALIS would look like.

The evaluation of these architectures was carried out on top of the PlanetLab Europe (PLE) and employing the Network Experimentation Programming Interface (NEPI) software for defining the experiments and retrieving the results. Therefore, the repeatability and validity of the results is entrusted to the experimentation management software and to the testbed itself respectively.

The results of this study [8] were partly the outcome of the OpenLab [9] project.

1.2.3 H.264/SVC video delivery and Information Centric Network as a Service (ICNaaS) on SDN

From the previous work done, the adoption of DASH (a suboptimal solution in terms of bandwidth consumption) by the industry, the feeling that a clean-slate approach is not welcome and that the easiness of deployment is a requirement for any proposal, in addition to the appearance of SDN in the horizon provided with the spark to light the subsequent proposal of this Thesis.

Leveraging on SDN, the concept of ICN in which the edges of the communication depend on the content and not on the network identifiers can be carried out by installing the necessary flows in the network elements. Therefore, the centralized control plane from SDN simplifies the network based optimizations for higher layers such as the application level. Since DASH is the industry key technology for the video transmission, if no clean-slate approach is to be offered, DASH must be the key element of the proposed solution. As such, the first challenge is to steer HTTP traffic. The second challenge is to steer the traffic not only based on the protocol but also on the data being transported (the Uniform Resource Locator (URL)) and to maintain a register of where the content can be located. Finally, taking advantage of the gained knowledge about H.264/SVC and scalable codecs in general, specific optimizations are proposed for such a case.

Nonetheless, there is another key factor for ICN like deployments to be accepted by the industry. Providing with the easiness of operation for the users implicated in video content distribution, content provider, network operator and the client. In addition, the XaaS approach is also taken into account by offering an ICNaaS solution that will allow content providers to control the video delivery from the origin to the destination being able to remove or limit the need for third parties CDNs on demand. On the other hand, CDN operators are not rejected directly by the proposal by allowing them to remain as caching muscle for the content oriented architecture.

The work related to this study was initiated as part of the GN3plus [10] project.

1.2.4 Experimentation infrastructures

One of the objectives of this Thesis was to be able to experiment with different methods and compare them in terms of validity, complexity and capabilities for future research. Therefore, simulation software like NS-2 [11] was employed as the first approach. Next step

was to employ a public distributed testbed like PLE in which external influence is part of the results obtained and finally a exclusive testbed is employed such as Gaia Testbed [12] partly to avoid the external influences but mainly due to the use of specific hardware.

1.3 Contributions

As a result of the previously highlighted objectives and the haunt to face them, some contributions in form of software were made, some of those contributions didn't make it to this Thesis but made it to the open source community.

As part of the SCTP research, the ns-2 simulator had to be modified so that the Real-time Transport Protocol (RTP) implementation already present accepted variable length packets as payload. That modification allowed to simulate the video streaming over the rtp, in addition, synchronization was also introduced so that the packets being sent were in line with the video frame rate.

During the same period of time, several patch sets were merged into Open source audio and video processing tools (LibAV) [13] master branch therefore contributing to the networking and coding capabilities of the ffmpeg fork. An early SCTP implementation was updated and merged finally becoming available for anyone. In addition, Real-time Streaming Protocol (RTSP) and Real-time Messaging Protocol (RTMP) listen function so that server side communication with these protocols could be set was added. As a side study the wavelet based video coding was inspected as a parallel line for the scalable video alternative, therefore libschroedinger and dirac support was also updated for the library.

The amendments made publicly available can be found here: `https://patches.libav.org/project/libav-devel/list/?submitter=410&state=*&q=&archive=both&delegate=`

The involvement in SDN resulted finally in the collaboration within Open Network Operating System (ONOS) [14] community as part of the 'ONOS brigades'. The work was in this case directed to the capabilities of ONOS regarding meters, a key feature for Quality of Service (QoS) and bandwidth control for which this Thesis has some proposals.

The amendments are publicly available and can be found here: `https://gerrit.onosproject.org/#/q/owner:%22Jordi+Ortiz+%253Cjordi.ortiz.umu%2540gmail.com%253E%22`

Therefore, this Thesis have made contributions to the community that will hopefully help others businesses and research projects.

1.4 Thesis structure

The reminder of this Thesis is structured as follows:

Chapter 2 acts as an introduction and state of the art of the technologies employed thorough the Thesis and introduces the open gaps that the subsequent chapters will be facing.

Chapter 3 presents the work produced in relation to SCTP as video streaming protocol and presents the results.

Chapter 4 evaluates FI architectures for video streaming.

Chapter 5 introduces the ICNaaS architecture based on SDN.

Chapter 6 evaluates the architecture introduced in the previous chapter and provides with results.

Chapter 7 exposes the conclusions and future work.

Finally, the Glossary and Bibliography are presented.

1.5 Related publications

1.5.1 Indexed Journals

- *Integrating Adaptive Video Streaming Service with Multi-access Network Management* (Mobile Networks and Applications. The Journal of SPECIAL ISSUES on Mobility of Systems, Users, Data and Computing 2012 - Q2). Tiia Ojanperä, Markus Luoto, Jordi Ortiz & Mikko Myllyniemi

  `http://scimagojr.com/journalsearch.php?q=27306&tip=sid&clean=0`

- *SCTP as Scalable Video Coding transport* (Eurasip Journal on Signal Processing 2013 - Q3). Jordi Ortiz, Eduardo Martínez, Antonio Skarmeta

  `http://scimagojr.com/journalsearch.php?q=15300154801&tip=sid&clean=0`

- *Matching federation identities, the eduGAIN and STORK approach.* (Journal of Future Generation Computer Systems 2017 - Q1). Elena M. Torroglosa-Garcia, Jordi Ortiz-Murillo, Antonio F. Skarmeta-Gomez. `http://scimagojr.com/journalsearch.php?q=12264&tip=sid&clean=0`

1.5.2 Book Chapters

- *Information Centric Networking Future Internet Video Delivery* (In book, "User-Centric and Information-Centric Networking and Services Access Networks, Cloud and IoT Perspective", CRC Press, USA.). Jordi Ortiz, Pedro Martinez-Julia, Antonio Skarmeta [3]

  `https://www.bookdepository.com/User-Centric-Information-Centric-Netwo 9781138633322`

- *Federated Experimentation Infrastructure Interconnecting Sites from Both Europe and South Korea (SmartFIRE)* (In book, "Building the Future Internet through FIRE. 2016 FIRE Book: a Research and Experimentation based Approach"). Kostas Choumas, Thanasis Korakis, Jordi Ordiz, Antonio Skarmeta, Pedro Martinez-Julia, Taewan You, Heeyoung Jung, Hyunwoo Lee, Ted "Taekyoung" Kwon, Loic Baron, Serge Fdida, Woojin Seok, Minsun Lee, Jongwon Kim, Song Chong and Brecht Vermeulen. [15]

  Book: `http://www.riverpublishers.com/research_details.php?book_id=427`

  Chapter: `http://www.riverpublishers.com/pdf/ebook/chapter/RP_9788793519114C30.pdf`

1.5.3 Conferences

- *Towards User-driven Adaptation of H.264/SVC Streams.* (QoEMCS Junio 2010) Tampere. Finlandia. Jordi Ortiz Murillo, Michael Ransburg, Eduardo Martínez Graciá, Michael Sablatschan, Antonio F. Gómez Skarmeta, Hermann Hellwagner

- *Scalable Video Transmission in Home PLC networks.* (WMDCT Septiembre 2010) Valencia. España. Eduardo Martínez, Jordi Ortiz, Alejandro Rosúa, Antonio F. Skarmeta, Marcos Martínez.

- *Analysis and performance modeling of the packet-level loss process in wireless channels.*(MSWiM Octubre 2010) Bodrum. Turquía. Eduardo Martínez Graciá, Jordi Ortiz Murillo, Antonio F. Gómez Skarmeta.

- *Scalable Video Coding Impact on Networks.* (SVCVision Septiembre 2010). Lisboa. Portugal. Michael Ransburg (Klagenfurt University, Austria), Eduardo Martinez

(Univ. of Murcia, Spain), Tiia Sutinen (VTT Technical Research Centre of Finland), Jordi Ortíz (Univ. of Murcia, Spain), Michael Sablatschan, Hermann Hellwagner (Klagenfurt University, Austria)

- *Efficient SVC-to-AVC Conversion at a Media Aware Network Element* (Demo Paper - SVCVision Septiembre 2010). Lisboa. Portugal. Michael Sablatschan (Klagenfurt University, Austria), Jordi Ortíz (Univ. of Murcia, Spain), Michael Ransburg, Hermann Hellwagner (Klagenfurt University, Austria)

- *Video Adaptation based on SVC File Format* (IARIA CONTENT 2013). Eduardo Martínez, Jordi Ortiz, Rafael López, Antonio Skarmeta

- *Evaluating Video Streaming in Network Architectures for the Internet of Things* (IMIS/esIoT2013) Pedro Martínez Juliá, Elena Torroglosa García, Jordi Ortiz Murillo, and Antonio F. Skarmeta.

- *Scholar European Electronic Identity Federation* (TNC2015). Jordi Ortiz, Pedro Martinez-Julia, Christos Kanellopoulos, Antonio Skarmeta

- *SDN Integration and Management Solutions for Campus Network Enhanced Services* (ICIN2016) Jordi Ortiz, José I. Aznar, Alaitz Mendiola, Kostas Giotis

- *Towards an SDN-based Bandwidth on Demand Service for the European Research Community.* Alaitz Mendiola , Jasone Astorga , Jordi Ortiz , Jovana Vuleta-Radoicic, Artur Juszczyk, Kostas Stamos, Eduardo Jacob and Marivi Higuero (SDNFlex 16)

- *Deploying SDN in GEANT production network.* P. Luigi Ventre (University of Rome Tor Vergata, Italy); J. Ortiz (University of Murcia, Spain); A. Mendiola (University of the Basque Country, Spain); C. Fernandez (I2CAT, Spain); A. Pavlidis (National Technical University of Athens, Greece); P. Sharma (RENATER, France); S. Buscaglione (GEANT, United Kingdom (Great Britain)); K. Stamos, Mr (University of Patras and CTI & Technological Educational Institute of Patras, Greece); A. Sevasti (GRNET, Greece); D. Whittaker (Corsa Technologies, Canada) (IEEE SDN NFV 2017)

- *Leveraging OTT and ISP cooperation to enhance end to end QoS by exchanging valuable resources.* (ICUFN 2018). Michele Scarlato, Jordi Ortiz, Cristian Perra, Antonio Skarmeta.

- *Managing AAA in NFV/SDN-enabled IoT scenarios.* (GIoTS2018). Alejandro Molina Zarca, Dan García, Jorge Bernal Bernabé, Jordi Ortiz, Rafael Marín, Antonio Skarmeta.

Chapter 2

Problem Statement - Video in Future Internet

2.1 State of the Art

Human beings are visual creatures, "Video and audio are accepted as the most natural way to communicate" [16]. Even if visual information is important for humans, first photographies are from mid-1820s [17]. Video as we understand video nowadays, and omitting invents like Huygens' magic lantern, was firstly researched in 1880s and apart from the inclusion of sound and color small advances were introduced until the arrival of computers.

Computers, on the other hand, have evolved from simple calculators to something complex and powerful. Tasks performed by computers are heterogeneous since their conception, one of the roles adopted by computers is still and moving picture processing, that is video.

Computer networks appeared later as a consequence of the need of sharing information between devices. From the early 1970s [18] the idea of transmitting voice an video on packet networks like the Internet was there. Networks have been evolving to accommodate video and video codecs have been evolving to accommodate themselves to the available networks.

2.1.1 Video transmission

Once the transmission speed arrived to a minimum with which a page full of ASCII characters could be transmitted in a reasonable time, pictures have been transmitted [16] in

order to represent information in computers networks, starting by ANSI art representations in Bulletin Board Systems (BBS) in the early 1980s. Even the apparition of the HyperText Markup Language (HTML) observed the need for representing images as part of the information. Next evolution of visual data transmission due to enhancement of networking capabilities appeared in the form of audio and video transmission, the so called multimedia. From the first voice transmission using Network Voice Protocol (NVP) in the early 1970s in the ARPANET. The creation in 1980s of the 3Mpbs Wideband Satellite Network Protocol called the Stream Protocol (ST and later ST-II) [18] allowed the first video packet transmission thus creating the Packet Video Protocol. In the 90s the satellite network was replaced the terrestrial wide area networks that operated IP layer in parallel with ST and ST-II which were relegated to government and research networks mostly.

Contemporary to this early voice packet attempts, the digitalized version of telephone lines called Integrated Services Digital Network (ISDN) was deployed in conjunction of a set of video conferencing standards based around ITU recommendation H.320. At the same time in parallel with all the ITU[Pleaseinsert\PrerenderUnicode{âĂŹ}intopreamble]s Telecommunication Standardization Sector (ITU-T) ISDN based initiatives, the World Wide Web (WWW) phenomenon arose and popularized the audio and video content within web pages. RealAudio and QuickTime were the first approaches to multimedia delivery over the internet closely followed by Internet Engineering Task Force (IETF) standards such as Real-time Streaming Protocol (RTSP), Real-time Transport Protocol (RTP), Session Description Protocol (SDP) among others that raised the popularization of audio and video consumption over the Internet which was joined by intense efforts in evolving video and audio coding.

In general, any video file can be uploaded to a server with any file transfer protocol such as FTP and be downloaded by similar means, HyperText Transfer Protocol (HTTP) among others. Nevertheless, downloading a video file from a content server as it is and reproducing it afterwards is not considered Video Streaming. A video streaming or video on demand service in addition to providing with the data to the client, controls the delivery rate providing with real-time playing capabilities. In addition, it is considered that a streaming service should provide Video Cassette Recorder (VCR) capabilities meaning that the stream can be paused, rewind or fast forwarded (in case of non-live content). In the case of live video streaming, the content needs to be encapsulated onto the transport mechanism on-the-fly. The major difference between video on demand and live streaming is the absence of a prearranged end time [16].

Both video on demand and live streaming depend on the amount of available bandwidth to provide a real-time service. If the available bandwidth is cut down underneath the video

bit-rate being streamed, the Quality of Experience (QoE) perceived by user is reduced. To mitigate or even eliminate the drawbacks caused by network shortage, adaptation is performed. In the case of live streaming only transcoding, i.e, decoding and reencoding of the video source with different parameters, is possible unless a scalable video codec is used. With video on demand the possibility of encoding multiple formats and storing them, either in different files or in a multi-hint container is also possible. Adaptation might produce other drawbacks related to the switching between bit-rates such as reinforced artifacts or video quality degradation and upgrade switch, these changes can produce a loss in the QoE to a user which might prefer a constant reduced quality [19].

Interest in video transmission on the Internet comes from the early 90s [18]. This interest appeared partly as a consequence of the increase in computer processing power. This increase allowed real time capture, compression and transmission of video streams. The research in multicast IP and other related datagram transmission techniques also contributed to rise the interest in this field. RTP [20] was introduced by the IETF in the period 1992-1996 and became the sole transmission protocol for multicast conferencing systems. RTP not only allows data transmission but also managed data control. In addition to RTP as transport protocol, a solution for announcing multicast sessions was introduced, Session Announcement Protocol (SAP) [21]. In addition, to describe the transport streams the SDP [22] was introduced. In parallel with the multicast conferencing systems the WWW emerged. The increases in bandwidth in addition to the already mentioned increase in computing processing power allowed the inclusion of multimedia, audio and video, content into web pages. The pioneers in this field were Realtime and Quicktime. At that time (1998) the RTSP [23] standard appeared providing the VCR like control. RTSP relied on already existing protocols and followed a command signaling very similar to the already widespread HTTP. Other protocol worth mentioning is Adobe's Real-time Messaging Protocol (RTMP) [24].

At the same time Video and Audio coding techniques evolved. Video coding is considered to have 4 steps, partition, predict (subtract) transform and entropy encode meanwhile video decoding has the reverse 4 steps, entropy decode, inverse transform, predict (add) and reconstruct. Different partition, prediction and transform techniques have produced an evolution and wide variety of video codecs.

2.1.2 Codec evolution

Regarding source video compression, it is still common to find MPEG-2 and Advanced Video Coding (H.264/AVC) as the video codecs in use. But the main profiles of these codecs

were designed to cope with a fixed space-temporal video signal because they were conceived to be employed with guaranteed resources for transmission and decoding. Although the extended profiles permit some degree of scalability, they are seldom used because of the strong processing power that they compel. In video streaming, nowadays the prevalent choice to handle the adaptation to varying network conditions comprises the generation of several versions of the compressed video at different bit-rates and spatial dimensions. During the streaming session the server monitors the connection to decide which is the best version to use. The server can jump from one to another version at periodic points in the video time line Less frequent is the use of real-time video transcoding to perform a fine-grain adaptation. Both solutions involve a prohibitive amount of storage or processing resources at the server side.

The reference in video coding field were and continue nowadays being the codecs developed by ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG). Starting from MPEG-1 [25] Part 2 firstly released in 1993 which was rapidly continued by MPEG-2 [26] Part 2 also known as ITU-T H.262, released for the first time in 1995 and last amendment made in 2013, which became the standard de facto for video storage and transmission. Among its capabilities was to maintain a fixed rate that could be introduced into a Integrated Services Digital Network (ISDN) channel. The next remarkable evolution in video codecs appeared in 2003 (evolving until 2014) with the MPEG-4 [27] Part 10 H.264/AVC or ITU-T H.264 which is considered today's standard and in integrated in almost any device available. Other alternatives in video coding have appeared in this period of time. The VPx family created by On2 and purchased by Google to make it free for use with open source implementations. The Xiph Theora [28] family which was born completely Open or the BBC Dirac codec based in wavelets instead of the typical DCT approach.

In the present the two main video codec standards are High Efficiency Video Coding (HEVC)/H.265/MPEG-H [29] and Google's VP9 [30], both released in 2013. Both codecs focused in reducing the amount of bandwidth requested for a certain quality level in a 50% with respect to their predecessors H.264 and VP8 respectively. Other proposals for stereoscopic video and scalable video [31] [32] have been produced along the way. Some were related to enhanced visual experience such as Stereoscopic or Multi View Coding [33] techniques while others such as Scalability extensions were directed to enhance network and storage adaptability of the codecs.

2.1.2.1 Scalable Video Coding (SVC)

The glssvc [34] [35], an standardized extension of the well known H.264/AVC [36] video codec. Scalable Video Coding (H.264/SVC) enables the generation of a video coded bitstream from which a set of different video representations, defined by operational points in the spatial, temporal and quality video dimensions, can be extracted. Each of them is characterized by a bandwidth requirement. For streaming applications, the server can select the appropriate video representation to be delivered according to an updated description of the streaming context that includes the available bandwidth and the end device features (screen size, for example). The selected video representation is structured as a set of layers dependent in an incremental way. For video transmission, the set of video layers can adequately be split in different streams according to their importance in the hierarchical relationship. Scalable coding has a long tradition in compression standards. In a certain way, progressive modes of JPEG were a form of scalable coding, and MPEG-2 SNR, Spatial and High profiles were design to permit the generation of enhanced video.

A scalable bitstream consists of a number of layers, including a base layer and one or more enhancement layers [37]. The base layer can be decoded independently, and provides the elementary video quality which in turn is backward compatible with H.264/AVC. A higher quality can be obtained by decoding the base layer plus enhancement layers, improving the perception of the decoded sequence as more enhancement layers are used. There are three modes of video scalability: spatial scalability, temporal scalability and quality or Signal-Noise-Ratio (SNR) scalability.

When using spatial scalability, the base layer is coded at a low spatial resolution, and enhancement layers give progressively higher spatial resolution. With temporal scalability layers improve the frame rate. Take into account that temporal scalability is available in any video codec that enables a hierarchical structure of reference frames in a group of pictures (GOP), but H.264/SVC includes control headers to identify temporal layers. Quality scalability generates layers with progressively higher picture fidelity. The base layer contains a strongly compressed version of each picture, and enhancement layers incorporate more information to increase the SNR value. These three scalability dimensions can be combined together in a three dimensional coordinate space, as shown in figure 1. Note that a scalable video can use any combination of the three dimensions. For instance, it is possible to use quality and temporal scalability, and no spatial scalability at all.

H.264/SVC inherits from H.264/AVC the division of the codec design in two main blocks: Video Coding Layer (VCL), in charge of the source video coding, and Network Abstraction Layer (NAL), dedicated to the adaptation of the bitstream produced by the

VCL to networking or storage. For video streaming purposes only the NAL subsystem needs to be understood and studied since it provides enough information to identify the data pertaining to each layer. This is, precisely, the advantage of the VCL and NAL separation.

The elementary concept at the NAL level is the NAL unit (NALU). An H.264/AVC NALU contains a header followed by a raw byte sequence payload. This header is made of a unique byte, and includes a field to identify the type of NALU. A complete picture can be segmented into multiple NALUs each carrying usually a coded slice or control information. The set of NALUs needed to decode a complete picture is called access unit. H.264/AVC introduces a special type of access unit called Instantaneous Decoding Refresh (IDR), which can be identified through the inspection of the NALU header. A sequence of access units starting with an IDR can be decoded independently of any previous pictures in the bitstream. The maximum NALU size can be configured as a parameter of the encoder, in order to be adjusted to the maximum transmission unit (MTU).

H.264/SVC is backwards compatible with H.264/AVC. The base layer of a scalable video is represented with a set of NALUs that can be decoded by any compatible H.264/AVC decoder. Additional NALU types are defined by H.264/SVC. The main characteristic of these new NALUs is the use of a sequence of three bytes, following the H.264/AVC header, that contain three identifiers: temporal identifier (TID), dependency identifier (DID) and quality identifier (QID). These identifiers represent a point in the temporal, spacial and quality scalable dimensions respectively. The inspection of these fields permit to identify NALUs belonging to a specific enhancement layer. H.264/SVC access units start with base layer NALUs followed by enhancement layer NALUs, organized with increasing values in the (DID, QID, TID) triple identifier. The exact number of NALUs per access unit depends among others on the number of enhancement layers. Figure 2.1 shows the structure of H.264/SVC and the relations between identifiers and layers as well as the relation with H.264/AVC.

A special H.264/SVC control NALU at the beginning of the bitstream contains metadata about the scalable structure of the video. It is the Scalability Info, a description of the enhancement layers of the video that includes the resolution of the spatial layers, the frame rate of the temporal layers, and the average bandwidth required to transmit the video up to each layer. This information can be used at different control points in the streaming system in charge of doing rate shaping of the video. It could be the streaming server or a Media Aware Network Element (MANE) located at the frontier between two network domains, one with a big bandwidth and the other with a more limited one. No transcoding is necessary, but a simple discarding of NALUs. Dynamic switching between

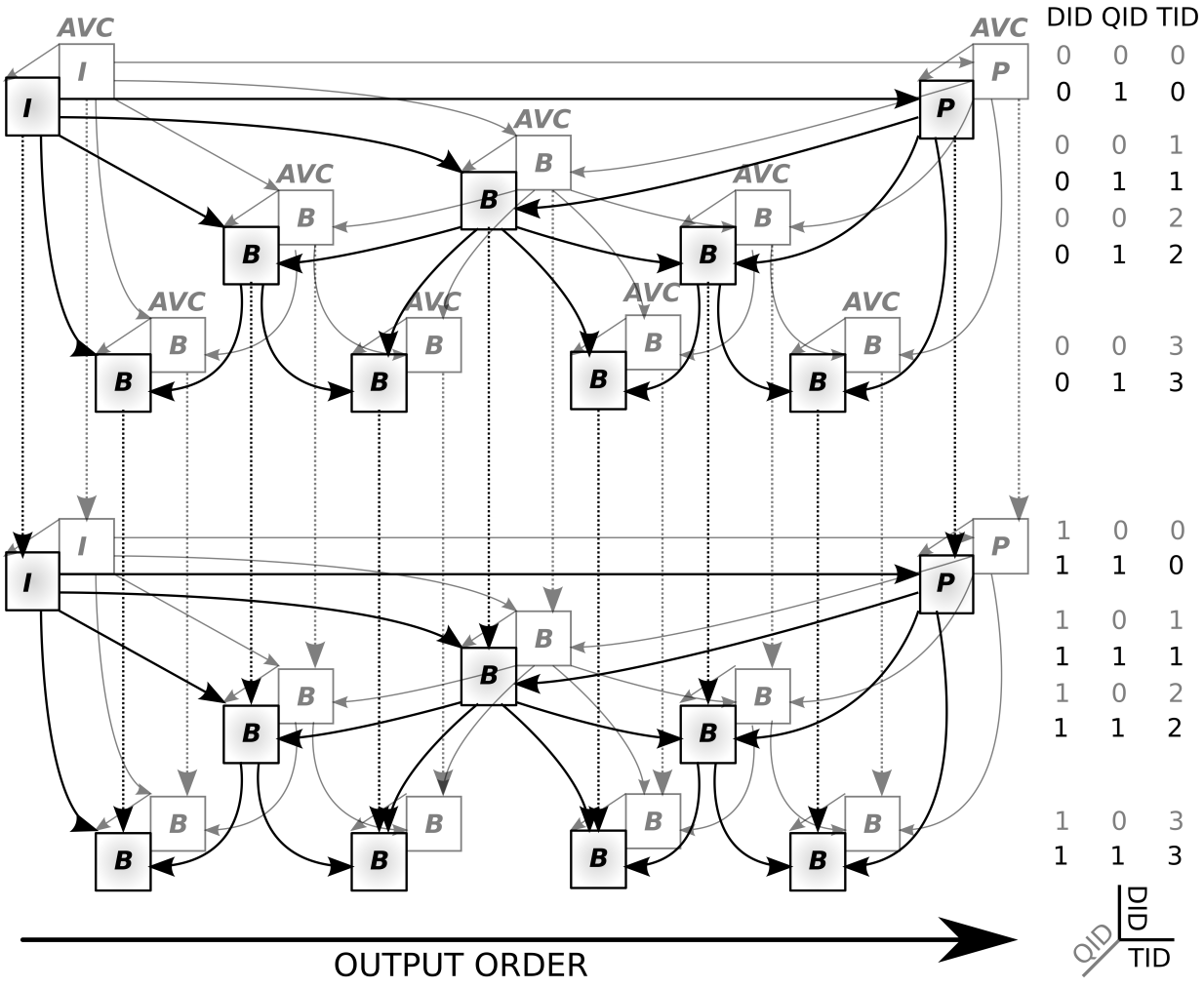


Figure 2.1: H.264/SVC Structure and dependencies

enhancement layers is possible during the streaming session at IDR access units. Scalable information can also be used when doing Forward Error Correction (FEC) or any other technique to improve the robustness of the more sensitive data. The base layer can be transmitted with more care than any other layer, as it is the more important piece of information required to perform a correct decoding.

2.1.3 HTTP

The Internet has become one of the best ways to exchange information all over the world. Part of its success is associated to the introduction of the Hypertext Transfer Protocol [38] (HTTP) that evolved in what is today known as the abstract concept World Wide Web (www). Many computer communication protocols have been designed for specific tasks but the most versatile has demonstrated to be HTTP. HTTP is transported using the Transmission Control Protocol [39] (TCP) which was designed having network congestion avoidance in mind when network bandwidth was a scarce resource.

Nowadays video distribution is considered one of the most bandwidth consuming services. (Cisco, 2015) points out that "Globally, consumer internet video traffic will be 80 percent of all consumer Internet traffic in 2019, up from 64 percent in 2014" [40]. Video streaming services have been historically addressed by specific protocols and technologies. With the growth of the home broadband services there is a new trend into using HTTP as transport protocol for multimedia services. One technical reason to accept this solution is the consideration that (Wang, 2008) "TCP generally provides good streaming performance when the achievable TCP throughput is roughly twice the media bitrate, with only a few seconds of startup delay" [41], [42] which is effectively true with the increase of home bandwidth.

With the increase of available bandwidth in the last mile, the need of adjusting the stream bit-rate to the available bandwidth has become less relevant. It is considered that having twice the bit-rate available as bandwidth using Transmission Control Protocol (TCP) (and thus HTTP) is sufficient for achieving good performance with a few seconds delay [43]. Providing multimedia content over HTTP [44] has some inconveniences related to suboptimal storage, exceeded bandwidth consumption or complex bit-rate logic, nevertheless the advantages regarding simplicity in deployment, economics and scalability have surpassed the inconveniences for the industry. In this approach the server becomes a passive element just offering the content and some metadata that allows the client to decide which resources need to be retrieved, in which order and at what moment of time. The approach is very attractive since any HTTP deployment is easily converted in a

Video Streaming deployment without specific infrastructure. As a counterpart control on bandwidth consumption of the provider is lost.

In 2009 Apple introduced their HTTP Live Streaming (HLS) [45] solution, Microsoft proposed the Microsoft Smooth Streaming meanwhile Adobe proposed the HTTP Dynamic Streaming and finally on the standardization bodies we can find the Adaptive HTTP Streaming (AHS) from the 3GPP and the MPEG Dynamic Adaptive Streaming over HTTP (DASH) [46] [47] [48]. Each of these systems are based on their own manifests and file formats. Apple system uses m3u playlist and video must be encoded with MPEG2-TS. Microsoft requires a server manifest and a client manifest file and defines a smooth streaming format based on the ISO Base media File Format [49]. Adobe makes use of their own Flash media manifest and F4F file format (also based in ISO Base Media File Format. The 3GPP proposal (AHS) defines a Media Presentation Description (MPD) acting as manifest file and also extends the ISO Base Media File Format for storing the media. Finally MPEG proposal (DASH) adopted the 3GPP's MPD description file as a starting point while media segments must be compatible with the ISO Base Media File Format (ISOBMFF) [49]. As can be easily seen there are two common factors in each approach. The definition of a manifest/session file and the compliance with the ISO Base Media File Format.

These HTTP based streaming systems can take advantage of any existing and/or incoming enhanced distribution mechanisms designed for standard HTTP content including caching systems, proxies, load balancers, ... and off course any Uniform Resource Locator (URL) based enhancement.

2.1.3.1 DASH

The Dynamic Streaming over HTTP or DASH is the proposal made by the MPEG to the trend of HTTP based streaming. The idea behind these HTTP based streaming mechanisms is to produce metadata with information about the media to be streamed and the media itself so that can be served by standard HTTP servers with the subsequent advantages in terms of deployment, taking advantage of all the optimization mechanism that have been researched since its definition.

The standard [46] in particular takes advantage of the MPD as defined by the 3GPP and the ISOBMFF. Each mpd file has information related to the bitstream which can be in a single file and be accessed by means of the byte-range HTTP header, a segment list (each segment indicating the full-url), a time based segment list in which each url has a reference of the starting time or finally a numbered/sequential segment list. Listing 2.1

shows a typical MPD file, in this case one corresponding to sequential segment list with just one representation, it must be noted that multiple representations can be defined in the same MPD file and switching between representations is allowed to the clients, what that switching implies for the codec is out of the scope of this thesis but suffice to say that codecs like H.264/AVC define the switching frames to be able to change video quality without the need of resetting the decoder. Among the parameters shown, there are some which are quite important for the player since help the decision algorithm within the DASH client which representation to select or when a particular chunk should be downloaded. Line 7 defines the total duration of the stream while line 8 defines the minimum quantity of data corresponding to the value in time units to be stored before start playing, the so called buffering time. Frame rate and frame size are part of the representation while an adaptation set containing multiple representations has the maximum frame rate and frame size values among all the possible representations.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2    <MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xmlns="urn:mpeg:DASH:schema:MPD:2011"
4      xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011"
5      profiles="urn:mpeg:dash:profile:isoff-main:2011"
6      type="static"
7      mediaPresentationDuration="PT596.458S"
8      minBufferTime="PT2.0S">
9        <BaseURL>http://serverexample.domain/path</BaseURL>
10       <Period start="PT0S">
11         <AdaptationSet bitstreamSwitching="true"
12           mimeType="video/264"
13           startWithSAP="1"
14           maxWidth="640" maxHeight="360"
15           maxFrameRate="24"
16           par="16:9">
17           <SegmentBase>
18             <Initialization sourceURL="isobmffbasechunk.avc"/>
19           </SegmentBase>
20           <Representation id="0" codecs="AVC" mimeType="video/264"
21             width="640" height="360" frameRate="6"
22             sar="1:1" bandwidth="390965">
23               <SegmentList duration="12" timescale="6">
24                 <SegmentURL media="isobmffchunk0.avc"/>
25                 <SegmentURL media="isobmffchunk1.avc"/>
26                 <SegmentURL media="isobmffchunk2.avc"/>
```

```
27                  <SegmentURL media="isobmffchunk3.avc"/>
28                  <SegmentURL media="isobmffchunk4.avc"/>
29                  <SegmentURL media="isobmffchunk5.avc"/>
30                </SegmentList>
31              </Representation>
32            </AdaptationSet>
33          </Period>
34        </MPD>
```

Listing 2.1: Typical DASH MPD file

The server does not necessarily have any logic to support dash streams, although it is not forbidden. Usually, the intellect in this system falls on client shoulders. This approach has introduced a new field of research in caching algorithms among others that now are directed by the view of the network of the client or even user-driven as is happening in most Video on Demand (VoD) services that allow the user to change the desired quality of the stream without the need of a signaling channel to the server side, contrary to the old fashioned RTSP way among others.
This introduction to DASH does not intend to be complete and readers are pointed to bibliography for more details, nevertheless the basic concepts needed for understanding the rest of this thesis have been introduced.

2.1.4 Content Delivery Network (CDN)

'A CDN is a collection of network elements arranged for more effective delivery of content to end-users' [50].
Nowadays content delivery, with emphasis in video streaming, is the major source of bandwidth consumption in the Internet, so efficient and effective content distribution is a key aspect to deploy bandwidth demanding services at large scales. As a solution, Content Delivery Networks (CDNs) are being deployed and offered to content providers and carriers as cost saving solutions, but they add undesired complexity to end-to-end operations and content state management, becoming too closed and hard to adapt to new workflows.
The so called Content Delivery Network (CDN) services aim to enhance network performance as well as perceived Quality of Service (QoS) by replication of content into net- work edges caches maintaining the content correctness.
Those solutions are also limited by the strict environment established by underlying architectures, which are typically tied to IP and conducting to the ossification of networks. CDNs appear as a consequence of growth in network traffic and the adoption of

broadband networks. The latter increases the expectations by end-users that frequently got disappointed by the lack of Quality of Service (QoS) produced by single point congestion, the web server. Content providers see the Internet as a transport for their contents and suffer from third parties issues such as Internet Service Provider (ISP) congestion.
To mitigate the downgrade in the perceived service quality some actions have been historically taken.

- Enhance the web server hardware, which has effect to a certain extent but is limited and might finally result in completely migrating to new hardware, not to speak about the costs related to this alternative. Server farms are another possibility but still lack the proximity to the end-user since they are usually collocated in the content-provider premises.

- Deployment of proxy caches, usually done by the ISP or the provider, geographically distributed and usually referred as hierarchical caching. In the case of ISP proxies all the traffic for a end-user goes through the same proxy. This provides enhancements in terms of bandwidth saving and performance improvements. In addition, the caching is based on client demands which might potentially turn back the bottleneck to the origin, despite being mitigated.

To overcome these issues, the CDN concept appeared in the 1990s having among their functionalities:

- Manage redirections to the closest surrogate server/proxy/cache to take profit of location.

- Content replication in a pull model, where the CDN can replicate content by monitoring the content provider or in a push fashion.

- Offer differentiated services to specific users or groups of users.

- Management services of network elements and caches among others, as well as, accounting services.

CDNs outperform standard content delivery by strategically deploying content replication network elements in widespread locations. The CDN can be seen as a new virtual overlay on top of the application layer since it sits on top of protocols as HTTP or RTSP.
Although there is no official CDN generation distinction, it is considered an evolution in CDNs from static and dynamic content provision on the first generation to video and

multimedia content in general on the second phase. Some might also say that the third generation is the social media, although is it not different from the formers, and the generation to come to be the Internet of Things (IoT).
Among the desirable business characteristics of a CDN, scalability, security, reliability, responsiveness and performance are accounted. One of these characteristics is of great importance and not trivial for the acceptance of CDNs by the industry, security. Apart from ensuring the stability of the service against massive attacks such as Denial of Service (DoS), a CDN needs to ensure the content-provider that the content is not accessible by illegitimate customers nor corrupted and in the other hand ensure the customer that the content received is legitimate. There is a 3 point trustiness relation that has been accepted but implies in most cases inconsistencies such as a third partie (the CDN) being presented to the customer as the content-provider. This relation has introduced important controversy mostly speaking about HTTP over TLS (HTTPS) and the certificate management.

2.2 Stream Control Transmission Protocol (SCTP)

SCTP is a general purpose transport protocol initially conceived for conveying telephony signaling messages over IP best-effort networks. The baseline SCTP, defined in RFC 4960 [51], offers a transport service that aims to solve some well-known problems of TCP that affect the performance of highly delay constrained services. This section is dedicated to the description of the main features of SCTP and Concurrent MultiPath Transfer for Stream Control Transmission Protocol (CMT-SCTP). For a complete description of the protocol, the reader is referred to [52]. An excellent survey about the SCTP research can be found in [53].
SCTP is located at the transport level in the Internet network architecture, and works directly over IP. It permits to use a connection oriented service, similar to that of TCP, but bypassing some of its limitations such as the head-of-line blocking or the restriction of using single-homed connections. Hence, SCTP offers an standardized alternative to the implementation of application-dependent reliable data transfer protocols on top of UDP.
The protocol employs the concept *association* to manage the relationship between two *endpoints*. At a given time, only one SCTP association can exist between two SCTP endpoints. The association is established with a four-way handshake based on a cookie mechanism that protects against synchronization attacks. An SCTP endpoint on a multi-homed host is a combination of transport addresses from which SCTP packets can be received. All transport addresses involved in an SCTP endpoint have the same port but

can use multiple IP addresses. Participating addresses are notified to the other end during the association setup. This multi-homing feature is used as a fault-tolerant mechanism for the association. An SCTP peer monitors which of the addresses at the other end are available for receiving user messages by means of special control messages called *heartbeats*. Responding addresses are considered *active* and thus available for communication. One of them will be used as the *default* destination, so that there will be a *primary path* between both ends, but if there is a failure detection - the primary address stops sending back acknowledgments - a different destination address can be used to generate a backup path, and the communication can continue.

Additionally, SCTP permits to bind user messages to *streams* in the context of an association. During the association setup, both ends negotiate the number of streams that will be used. A stream is a unidirectional sequence of messages that SCTP must deliver in order to the upper layer at the receiving end. Streams are implemented with stream identifiers and stream sequence numbers (SSN) that are attached to user messages. When a transmission error occurs, the negative effects will be restricted to the streams involved in the packet loss. As a consequence, the increase in the delay due to retransmissions (head-of-line blocking) will not affect other streams. SCTP permits sending messages that bypass the normal ordered delivery. These messages are delivered as soon as they are received.

SCTP packets contain a common SCTP header, followed by one or more *chunks*. A chunk may carry user data or control signaling, and a single SCTP packet may transport user data associated with multiple streams. In order to implement acknowledgments and loss or duplicate detection, data chunks have a unique Transmission Sequence Number (TSN) that is maintained at the association level. Regarding the size of user messages, SCTP may fragment a message that causes the packet to exceed the path MTU. The receiving end will reassemble fragments in order to keep the message boundaries when data is finally delivered to the upper layer. As a consequence, SCTP is a *message oriented* protocol, similar in this aspect to UDP, instead of a stream oriented protocol such as TCP.

SCTP employs a *flow control* mechanism that is nearly the same as the one used in TCP. The algorithm is based on the use of a *receiver window* (rwnd) variable at the sender side. This variable gives an indication of the available buffer space at the receiver end. During the association setup, each endpoint notifies its reception buffer size, and rwnd is initialized with this value. When a data chunk is transmitted, the sender endpoint subtracts the size of the data from the current value of rwnd. A special type of control chunk, called Selective Acknowledgement (SACK), is used by the receiver to inform about data reception. At the sender end, SACKs are processed to check if the rwnd must be increased. Three pieces

of information included in the SACK are used to update the rwnd value. The first one is called *advertised receiver window* (a_rwnd), and it indicates the available space in the reception buffer. As the receiver buffer is filled with data chunks, the a_rwnd variable is decreased, and it is increased when chunks are delivered to the upper level. The second one is the cumulative TSN ack, that is, the largest TSN received in sequence. And the third piece of information is a list of blocks of consecutive data chunks that have been received after a gap of missing chunks. When a SACK is received, the cumulative TSN and the list of gap blocks are used to calculate the amount of *outstanding bytes*, that is, the total size of chunks already sent and not yet acknowledged. The rwnd variable is updated with the a_rwnd value minus the outstanding bytes. At any time, if rwnd equals zero, the sender stops sending data. Nevertheless, if there is no congestion, the sender can have one outstanding data chunk per round trip time to force the reception of a_rwnd updates.
The *congestion control* algorithm used in SCTP employs the well known strategy called *additive increase and multiplicative decrease* used in TCP, but with some modifications needed due to its multi-homing nature. The congestion control uses three variables to regulate the transmission: the *congestion control window* (cwnd), the *slow-start threshold* (ssthresh) and the *partially acknowledged bytes* (partial_bytes_acked). The main difference between the congestion control algorithms used in TCP and SCTP stem from the fact that SCTP uses separate cwnd and ssthresh variables for each of the destination addresses, and as a consequence, the congestion control is applied independently to each path. The algorithm has two main states per destination: *slow-start* and *congestion avoidance*. The initial slow-start state probes the path to determine the available capacity. The cwnd variable keeps an approximation of the amount of data that can be injected into the path before causing congestion. This value is used to limit how much outstanding data can be flying to the destination address. The amount of outstanding data is called *flightsize*. At any given time, the sender must not transmit new data if the flightsize is greater or equal to the cwnd of the destination. During the slow-start phase, the cwnd have a minimal value of one MTU and the ssthresh could be initialized to the receiver window. The value of cwnd increases when an incoming SACK advances the cumulative TSN point. Take into account that SCTP associations can have multiple destination addresses, and as a consequence a SACK received from one destination address may acknowledge data sent to other addresses. If some of the acknowledged data was sent to a destination address, and the congestion window is being fully utilized (the flightsize is greater or equal to the cwnd) the sender increases cwnd proportionally to the amount of acknowledged data sent to that destination. When the cwnd reaches the ssthresh, there is a transition to the congestion avoidance state. During congestion avoidance, the cwnd is incremented only by one MTU

when the congestion window is fully utilized and a SACK increases the cumulative TSN point. The auxiliary variable partial_bytes_acked helps in the implementation of this mechanism, counting the amount of bytes acknowledged since the last update of the cwnd variable.
The multiplicative decrease of the congestion window is activated when a packet loss is detected. One of the mechanisms to detect packet losses is based on the retransmission timers. Each time a data chunk is sent to a destination address, the retransmission timer of that destination is initialized with the value of the estimated Retransmission Timeout (RTO), computed with the smoothed round-trip time and variation of the corresponding path. If the retransmission timer expires, SCTP assumes a severe congestion problem causing the congestion control to go back to the slow start state. Additionally the ssthresh is reduced to a half of the cwnd, and the cwnd is reset to one MTU. On the other hand, if the loss is detected with the inspection of gap blocks in a received SACK, both the ssthresh and the cwnd are reduced to a half of the previous value of cwnd. If three consecutive SACKs report the same missing TSNs, *fast retransmission* is activated. In this case, the sender determines how many chunks marked for retransmission fit in a single packet, and this packet is sent ignoring the value of cwnd and without delay. Then the transmission enters in *fast recovery* mode and the highest outstanding TSN is marked as the fast recovery exit point. While in this mode, the ssthresh and cwnd should not be reduced due to subsequent fast recovery events. The fast recovery mode is exited when all TSNs up to and including the exit point are acknowledged.

2.2.1 CMT-SCTP

CMT-SCTP is a proposed extension for SCTP designed to enable simultaneous data delivery through all the available paths between a pair of source and destination endpoints. The extension, proposed by researchers from the University of Delaware, is fully described in [5]. The goal pursued by CMT-SCTP is the increase in fault-tolerance and in global transmission performance with respect to the baseline SCTP, where only one path is used at a time. The extension is designed under the assumption that the bottleneck queues on the end-to-end paths are independent. This assumption is important to maintain a TCP-friendly flow and congestion control over all the available paths.
CMT-SCTP schedules the transmission of new data through the available paths as soon as the corresponding cwnds allow it. When there is space for transmission simultaneously for several destinations, data is sent uniformly through all of them. However, taking into account that each path has different delay and bandwidth characteristics, this simultaneous

transmission incurs in a significant packet reordering at the receiver end. This effect, connatural to CMT-SCTP, provokes a substantial degradation in the performance of the basic algorithms of SCTP. In [5], three negative effects of packet reordering are identified: unnecessary fast retransmissions triggered by an increase in the number of gap reports; a slow increase of cwnd due to the lack of cumulative TSN acknowledgements; and an increase of acknowledgment traffic due to immediate delivery of SACKs when out-of-order data is detected at the receiver end.
CMT-SCTP uses three new algorithms to cope with these problems. The *Split Fast Retransmission* (SFR) algorithm keeps an independent virtual queue for each path within the retransmission buffer. Additional variables associated with each virtual queue permit to apply the fast retransmission procedure on a per destination basis, keeping track of the path that was used to transmit each TSN. The *Cwnd Update for CMT-SCTP* (CUC) algorithm allows to increase the destination cwnd with SACKs that do not advance the cumulative TSN point. The algorithm tracks the earliest outstanding TSN per path and updates the path's cwnd just with the information contained in gap blocks, even if the cumulative TSN is not modified. The *Delayed Ack for CMT-SCTP* (DAC) algorithm is designed to reduce the acknowledgment traffic. The baseline SCTP protocol is designed with the assumption that data received out-of-order indicates possible loss, and accordingly the receiver should immediately send a SACK with a gap report to trigger fast recovery as soon as possible. The DAC algorithm specifies a different receiver and sender behavior to infer when a gap report is caused by loss and not reordering. In [5], the authors indicate that the combination of the three mentioned algorithms improves the aggregate cwnd growth in comparison with multiple SCTP associations (one per path) between the sender and the receiver. The same article explores different retransmission policies and concludes that it is beneficial to send retransmissions to the destination with the largest cwnd or ssthresh, instead of using the same destination employed in the initial transmission.

2.3 SDN

The past recent years have witnessed the birth of a new paradigm in networks. The so called Software Defined Networking (Software Defined Networking (SDN)) have evolved from a promising technology enabler to a production ready solution already deployed in several environments.
'SDN was developed to facilitate innovation and enable simple programmatic control of the network data-path' [54]. OpenFlow Protocol (OpenFlow) is considered the industry's

favourite approach for SDN by means of its control plane and data plane separation and its ability during its early years to accommodate existing hardware.
But SDN is not a solution designed explicitly to address a certain problem but a solution emerged from a evolution in networking history. To understand how SDN has become so rapidly a key technology, the history of programmable networks as a way to facilitate network evolution needs to be revisited.

2.3.0.1 First approaches to externally control the data-plane

In the last decade of last century, computer networks were formed by network devices exchanging information with their neighbors. That information was needed by the running algorithms to take the decisions that finally were translated to hardware instructions which at last made the information flow.
The devices took decisions based on their local perception on the network but were unable to see the big picture. Network operators on the other hand were willing to gain some network-management capabilities that weren't available at the moment like traffic engineering with which they would be able to mitigate network problems thanks to the operator heuristic knowledge.
The increase of link speeds and congestion in backbone networks those days lead vendors to implement traffic-forwarding logic directly in hardware. In addition, the hardware and the software (firmware) of the devices was tightly tied to the protocols accepted by the device which in turn slowed down the evolution of protocols and the deployment new network solutions since it was up to the device vendors their implementation and deployment. Some alternatives to split control and data planes started to arise.
Giving access to the hardware implemented data plane capabilities to a remote entity allowed to extract the control protocols to commodity hardware which was also increasing its capabilities dramatically those days.

Open Signalling

Around 1995 the Open Signalling Working Group (OPENSIG) [55] [54] Working Group realised that, in order to speed up network equipment deployment in ATM networks, the vertically aligned closed switches should be able to expose their network hardware by means of open programmable interfaces. As a consequence of the group research, the General Management Switch Protocol (GSMP) was proposed. The aim of GSMP was 'simply' to control a label switch, meaning operating resource reservation, port and connectivity configuration, etc... the Working Group stayed active until 2002.

Active Networks

In parallel (1997) with OPENSIG, the Active Networks initiative appeared. Network research those days was based on small experiments in laboratories that were lately verified with simulations of bigger networks and if funding and interest was continued, the proposal was submitted to Internet Engineering Task Force (IETF) to potentially become a standard. The process was tedious and some researches tried to apply the easiness in reprogramming similarly to software development in stand-alone PC to networks.
In addition of the device programmability capabilities introduced by OPENSIG, the concept of 'capsules' is introduced. The capsules were defined as small fragments of code carried in user messages, that were intended to be interpreted by 'Active' routers which would coexist with 'standard' routers. The key technology enablers to trigger the Active Networks appearance were the reduction in computing costs and the appearance of portable code such as java, as well as virtual machine concepts that allowed sandboxing processes in the device.
Active Networks didn't attract the attention of the industry (with the exception of DARPA) but came ahead of their time in terms of innovation introducing the foundation key concepts for SDN:

- Network function programmability thus allowing innovation. In the case of Active Networks the focus was on data-plane programmability while later work has been focused on control-plane to regain interest in data-plane lately.
- Network virtualisation and program demultiplexion by means of specific packet headers.
- Focused on middle-boxes and how to compound functions through the network.

On the other hand, it was not clear at that time where Active Networks should be applied, they were expensive (since active routers which were dedicated hardware needed to be deployed) and had some problems related to security and interoperability. In addition, the use of capsules implied a user with a programmer role which is not always the case.

2.3.0.2 Extracting the control-plane from networking devices

In the early 2000s it was clear that that network-management was something desirable and to be researched. The new focus was on network administrators and how to simplify their lives while expanding their capabilities. In that sense, providing control-plane programmability and de-localize the control plane decisions from a device scope to a network scope making them wiser looked like the path to follow.

Forwarding and Control Element Separation (ForCES)

The IETF with the ForCESworking group proposed an open standard interface to the data plane to enable innovation in control-plane software. To that intent a framework [56] and a protocol [57] were defined so that the control-plane and the data-plane could be logically separated which in turn makes them available for physical separation. If the interface to communicate the control-plane with the data-plane is open and standard effectively allows multi-vendor scenarios.

In ForCES terminology any device in the network is considered a Network Element (NE) that is composed of one or more Forwarding Elements (FE) as well as one or more Control Elements (CE) that collaborate together to produce the provided capability. The FEs compose the data-plane and the CE compose the control-plan. Apart from the FEs and CEs which are considered part of the NE, the ForCES framework defines two new entities, the 'CE manager' and the 'FE manager' which may or may not be located outside the NE. The CE manager is responsible of determining which FEs will control a FE. A FE manager, is responsible of determining to which CE a FE will communicate. There are no restrictions on how the managers take their decisions giving place for innovation.

The ForCES protocol was designed for the signaling between a FE and a CE and although is dependent on the communication between the managers and the FE/CE, it is not employed on that communication. The ForCES protocol is able to transport packets from the data-plane (received in FEs) and intended for control such as Border Gateway Protocol (BGP). The FEs are able to identify the control messages and redirect them to the proper CE, since all the CEs do not necessarily implement all the control protocols but rather a subset of them.

Unfortunately ForCES didn't attract the attention of network vendors since it produced a breach from what was actually deployed.

Routing Control Platform (RCP)

Similarly to ForCES, RCP [58] was aimed in controlling the data-plane of the device but contrary to the former no protocol was defined but an already existing one was used, BGP. Therefore, RCP is limited to control layer 3 networking, meaning Internet Protocol (IP). The idea is to install forwarding-table entries in legacy routers. Therefore a RCP is a single entity per Autonomous System (AS) that decides the values the routing tables must have thanks to having a full knowledge of the network topology. The proposal specifies three steps: replacing Internal Border Gateway Protocol (iBGP), act as External Border Gateway Protocol (eBGP) endpoint and finally being able to change inter-AS routing by employing RCP instead of routers in order to exchange routes via eBGP or any other protocol existent or to be invented.

In addition, higher level policies could potentially be implemented in RCP which would mitigate the unexpected results of BGP attributes employed until then to that intent.
This logically centralized entity, RCP, needs to be reliable and physically distributed to cope with possible failures. The challenges identified for RCP centralized controller will accompany data from control separation centered network evolution history.

Network Configuration Protocol (NetConf)

Firs time proposed in 2006 [59] and revisited in 2011 [60], the IETF NetConf protocol is intended for configuring network devices, it is based in eXtensible Markup Language (XML) and acts on the devices via Remote Procedure Call (RPC).
NetConf is arguably considered SDN or programmable network. First of all, data-plane and control-plane are not separated, to continue with, it does not intended to micro-manage state of the device, the primary role of NetConf is change device configuration which will evolve the state but not directly influence on it. The working group also defined the YANG data modeling language to model state and data configuration, RPC calls and notifications. Despite its limitations, it highly accepted by vendors and is becoming more and more usual in controllers southbound interfaces as a mean to influence in network. Thinking in proactive network programmability NetConf still has some usage.

Ethane: A Protection Architecture for Enterprise Networks (Ethane)

Trying to rise the abstraction level in enterprise network-management a step further, Ethane focused on converting policies and security directives directly to network rules. To achieve that objective, Ethane, simplifies the network elements to simple flow tables that receive the flows to be installed from a central entity, the controller, that is able to understand high-level directives and convert them to low-level instructions.
Ethane is considered the predecessor of OpenFlow, appearing only one year (2007) before the latter, and established the foundations of what today is understood as the SDN paradigm.

2.3.0.3 Network Operating System (NOS) era

The separation of control-plane and data-plane and extracting the former from the network elements lead almost unavoidably to the appearance of network controllers. These initially simple controllers have been evolving unstoppable to follow the computer operating system analogy by offering a base layer on top of which applications can be deployed to produce added value. In addition, the drivers approach is replicated by offering what is commonly known as southbound protocols that allow enforcing the application operations on different network elements as can be seen in Figure 2.2.

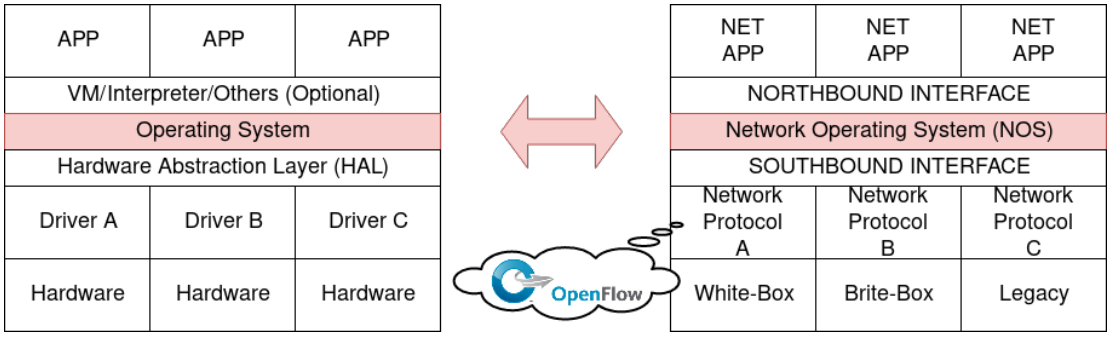


Figure 2.2: NOS to pc operating system analogy

One of the most widespread southbound protocols for NOS is OpenFlow and many would say that NOS were born because of OpenFlow. Indeed, as the successor of Ethane the OpenFlow switches are so simple that are considered useless unless they are managed by a controller/NOS.

OpenFlow

The OpenFlow [61] protocol was born as a result of Stanford's Clean Slate Program while trying to provide an experimentation environment comparable to real life but at the same time at a tractable scale, their objective was set on to the campus networks.

Two are the key factors why OpenFlow has become so widespread, its capability to enable and absorb existing hardware and the versatility of its proposal covering from layer zero to layer 4 in the Open Systems Interconnection (OSI) model. This versatility attracted immediately the attention of the research community eager to experiment with clean slate networks to produce new solutions that can finally break the Internet ossification [62] [63].

Motivation to design and propose OpenFlow was 'As researchers, how can we run experiments in our campus networks?' [61]. Authors stated that there were two approaches to achieve switch programmability through open interfaces those days: Asking the vendors which was unlikely to happen, since that would lower the barrier-to-entry for new competitors or use open software platforms that provided low performance, low port density or were extremely expensive for the purpose.

With the following objectives in mind:

- High-performance and low-cost implementation
- Unrestricted research capabilities
- Production and research traffic isolation

the authors proposed OpenFlow as an open protocol [64] to program different switches

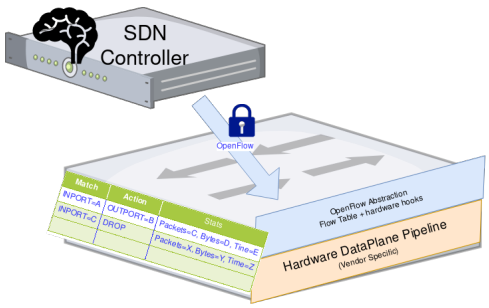


Figure 2.3: OpenFlow concept.

HARDWARE (802.1)		ETHERNET (802.3)			IP			TCP	
In Port	VLAN ID	SOURCE ADDRESS	DESTINATION ADDRESS	TYPE	SOURCE ADDRESS	DESTINATION ADDRESS	PROTOCOL	SOURCE PORT	DESTINATION PORT

Figure 2.4: Match fields for OpenFlow "Type 0" switch.

and routers. The idea is to extend existing hardware functionality by taking profit of already existing flow-tables (which have general common similarities regardless the vendor) providing access to create, delete and modify entries from the flow-table.
So OpenFlow requires three components to be present on a switch:

- OpenFlow protocol implementation that is responsible from parsing and creating OpenFlow messages employed for the communication with the controller.
- A secure channel between the switch and the controller with which the OpenFlow messages are transported.
- Flow tables that associate a flow or match pattern with an action to be performed. Initially three basic actions were defined, *forward* the packet to a port, *encapsulate*the packet and send it to the controller and *drop* the packet silently discarding it.

Apart from matching patterns and actions, the flow-tables store statistics for each flow-table entry storing amount of data and time that the flow was idle. This data can be useful not only for the straightforward maintenance of the flow table by removing unused entries but also for changing network behavior based on the values or for future forensics analysis.
The authors studied pipelines and located a common set of fields for the matching pattern that were present in general in any vendor device. These field became the T̈ype 0ïheader fields (see Figure 2.4) for the first version of OpenFlow.
The OpenFlow protocol has continued evolving being its last version the 1.5.1 [65] and 1.6

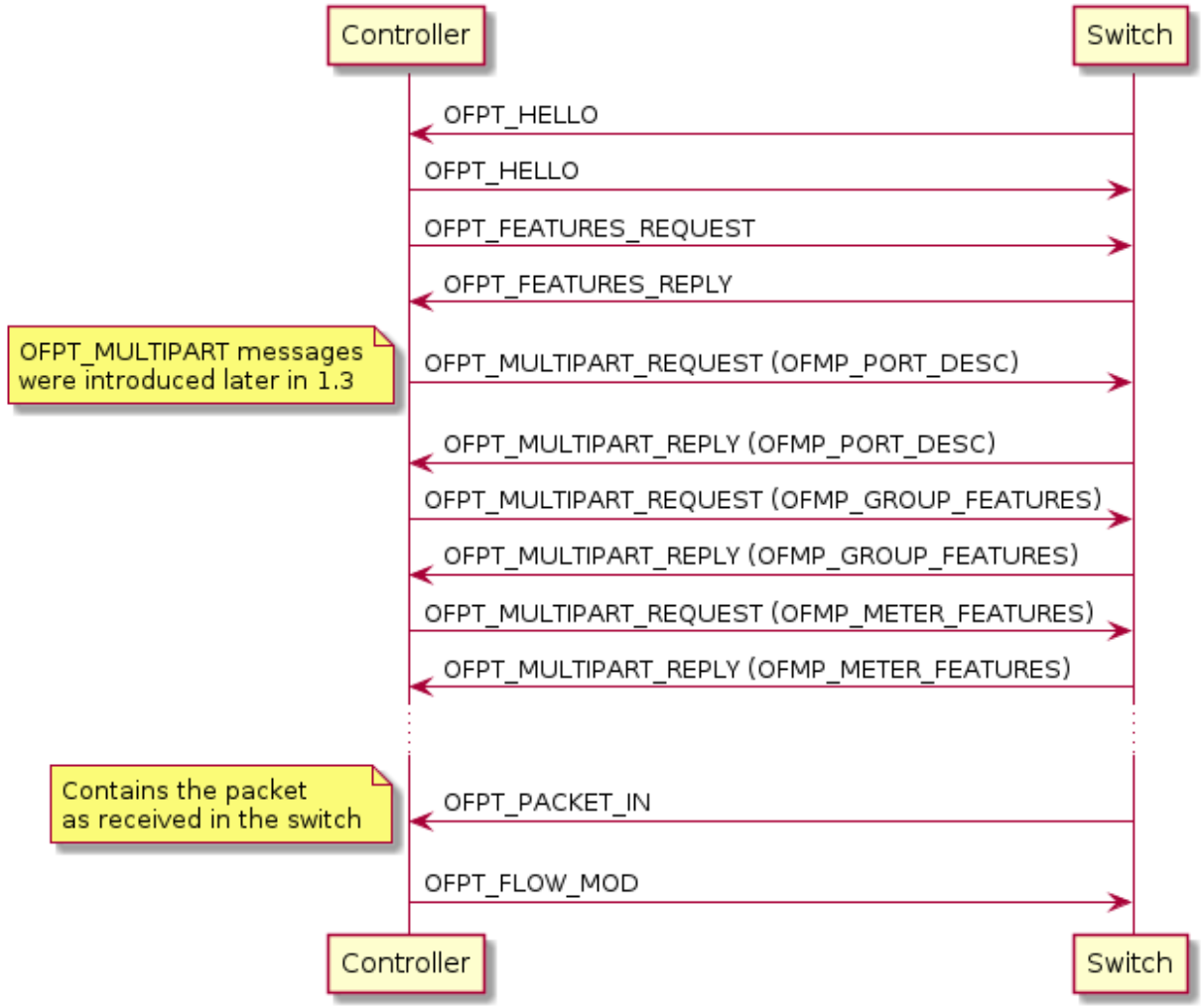


Figure 2.5: OpenFlow sequence diagram

is about to come out but the more deployed and available from vendors the 1.3.X [66] in which we will focus to summarize the extensions suffered by the protocol since its definition. OpenFlow 1.3 added new possibilities to the base definition and not limited by the need of being backward compatible with existing devices. The devices fully supporting this version of the protocol must be specifically addressed for OpenFlow. Among the new possibilities the metering capabilities is one of the more interesting allowing bandwidth enforcements related to flows and not ports.

Figure 2.5 shows the messages exchanged in an OpenFlow communication.

As a consequence of OpenFlow two type of OpenFlow hardware devices are defined:

OpenFlow-dedicated

These switches only run OpenFlow and receive all the control-plane decisions via the open

interface. They act as simple forwarding elements and (depending on the vendor) become hubs or blocks if no controller is attached.
In this category fit what is in the argot known as '*white boxes*' which refer to new vendors that taking profit of the open interface and the higher accessibility to silicon and lower prices offer as an alternative to the historically present vendors from the old days.

OpenFlow-enabled

In this category fits any device that has been provided with new software or firmware to expose an OpenFlow interface even if not designed by the vendor initially for that purpose or the ones that the vendor willingly offers OpenFlow as a feature but, on the contrary to the OpenFlow-dedicated switches, these devices still have independent switching capabilities that allow to run production legacy traffic at the same time that the OpenFlow service is provided.
The production traffic needs to be isolated from experimental. Two approaches are known to do so: Adding a special action to send the packet to the processing pipeline (standard switch pipeline with the legacy protocols) or create separate Virtual Local Area Network (VLAN) for the experimental traffic.
A new term has been coined as '*brite boxes*'. These boxes are not technically speaking OpenFlow-enabled boxes since they are not boxes that open their internals to OpenFlow as happened before but are designed to be opened and offer a small amount of the legacy functionality mainly for migration purposes and service deployment simplicity. The term brite-box was introduced by Karen Benson as a contraction of '*Branded Switching + White-Box Switching*' [67].

Proactive vs Reactive forwarding

One of the actions added to OpenFlow and inherited from Ethane among others is the ability to send the received packet to the controller. The packet is encapsulated in an OpenFlow message and is sent over the OpenFlow secure channel to the controller so that can be analyzed and can potentially produce changes on the network, meaning new flows to be deployed on that switch or any other switch in the network. This behavior is denominated reactive forwarding.
Reactive forwarding is clearly a key capability that will define future networking but it introduces some handicaps to be at least inspected when designing a OpenFlow solution. It introduces a negligible impact on the flow introduced by the Round Trip Time (RTT) between the controller and the switch. This is partially true since widely deployed SDN networks could have controllers far away from devices. Also, it might not be a good idea to apply this kind of forwarding to protocols without session, such as User Datagram Protocol (UDP) in which the controller would be asked for each packet coming into the network.

On the other hand proactive forwarding is off course allowed and even desirable. In that case, the controller would install flows on the devices that would provide with the rules to forward packets in the ordinary network operations and would leave the reactive approach to special or out of the normality cases.

2.3.0.4 Controllers

From an OpenFlow point of view, a controller is an entity that sends flows to the devices. But controllers, as we pointed out before, have become something more. Indeed their main purpose is to manage the control-plane and take any decisions based on the centralized network information obtained and the policies embedded by the network operators.
These policies are becoming more and more applications that sitting on top of the controller modify the network behavior not only based on standard protocols and network location but on higher level information and, in some cases as pointed out before, heuristic information. Among the functionalities covered by the controller, forensics is becoming one of the most demanded features and some vendors like Big Switch with their fabric use-case leverage on SDN to monitor the production network. Several are the unthinkable use-cases that have become affordable thanks to the appearance of the SDN paradigm and among them there is one clearly outlined by the related bibliography [63] [68] [69], the Information Centric Networking (ICN).

2.4 Future Internet - ICN

The Future Internet (FI) proposals are centered in providing an evolutive or disruptive path to evolve the underlying network architecture. Some of these architectures have been centered in separating identifiers from locators.
In order to resolve part of the limitations exposed by CDNs and thus break the ossification of networks in content distribution, the Information Centric Networking (ICN) [70] paradigm has proliferated to exploit the possibilities that intermediate network elements have to achieve efficient and effective content delivery. Within ICN we find some outstanding solutions, such as NDN/CCN [71], PURSUIT [72] and NetInf [73], which provide mechanisms to change the way content is represented in the network traffic and intermediate elements.
At the same time, the Information Centric Networking (ICN) paradigm employs intermediate network elements for content distribution while also separating identifiers from locators.

2.4.1 Separation of Identifiers and Locators

Instead of fixing the problems with complementary mechanisms, there is some consensus in the research community in that some problems related to end-to-end communications may be resolved by separating the location and identifier of a network node. This approach is followed by two outstanding architectures, Host Identity Protocol (HIP) [74] and Location/ID Separation Protocol (LISP) [75,76], as well as other derivatives and totally new proposals.

The HIP introduces cryptographic host identifiers forming a new global name space as a new intermediate layer between the IP and transport layers. It decouples the endpoint identifier and locator enabling the transport on host identifiers and routing on IP addressing that serve as pure locators. Although this seems a good solution, it presents many problems to be deployed because of the intrinsic meaning of identifiers and, in general, its weak solution to all requested capabilities for the Future Internet (FI).

On the other hand, the LISP is a routing-based solution using map-and-encap at border routers. The upstream IP address of border routers is used as Routing Locators (RLOCs) of hosts residing in the local domain to perform inter-domain routing, while intra-domain routing is performed using conventional IP addresses also referred to as Endpoint Identifiers (EIDs). These EIDs can potentially be associated with a group of RLOCs to support multi-homing. LISP tunnels data packets between source and destination RLOCs using the LISP database, which contains the RLOC/EID relation for each local domain to interpret packets. It also uses an on demand cache to select the destination RLOC for sending packets towards specific destinations. As the main drawbacks exposed by this architecture we highlight the need of maintaining a LISP database, because it raises scalability concerns, and that the creation of an on demand cache to route packets among different domains is not clearly defined. As a HIP derivative, the BLIND architecture [77] enhances HIP with security and identity protection, as well as location privacy by introducing new forwarding agents. Even though this is an interesting architecture from the security point of view, it inherits the same problems that HIP has.

Also being similar to HIP, the Routing Architecture for the Next Generation Internet (RANGI) [78] introduces a new layer called *Node Identity Internetworking Architecture (NodeID)*, which is used for transport related communications instead of IP addresses. RANGI adopts a hierarchical structure of host identifies employing special IPv6 addresses. The main benefit of RANGI is scalability but its main drawback is the complexity related with fundamental modifications into the IP addressing scheme and the Domain Name System (DNS) entry to store the required mappings.

The NodeID [79] proposal from the EU FP6 *Ambient Networks* project is also similar to HIP and LISP and introduces an architecture based on separation of locator, address, and administrative domains. Nevertheless, it is a network layer solution based on a locator/identifier split approach, and thus did not address session or identity management at all. Also, this architecture suffers from scalability problems related to inter-domain routing; a process based on routing tags, which contain domain information about the location of a concrete node identifier.

The Mobile Oriented Future Internet (MOFI) [80] proposes an architecture that considers, from the beginning, that network elements (hosts) can move across networks while also considering the existence of current IP networks as backbone for communicating separated edge networks. In contrast, it proposes new control and data planes for those edge networks to optimize mobility. The simplicity of this architecture and the organization of the functions provided in separated functional blocks (components) make it very interesting for adopting new functions into the Internet. However, it does not address the security or discovery problems.

The ILNP architecture [81] approaches the id/loc split by proposing a modification to the current DNS to use it to resolve names to identifiers and identifiers to addresses. It also proposes the incorporation of the identifiers as part of the IPv6 addresses used in network interactions. This proposal currently is being studied by the Internet Research Task Force (IRTF) [82]. Although this is a very lightweight and promising approach, it is host-centric and does not provide fine granularity or flexible naming. Also, the DNS infrastructure is not prepared to support the highly dynamic workloads required by the FI.

Finally, defined within the AKARI project, which was carried out by the National Institute of Information and Communication Technology of Japan (NICT), we find the Heterogeneity Inclusion and Mobility Adaptation through Locator ID Separation (HIMALIS) architecture [83]. It proposes a complete architecture that shares features with HIP and LISP but targets on a new identification scheme, different from IP and capable to support sensor networks and the IoT. The major benefits of HIMALIS reside in their simple targets of heterogeneity and mobility. It provides the necessary functional blocks to complement the current Internet architecture and allow it to evolve to a more functional network. However, there are still functions not covered by this architecture, specially those related to address the identities of network elements and manage communications according to them.

2.4.2 HIMALIS

The HIMALIS experimentation framework is formed by different modules that can be deployed both together in one machine or in separated machines. Depending on the installed components, a network element will play a role or another. Those elements are HNR, DNR, IDR, GW, Viewer, and Clients. Below we give a brief description of each component.
The Host Name Registry (HNR) is an element dedicated to keep a mapping table with the domain names of the registered hosts in the network and their identifiers and addresses. It is used by other network elements, such as clients, to determine the identifier and locator of a host.
The Domain Name Registry (DNR) is a global and hierarchical infrastructure that manages the assignment of domains with HNR elements. It is used by network elements to determine the HNR that responds to the resolution requests of the specified domain.
The ID Registry (IDR) is a global and overlay infrastructure used to communicate updates of the identities and locators to the gateways of the corresponding communication nodes. Thus, every node retains its reach-ability after moving from one network to another.
The gateway (GW) component provides the mechanism to connect access networks, which use their own locator namespace, to the global transit network (the Internet), which also uses its own locator namespace (IP addressing). This element contacts with the HNR to resolve the locator to which send the packets it receives, both from the global network or from the local network.
The Viewer component provides the possibility of knowing the current state of the current mapping tables of the gateways. It shows the host names, their identifiers and their locators at certain specific time, also showing how they are updated when a host moves from one network to another.
Finally, the clients are special software entities that use the HNR of their corresponding domain to resolve host names to their identifiers, establish new communications, and and configure the hosts with the necessary artifacts to permit them send and receive identifier-based packets.

2.4.3 Integrated Content Delivery

Besides the architecture approaches presented in previous subsections, the ICN approach has been a disruptive but well accepted and supported trend for future networks. It breaks with current node-centric networking by claiming that the majority of data communications

in the Internet are held to request or deliver content from a provider to a consumer, so they have an information-centric nature. Today, information is hosted by specific nodes and the only way for retrieving it is via establishing an end-to-end communication with them. Recent research efforts towards developing architectures for the FI have adopted a totally different point of view to address network operations by raising the role of information to the center of communications. In these approaches, the network does not connect nodes with links and processes via end-to-end connections, as is the case in the current Internet, but connects information consumers with the information they request, that is provided in a decoupled way by producers and distributors. This opens up new opportunities for accommodating, for example, sporadically connected nodes efficiently. However, it is not considering the identities of network nodes in any way.

The EU-funded projects 4WARD [84] and SAIL [85], its successor, defined and motivated the introduction of a new FI architectural paradigm called Network of Information (NetInf) [73] that extends the concept of identifier/locator split with another level of indirection and decouples self-certifiable objects from their storage location/s. NetInf distinguishes between Data Objects (DO), always encoded in a particular scheme (bit-pattern), and Information Objects (IO), i.e., information at a level above particular encodings. For example, much of the web content found today is semantically related, but this relationship is not represented in any way in the Internet. Similarly, relationships between copies of the same content are not represented. By employing bindings between IOs and DOs, information can be unambiguously replicated and located in the network. Moreover, since objects are self-certifiable, users can effectively employ access patterns that are reminiscent of any-casting and obtain the object that is *closer* (in network terms) to their access device.

Another pair of EU-funded projects, Publish-Subscribe Internet Routing Paradigm (PSIRP) [86] and Publish Subscribe Internet Technology (PURSUIT) [72], which is its successor, also approach ICN but from a totally different point of view. They propose a publish/subscribe scheme for the Internet, where information consumers *subscribe* to the information they want and information providers *publish* that information. In contrast with NetInf, this solution also approaches the information (data) delivery process and propose different identification schemes for information, subscribers, publishers, and intermediate nodes.

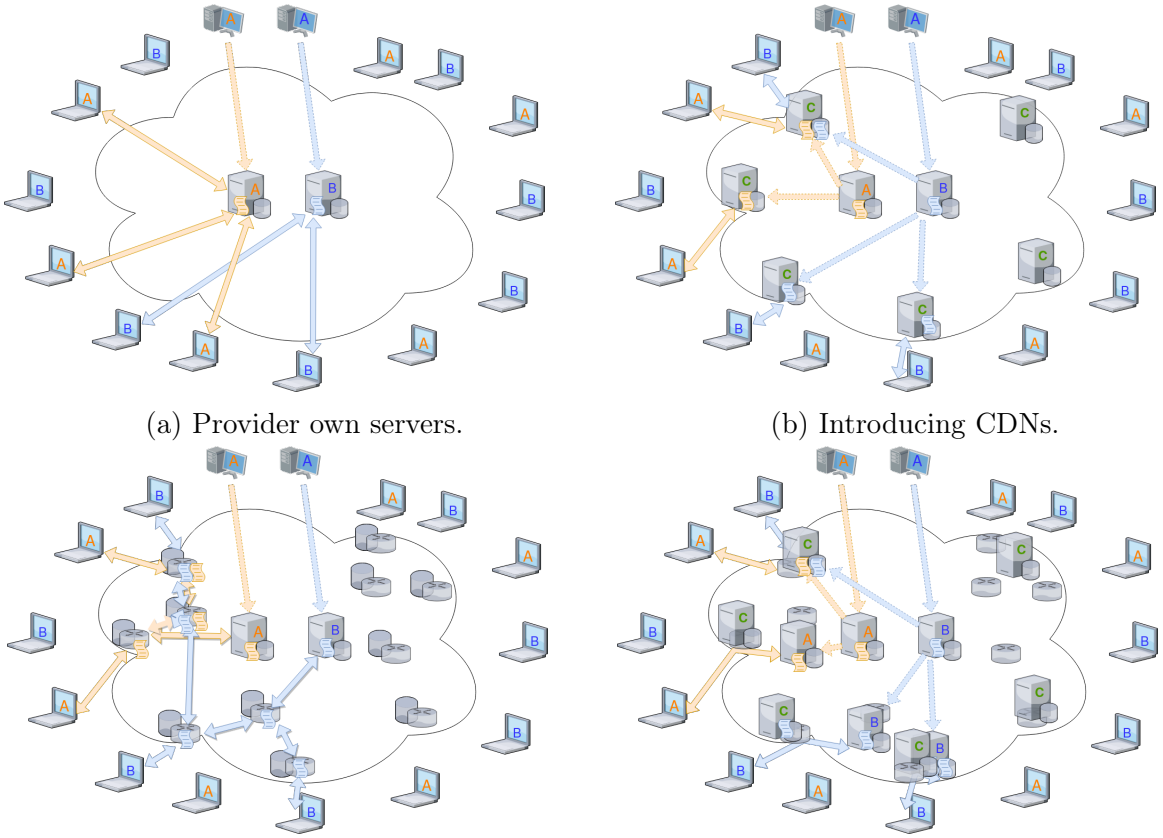
The Data-Oriented Network Architecture (DONA) architecture [87] is a communications architecture that replaces DNS names with self-certifying flat names and a name-based anycast primitive above the current Internet. Instead of certifying the content, DONA certifies the publishers and labels the data. Also, the data can not be dynamically

generated, it must be first registered in the trusted resolution handlers (RHs). Once the content requested by a client is found, it is delivered using IP routing. The security in DONA is achieved by content and provider validation.
The Content-Centric Networking (Content-Centric Networking (CCN)) approach [71] proposes an architecture similar to PSIRP/PURSUIT but enlarging intermediate content-centric elements to all intermediate elements in the network (i.e. switches and routers). In CCN, information consumers tell the network their *interest* for certain information item (content object) and then the network operates to provide it. Information producers do not need to react to consumers, they just drop the information to the network, as the network will deal wit it. This approach can be seen as a globally distributed cache, that fits perfectly in CDN scenarios. The problem with this architecture is that also leave aside the identity of communication parties, so it does not considers endpoint security or privacy. It is highly extensible, as shown in [88], and supports many traffic types.

2.5 Information Centric Network as a Service (ICNaaS)

CDNs are usually third parties offering their expertise and resources for caching or replicating content near the user. We introduced previously 2.1.4 the CDN [50] concept and stated that it is a three actors system. Now we contradict that sentence by introducing the fourth actor in the system, the ISP. ISPs have been seen as transport pipes but have been evolving recently to provide differentiated services and content as a consequence.
Content distribution services are traditionally implemented through a mix of techniques like HTTP redirection, DNS load distribution, anycast routing, and application-specific solutions, among others. As a result, a complex distributed system is in charge of redirecting users' requests to clusters of network caches. Such decision, i.e. the right cache(s) that must serve the content, is usually made based on the communicating endpoints (IP addresses) involved. Contrary to this, the *Information-Centric Networking* (ICN) paradigm advocates for delivering requested resources based on their name and independently from the data transport [82]. This can potentially increase the efficiency and scalability of content distribution, but it typically requires the deployment of state-of-the-art protocols like CCNx [71].
On the other hand, in the last few years we have witnessed the rise of *Software-Defined Networks* (SDN) and the high momentum they have gained [54]. By means of a logically centralized controller that maintains the global view of the network and exposes a programmatic interface, SDN offers huge opportunities for network programmability,

service automation, and simplified management.
Can we leverage SDN to provision CDNs as a Service (CDNaaS) which are led by ICN (ICNaaS) principles? What are the benefits of such approach? What challenges must be faced? Our claim is that provisioning CDN services with SDN is possible without a considerable lost of performance while increasing the dynamism of caching systems by adopting the ICN principles.
Figure 2.6 tries to show how the content provision has evolved. The first approach and the more abstract view of service provisioning is that of a central server providing the content for each content provider, let these be A and B in Figure 2.6a. To overcome the limitations of the former approach and overcome the geographical distribution of clients, surrogate servers are located nearer to the user thus increasing the QoS, this kind of approach is what is known as CDN and usually are provided by third parties, reflected as C in Figure 2.6b. To avoid all the problems related to IP ossification and avoid the implicit relation between network addresses and geographical location, the ICN initiative breaks with this philosophy by basing the routing on the content itself instead of the host where it is present. To that end, URL alike based routing is performed and usually accompanied by in-network caching as shown in Figure 2.6c. The ICNaaS proposal goes a step further providing with the means of having a CDN alike caching which can be potentially instantiated per content provider as a service in which the location of the content is performed likewise ICN but at the same time offering the possibility to integrate existing caching mechanisms such as the ones offered by CDNs operators nowadays. Despite being innovative, it is less disruptive and accommodates existing infrastructure as well as existing business models. A very simple overview of the ICNaaS is shown in Figure 2.6d.
Among the vast bibliography related to ICN two are the most related contributions to the work that is going to be introduced in this thesis. Both studies employ DASH as a mean for VoD.
Authors in [89] introduce the 'Cache as a Service' proposal and base their OpenCache (OpenCache) in SDN for VoD. In their proposal the control and decision of what content is to be cached is delegated to the ISP Network Operations Center (NOC) that has the knowledge and ability to optimize network utilization and possibly save its precious up-link bandwidth to other ISP. The authors also introduce the concept that given certain Service Level Agreements (SLAs), the solution could be also exposed to content providers such as CDNs. We actually agree with this view and even go further thinking that what should be offered to content providers is the instantiation of exclusive ICN/CDN instances that can be customized and controlled by the providers and implemented in the ISP premises.
Among the motivation of the authors, they state that '*Currently, BoD requests are handled*



(a) Provider own servers.

(b) Introducing CDNs.

(c) Introducing ICN.

(d) Introducing ICNaaS.

Figure 2.6: From base HTTP services to ICNaaS

individually leading to an independent low in the distribution network serving each user's request' [89], not taking profit of the repeatability of video objects transmission over the same link or network. Although nowadays CDNs already are able to cache video content, it is true that this is not always possible and I agree with their view.

The OpenCache proposal defines three elements to define the architecture, the OpenCache controller (OCC) orchestrating caching and distribution functionalities, the OpenCache nodes (OCNs) in charge of actually caching the content and deployed over the ISP network, and the VoD server which is not necessarily within the same ISP network.

The OCC accepts regular expression syntax to define the matching of the content in the requests although limited by the granularity level of OpenFlow. It also implements the caching behavior and manages the OCNs' resources. Finally it maintains the OpenFlow forwarding flows. The OCN 'simply' caches the content and maintains session status with OCC.

Authors in [89] also highlight the benefits of leveraging on SDN to implement caching

services thanks to the dynamism in packet routing and rewriting and the transparency to the customer obtained thanks to packet rewriting. Additionally network hardware monitoring.
Similarly, authors in [90] evaluate the VoD paradigm on FI architectures, in particular CCN (which is one of the more deployed, studied and well known ICN implementations) and leveraging on SDN. The motivation for their proposal and the previous study is the collision of interests between client based rate adaptation in DASH and the in network transparent content caching paradigm and how the later confuses the former.
In the study made by the authors, they stated that even in a stable environment the video representation selected by the DASH client was not stable obtaining an oscillating pattern. Although the experimentation was performed with wireless connectivity, the authors clarify that the Wireless Local Area Network (WLAN) coverage was perfect and no mobility was employed. They also conclude that DASH chunk requests are spread among the available rates unlikely repeating them on the retrieval of the same video any time in the future.
In their evaluation of CCN for DASH the authors introduce a proxy to translate the HTTP requests to CCN interests and employ a SDN controller for network traffic. The findings are that ICN has two negative impacts on DASH: reduced cache hit rate and imprecise rate estimation in the client due to the difference appreciated in channels to cache and to VoD server. In their solution proposal, the authors rewrite the MPD offered to the client extending it with the information related to the cache.

2.6 Scalable video in FI

The Scalable Video approaches, designed to confront the challenge of transmitting and storing high-definition video while reducing the needed bandwidth and storage size, lost its *raison d'être* with the adoption of end-to-end HTTP transmission due to the appearance of the high bandwidth last-mile connectivity, as well as, the decrease on the storage prices. Nevertheless, new trends in communications such as video conferencing systems based on Web Real-Time Communications (WebRTC) and ICN itself, have reactivated the interest on this kind of scalable video compression mechanisms [91] that allows in-network bitrate adaptation within a single stream.
Video coding has suffered of a certain parallelism with video transmission, while video designed solutions were left unused in exchange of HTTP streaming, scalable coding and recent video codecs have been put apart by the omnipresent H.264/AVC.
Despite this trend in industry, research in the field of scalable video coding continued

and new standards such as the Scalable High-Efficiency Video Coding (SHVC) [91] have appeared. Vidyo in partnership with Google has also declared the intention of providing scalability for the open alternative codec VP9 [92] and its importance for the WebRTC systems.
Applying scalable media coding to ICN [93] [94] [95] is a topic naturally emerged due ICN's in-network caching concept and the heterogeneity of devices to receive the video content. To reduce the access time, the amount of bandwidth required for each particular version of the media content and to avoid in-network transcoding and not 'reinvent the wheel' directly pointed out, again, to scalable coding.

2.7 Conclusions

This section has introduced the basic technologies and have pointed to the primary bibliography related to them needed to understand the proposals that follow in next chapters.
After a brief introduction to video coding and video streaming evolution, highlighting H.264/SVC introduced in Section 2.1.2.1 that represents the scalable video coding approach, the SCTP protocol and its extensions have been described in Section 2.2 serving as reference to Chapter 3 proposal. The ICN concept employed in chapters 4 and 5 was introduced in Section 2.4 which in addition to SDN introduced in Section 2.3 act as the basis to the proposal ICNaaS motivated in Section 2.5 that is finally defined in Chapter 4 and evaluated in Chapter 6.
In the following chapters, the SCTP protocol will be evaluated as transport layer for H.264/SVC taking profit of the parallelism of SCTP stream with H.264/SVC layer which to the best of my knowledge was the first approach to linking both concepts with the aim to obtain different levels of protection depending on the application level information. HTTP based video streaming is evaluated in the environment of FI and for the nowadays so relevant IoT paradigm. Finally, a nouveau solution to provide CDN alike as a service with ICN capabilities is designed and evaluated leveraging on SDN to transparently provide with HTTP video streaming, in addition, the characteristics of the application level metadata is employed to steer the traffic to the appropriate cache which is demonstrated with the dependency characteristics of H.264/SVC video.

Chapter 3

Evaluation of Scalable video delivery over SCTP

As a first approach to close the gap between the application layer, and in particular video streaming, and the network layer, a conservative approach has been followed in which Internet Protocol (IP) ossification is continued and enhancements are introduced in a non disruptive way. To that end, within standardized bodies two elements have been added without having the impact expected, joining forces might produce the necessary outcome to boost their adoption.
On the Transport layer the Stream Control Transmission Protocol (SCTP) is the alternative to the well known alternatives Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). From the Application layer perspective, the contender is the Scalable Video Coding (H.264/SVC). Both elements have in common the subdivision into parts, while SCTP splits its association into streams, H.264/SVC separates the video stream into dependable layers. Although SCTP has been previously employed to transport video [96] [97], this is the first time to the best of our knowledge that the multi-streaming capability of SCTP is employed with a scalable codec.
In this chapter SCTP is evaluated as transport of H.264/SVC video on top of Real-time Transport Protocol (RTP).

3.1 Description

In parallel with the enhancement of computer capabilities in terms of processing power and network bandwidth has been a growth in the use of multimedia services and more recently video transmission is becoming the killer application, in the format of Video on

Demand (VoD), video conferencing systems and/or live streaming. Historically the two major transport layer protocols available TCP and UDP have been used for multimedia streaming, being the later favored for its maximum bandwidth throughput while the former has been the choice when resiliency was requested. Nowadays video is delivered to a highly heterogeneous device variety which usually implies wireless connectivity hence implying mobility in most of the cases. The two predominant network protocols are lacking capabilities in terms of resiliency or performance.

Although the way to transmit the video is a very important matter and is actually the term on which this document focus, there is a more important factor, the video itself. While it is still common to find ancient (in computer evolution scale) codecs such as MPEG-2 while the standard is using Advanced Video Coding (H.264/AVC). Several video codecs are available in terms of efficiency or even licensing but most of them lack the capability to be adapted efficiently during the transmission stage. The usual way to deal with such restrictions, the network might not be able to transmit the whole stream in time or the device might be unable to play the stream in the hardware available, is to either transcode (term employed for decode and reencode at once) the video stream to the new restrictions or pre-encode several versions of the stream at origin with the unavoidable computing overload. Therefore during the transmission stage the streaming the communication edges monitor the available bandwidth and configure the transcoding stage or switch over the different available representations of the media to one that suits the parameters. Both approaches have their drawbacks and benefits but in general transcoding is a Central Processing Unit (CPU) consuming process while encoding multiple versions is storage consuming.

Scalability has been present in video coding since early stages but it hasn't been until the definition of H.264/SVC [34] [35] that a real and industry standards capable codec appeared. The H.264/SVC is the standardized scalable extension for the also standardized and well-known H.264/AVC [36]. The major capability of H.264/SVC is to produce a video bit stream that produces different outputs in the temporal, quality or frame size domains. The three parameters can vary and each 3-tuple or triple is an operation point which in turn requires a certain bit-rate. Thanks to the Network Abstraction Layer Unit (NALU) structure (introduced in section 2.1.2.1), each H.264/SVC network data chunk is identified with the triple to which it belongs. Therefore, steering traffic to a certain bit-rate is just a matter of dropping the NALUs belonging to any non-dependable layer of the selected operation point. Every layer may depend from 'lower level' layers, meaning those with lower values in the triple defining the operation point. One of the key features of H.264/SVC is the backward compatibility of the *base layer* with the almost omnipresent H.264/AVC

which ensures a minimum service for almost any device in the world and usually with hardware decoding support.
With regards to transmission, both TCP and UDP are considered not well suited for the transmission of multimedia information when network conditions are poor. As an alternative to the lack of multimedia aware network protocol, the RTP [18] protocol is an application layer protocol that is usually encapsulated on top of UDP taking the flow, congestion control, retransmission events and redundancy to the application layer when needed. This approach has been extensively used and is considered as a proper method to transmit video (and audio as far as is concerned) over computer networks.
The SCTP protocol is a general purpose standardized transport layer protocol that was in origin designed for telephony signaling. It is able to provide a reliable full-duplex transmission with flow and congestion control, multi-streaming and multi-homing. The protocol, which has been already introduced in section 2.2, introduces the concept of stream in an association, meaning that the connection between two sctp peers for a certain pair of ports can transport different flows which in turn have their independent flow control reducing the head-of-line blocking TCP problem. SCTP is reliable using similar mechanisms to those of TCP but is inherently message oriented similarly to UDP. The streams are unidirectional and are delivered in order to the corresponding endpoint. SCTP allows each endpoint to have multiple network addresses and dynamically change them which was mentioned previously as multi-homing. Originally SCTP uses one of the addresses as the primary and the other addresses as fail-safe alternatives, nevertheless the Concurrent MultiPath Transfer for Stream Control Transmission Protocol (CMT-SCTP) extension has been defined [5] to support parallel usage of the multiple interfaces therefore balancing the traffic.
SCTP and H.264/SVC can naturally be combined to map the former streams to the later layers, taking advantage of the multi-streaming and also from the multi-homing mechanisms provided by SCTP. Both capabilities are of relevance for mobile devices which usually have restricted video hardware having usually multiple network interfaces on the other hand, like cellular and wireless. The proposal examines different strategies to transmit H.264/SVC with SCTP demonstrating how TCP and UDP get outperformed by the approach. The study was carried out with the ns-2 [98] simulator by employing an SCTP implementation provided by University of Delaware [99].
The remainder of this chapter is structured as follows. Section 3.2 the simulation environment is introduced and the multiple combinations of SCTP and H.264/SVC are introduced. Next results are show and analyzed in section 3.3. Finally, conclusions are presented in last section.

3.2 Evaluation Scenarios

In this section we aim to describe the simulation environment used to demonstrate the advantages of using SCTP as the H.264/SVC video transport protocol, in comparison with other protocols (RTP and TCP) currently used. The main strategy to show the out performance of SCTP is based on sending H.264/SVC layers as SCTP streams over a SCTP association, while also using the multi-homing features of the protocol. The scenario in which the comparison is established employs strongly lossy networks. The simulations are carried out using ns-2 (v. 2.34). It includes the implementation of the SCTP transport protocol with Delaware's University extensions.
In order to implement our source agent, we use trace files generated from real mp4 files created with a modified version of *mp4creator* developed in the SCALNET project [100] [7]. This tool permits to employ the mp4 file format as the container of H.264/SVC video, according to the ISO standard [101]. We use mp4 files that contain, in addition to the actual bitstream, some extractor and hint tracks that ease the streaming process done by the server. The generated traces contain not only the size of each RTP packet but also the packet number within the frame, the frame number, the sequence number and a stream identifier that is used within SCTP. We built two types of video source, one taking only one stream as input and the other taking multiple streams. We have also developed a video sink that generates an output trace of the received video events. This trace permits us to generate, using the original mp4 file, SVC files in which all the lost NALUs are removed. Instead of generating one H.264/SVC file to represent the received video for the whole transmission, we produce one H.264/SVC file per frame. With this approach we pretend to have a fine grain control over the decoding process to know whether a particular frame has been successfully decoded, or there is not enough information to decode it at all. In case a particular frame is not decoded, we replicate last decoded frame. With this we can calculate the PSNR value of each frame with the certitude that it is aligned with the original video sequence. Obviously this decision lead us to encode our test stream with I frames exclusively. We employ a unique video sequence for a complete set of transmission scenarios. The sequence is the union of Parkrun, Shields, Stockholm, and Mobcalm subsequences. We have encoded them using the Joint Scalable Video Model (JSVM) reference software, version 9.19. There are two layers, the base layer and one additional quality enhancement layer, with average bit rates of 6.8 and 23.5 Mbps, respectively. The video is VBR encoded, causing some severe variations in the transmission rate that globally rise over 45 Mbps at some segments of the video. The duration of the encoded stream is 2,116 frames streamed at 25 frames per second, permitting a test of 88.64 s. The generated

SVC file has an average PSNR-Y of 34.46 dB. The sequence is encapsulated in an mp4 file and hinted with an MTU limit of 1,460 bytes. Transmission using TCP is treated as a special case because it is not a message-oriented protocol. The TCP agent collocated with the video sink agent produces reception events indicating amounts of received bytes which do not respect NALU blocks. In this case, the agent waits until all the bytes for a NALU have been received before including it in the reception trace.
The simulated scenario is represented in figure 3.1. It is really simple, with two nodes connected by two duplex links. These nodes are actually modeled with three nodes each (a core node and two nodes for the network interfaces), as proposed by CMT-SCTP module authors [102]. Each core node employs a transport agent. In order to establish a comparison, we use SCTP, CMT-SCTP, TCP/FullTCP and RTP transport agents. At the emitter end, the transport agent is attached to a video source application that reads information from the pre-generated video trace. At the receiver end, the transport agent connects to a video sink that generates the reception trace. We left most of SCTP parameters in ns-2 with their default values. Nevertheless, there are some exceptions that have been modified. The *pathMaxRetrans_* and *changePrimaryThresh_* govern the decision of which interface to use in case a retransmission has to be done. We set both parameters to 1, implying that after a packet loss the SCTP agent will resend it on the secondary interface. Additionally, we set the *Reliability_* parameter to 0 to test the use the SCTP protocol in unreliable mode, which is basically a best effort approach similar to UDP. The results of each simulation include following information, averaged per second:

- *diff. time*, that is, the delay experimented by video packets in the transit from source to sink.
- *bandwidth* used in each interface.
- *discarded packets* detected in each interface, due to router congestion or link failure.
- *loss bandwidth* detected at the video sink, after packet retransmissions.
- *PSNR*, objective measure of the received video quality with respect to the original file.

Simulations use three different link bandwidths: 30, 45, and 60 Mbps. Both links have the same bandwidth. Each of these bandwidth configuration is tested with four different error distributions: no loss, uniform loses with 10^{-4} probability, with 10^{-3} probability and with 10^{-2} probability. Losses start at second 20 in the first link and at second 40 in the second.

Both links are configured with a constant delay of 10 ms. For each combination of network bandwidth and error distribution, we have tested eleven different transmission strategies. Due to the amount of combinations, we describe each one with a label that is used in the rest of the paper:

- *CMTMultiReliable*: CMT-SCTP transmission, with base and enhancement layers using different streams, both with reliable transmission.
- *CMTMultiUnreliable*: similar to the previous one, but using unreliable transmission for both layers.
- *CMTMultiMixed*: similar to the previous one, but using reliable transmission for the base layer, and unreliable for the enhancement layer.
- *SCTPMultiReliable*: Baseline SCTP transmission, with base and enhancement layers using different streams, both with reliable transmission.
- *SCTPMultiUnreliable*: similar to the previous one, but using unreliable transmission for both layers.
- *SCTPMultiMixed*: similar to the previous one, but using reliable transmission for the base layer, and unreliable for the enhancement layer.
- *SCTPReliable*: Baseline SCTP transmission, with base and enhancement layers using the same stream with reliable transmission.
- *SCTPUnreliable*: similar to the previous one, but using unreliable transmission.
- *RTPMulti*: RTP transmission, with base and enhancement layers being sent through the first and second interfaces respectively.
- *RTP*: similar to the previous one, but using only the first interface.
- *TCP*: TCP transmission with both layers sent through the first interface.

Table 3.1 summarizes the characteristics of these eleven strategies.

Strategy	*MultiStreaming*	*Reliable*	*Unreliable*	*Mixed*
CMTMultiReliable	√	√	◯	◯
CMTMultiUnreliable	√	◯	√	◯
CMTMultiMixed	√	◯	◯	√
SCTPMultiMixed	√	◯	◯	√
SCTPMultiReliable	√	√	◯	◯
SCTPMultiUnreliable	√	◯	√	◯
SCTPReliable	◯	√	◯	◯
SCTPUnreliable	◯	◯	√	◯
RTPMulti	√	◯	√	◯
RTP	◯	◯	√	◯
TCP	◯	√	◯	◯

Table 3.1: Strategies overview

3.3 Evaluation Results

This section shows the results of the simulations and describes the obtained values. There is an explosion in the combinations of the eleven transmission strategies, four error configurations and three bandwidth configurations. As a result, only the average values of all the combinations are summarized in tables 3.2 to 3.5. On the other hand, figures 3.2 to 3.12 present more detailed information for the 30 Mbps bandwidth and 10^{-2} uniform loss rate combination. This scenario represents the worst case of all those studied, with a strong packet loss rate and also congestion due to peaks in the video bandwidth.

Figures 3.2 to 3.12 represent a mosaic of three or four graphs for each transmission strategy. The top left plot shows data about the eth0 interface. Similarly, the bottom left plot presents the same information for the eth1 interface (some strategies use only the first one). Both plots present the bandwidth usage on the scale of the left axis of the plot, while the amount of discarded packets is represented with reference to the right axis. The use of two axis tries to ease the practical representation of data that would be impossible in some cases with just one axis. This dual vertical axis is also used in bottom right plot, in which the average delay of packets is shown using the left axis, whereas the loss bandwidth calculated at the video sink is shown using the right axis. Finally, the top right plot represents the PSNR of the received video.

As shown in figures 3.2, 3.3 and 3.4, CMT transmissions use both interfaces fully. All the other strategies present a clear unbalance in the use of both interfaces. Most of them employ preeminently the first interface, with the exception of RTPMulti, shown in figure 3.10, where the second interface is used to send the enhancement layer, that has a bigger

bandwidth consumption than the base layer. Some SCTP multi-interface transmission strategies (figures 3.4 and 3.5) make a sparse use of the second interface, reduced to the retransmission of lost packets belonging to the reliable stream.
Taking a look at figure 3.11 corresponding to the RTP scenario, it is possible to see how congestion affects strongly the quality of the transmission when the sending rate exceeds the available bandwidth. Each time the network usage of the first link exceeds 30 Mbps, the loss bandwidth at the reception rises clearly. This is also true in all the CMT-SCTP and SCTP cases where unreliable streams are used (figures 3.3, 3.4, 3.6, 3.7 and 3.9).
We present the information for all the scenarios in four tables, one for each error distribution value. These tables contain values that measure the average delay and Peak Signal Noise Ratio (PSNR) for each transmission strategy.

3.3.1 TCP

Looking at the tables 3.2 to 3.5, it is observed that TCP performs pretty good on error free environments, but as the number of errors rises up the average delay grows prohibitively.

3.3.2 RTP

As expected, RTP has the worst PSNR value but the best average delay value. We have to take into account that packet losses are not retransmitted, so there is no penalization to the delay but the negative influence in the quality of the received video is excessive.

3.3.3 Reliable baseline SCTP

The SCTP reliable approach outperforms TCP on lossy scenarios and scenarios where bandwidth is sufficient. As can be seen in Figures 3.6, 3.7, and 3.9, the receiver video agent stops receiving data from the SCTP agent approximately at the middle of the simulation. When the receiver window is full, the SCTP agent drops all incoming data chunks, causing the starvation of the video agent. The pathmaxretrans_ parameter set to 1 produces, when the available bandwidth is high and there is a big error rate, an increment in disordered packets, for which baseline SCTP is not well prepared. This is precisely the motivation of the three algorithms SFR, CUC and DAC, employed by the CMT-SCTP extension. Looking at the ns-2 traces it is possible to observe that the cumulative TSN in SACKs gets frozen from the indicated time instant, provoking a stall in the transmission.
Sending each layer in a different stream (SCTPMultiReliable) enhances the transmission performance on lossy scenarios in comparison with the use of a unique stream

(SCTPReliable), as can be seen in tables 3.2 to 3.5.

3.3.4 Unreliable baseline SCTP

In our opinion, the unreliable approach is not worth when retransmissions are deactivated. The results show that, with an error probability 10^{-4} and 10^{-3}, there is nearly no difference in the value of the average delay (at best it is equal) in comparison with the reliable approach, but, the big difference comes in relation with the PSNR value. The unreliable approach provokes a strong reduction in the quality of the video, traduced in black square artifacts due to complete video slices lost, which result in pictures with artifacts. With error probability 10^{-2} the unreliable approach outperforms the average delay of the reliable approach, but still having a very high cost in video quality. In this case, sending layers in different streams seems to be a good idea although the PSNR is low.

3.3.5 Mixed reliability with baseline SCTP

To get a compromise between error resilience and delay, we test scenarios in which the base layer is protected using a reliable transmission while the enhancement layer is sent with an unreliable approach. The results exposed in the SCTPMultiMixed columns in tables 3.2 to 3.5 show that this mixed approach globally reduces the average time with respect to the only-reliable approach, while obtaining a greater PSNR than the only-unreliable one. The enhancement of this approach could not only be measured in terms of the delay and objective quality, but also in the Quality of Experience (QoE) results for the user, as only packets of the enhanced layer will be affected by losses, reducing the overall impact on the quality of the image.

3.3.6 CMT-SCTP

We studied transmission mechanisms with CMT-SCTP and the three alternatives in relation with the reliability of the streams (reliable streams, unreliable streams and a mixed combination of both) and our conclusion is that, thanks to the good performance of using reliable streams in CMT-SCTP, CMTMultiReliable is always the best choice of all, because the delay is kept low and the PSNR is always the maximum.

Strategy	30Mb		45Mb		60Mb	
	avg diff time	*PSNR*	*avg diff time*	*PSNR*	*avg diff time*	*PSNR*
CMTMultiReliable	0.0057	34,4653	0.0056	34,4653	0.0055	34,4653
CMTMultiUnreliable	0.0057	34,4653	0.0056	34,4653	0.0055	34,4653
CMTMultiMixed	0.0057	34,4653	0.0056	34,4653	0.0055	34,4653
SCTPMultiReliable	1.6108	34,4653	0.0172	34,4653	0.0061	34,4653
SCTPMultiUnreliable	1.6108	34,4653	0.0172	34,4653	0.0061	34,4653
SCTPMultiMixed	1.6108	34,4653	0.0172	34,4653	0.0061	34,4653
SCTPReliable	1.6105	34,4653	0.0171	34,4653	0.0060	34,4653
SCTPUnreliable	1.6105	34,4653	0.0171	34,4653	0.0060	34,4653
RTPMulti	0.0061	33,4609	0.0052	34,4653	0.0052	34,4653
RTP	0.0094	29,8153	0.0059	33,6131	0.0052	34,4653
TCP	2.4409	34.4653	0.2865	34.4653	0.0583	34.4653

Table 3.2: No error

Strategy	30Mb		45Mb		60Mb	
	avg diff time	*PSNR*	*avg diff time*	*PSNR*	*avg diff time*	*PSNR*
CMTMultiReliable	0.0057	34,4653	0.0056	34,4653	0.0055	34,4653
CMTMultiUnreliable	0.0057	34,4653	0.0056	34,4653	0.0055	34,4653
CMTMultiMixed	0.0057	34,4653	0.0056	34,4653	0.0055	34,4653
SCTPMultiReliable	1.6146	34,4653	0.0174	34,4653	0.0062	34,4653
SCTPMultiUnreliable	1.6146	34,4653	0.0174	34,4653	0.0062	34,4653
SCTPMultiMixed	1.6146	34,4653	0.0174	34,4653	0.0062	34,4653
SCTPReliable	1.6149	34,4653	0.0172	34,4653	0.0061	34,4653
SCTPUnreliable	1.6149	34,4653	0.0172	34,4653	0.0061	34,4653
RTPMulti	0.0061	33,4229	0.0052	34,4272	0.0052	34,4272
RTP	0.0094	29,7677	0.0059	33,5600	0.0052	34,4159
TCP	2.4581	34,4653	0.2865	34,4653	0.0583	34,4653

Table 3.3: Uniform 10^{-4}

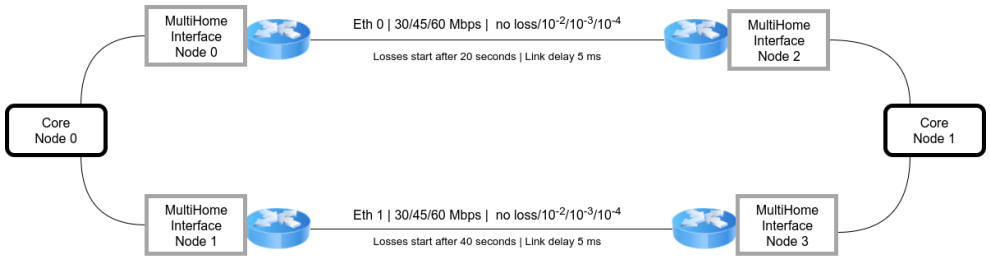


Figure 3.1: Ns-2 simulation scenario.

Strategy	30Mb		45Mb		60Mb	
	avg diff time	*PSNR*	*avg diff time*	*PSNR*	*avg diff time*	*PSNR*
CMTMultiReliable	0.0244	34,4653	0.0101	34,4653	0.0057	34,4653
CMTMultiUnreliable	0.0300	34,4596	0.0101	34,4653	0.0057	34,4653
CMTMultiMixed	0.0300	34,4596	0.0101	34,4653	0.0057	34,4653
SCTPMultiReliable	1.9877	34,4653	0.0458	34,4653	0.0556	34,4653
SCTPMultiUnreliable	1.9877	34,4653	0.0458	34,4653	0.0556	34,4653
SCTPMultiMixed	1.9877	34,4653	0.0458	34,4653	0.0556	34,4653
SCTPReliable	3.0093	34,4653	0.0466	34,4653	0.1124	34,4653
SCTPUnreliable	2.0130	34,4653	0.0466	34,4653	0.1124	34,4653
RTPMulti	0.0061	32,9780	0.0052	33,9753	0.0052	33,9753
RTP	0.0094	29,2786	0.0059	33,0309	0.0052	33,8717
TCP	2.9667	34,4653	0.4165	34,4653	0.1325	34,4653

Table 3.4: Uniform 10^{-3}

Strategy	30Mb		45Mb		60Mb	
	avg diff time	*PSNR*	*avg diff time*	*PSNR*	*avg diff time*	*PSNR*
CMTMultiReliable	1.3272	34,4653	0.0644	34,4653	0.6181	34,4653
CMTMultiUnreliable	0.4039	34,4372	0.7470	34,4417	0.4098	34,4367
CMTMultiMixed	1.7904	34,2869	0.7470	34,4417	0.4026	34,4427
SCTPMultiReliable	41.1406	34,4653	20.3130	34,4653	3.5188	34,4653
SCTPMultiUnreliable	17.2343	33,9208	12.4975	34,1825	13.0626	34,1603
SCTPMultiMixed	17.2308	33,9577	12.4961	34,1896	13.0586	34,1666
SCTPReliable	44.7756	34,4653	19.2490	34,4653	9.7437	34,4653
SCTPUnreliable	18.2282	33,9311	12.1949	34,1579	11.1983	34,0951
RTPMulti	0.0061	29,5685	0.0052	30,5368	0.0052	30,5368
RTP	0.0094	24,7738	0.0059	28,5860	0.0052	29,3284
TCP	37.1808	34,4653	25.5925	34,4653	23.5196	34,4653

Table 3.5: Uniform 10^{-2}

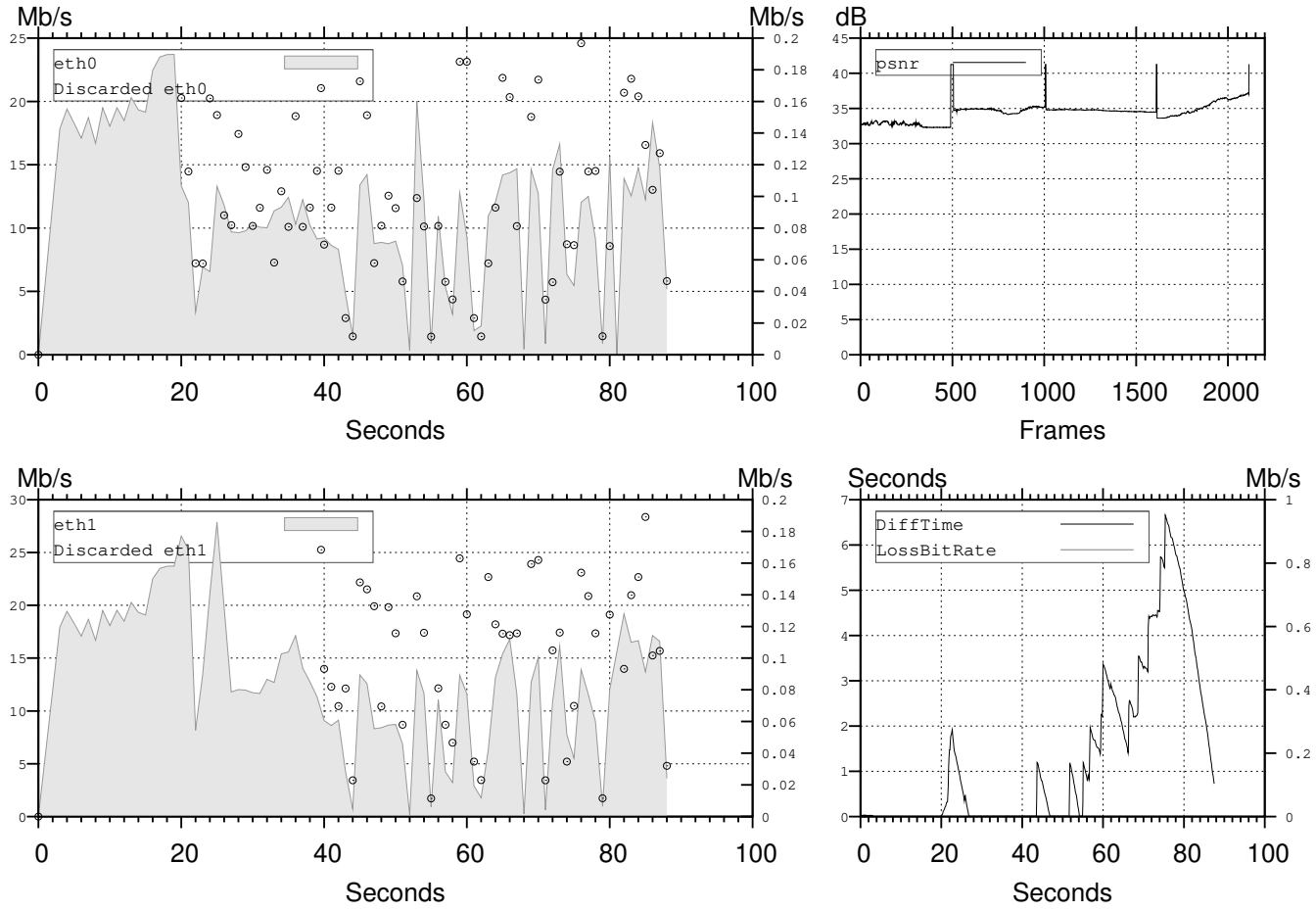


Figure 3.2: Uniform 10^{-2} 30 Mbps CMTMultiReliable.

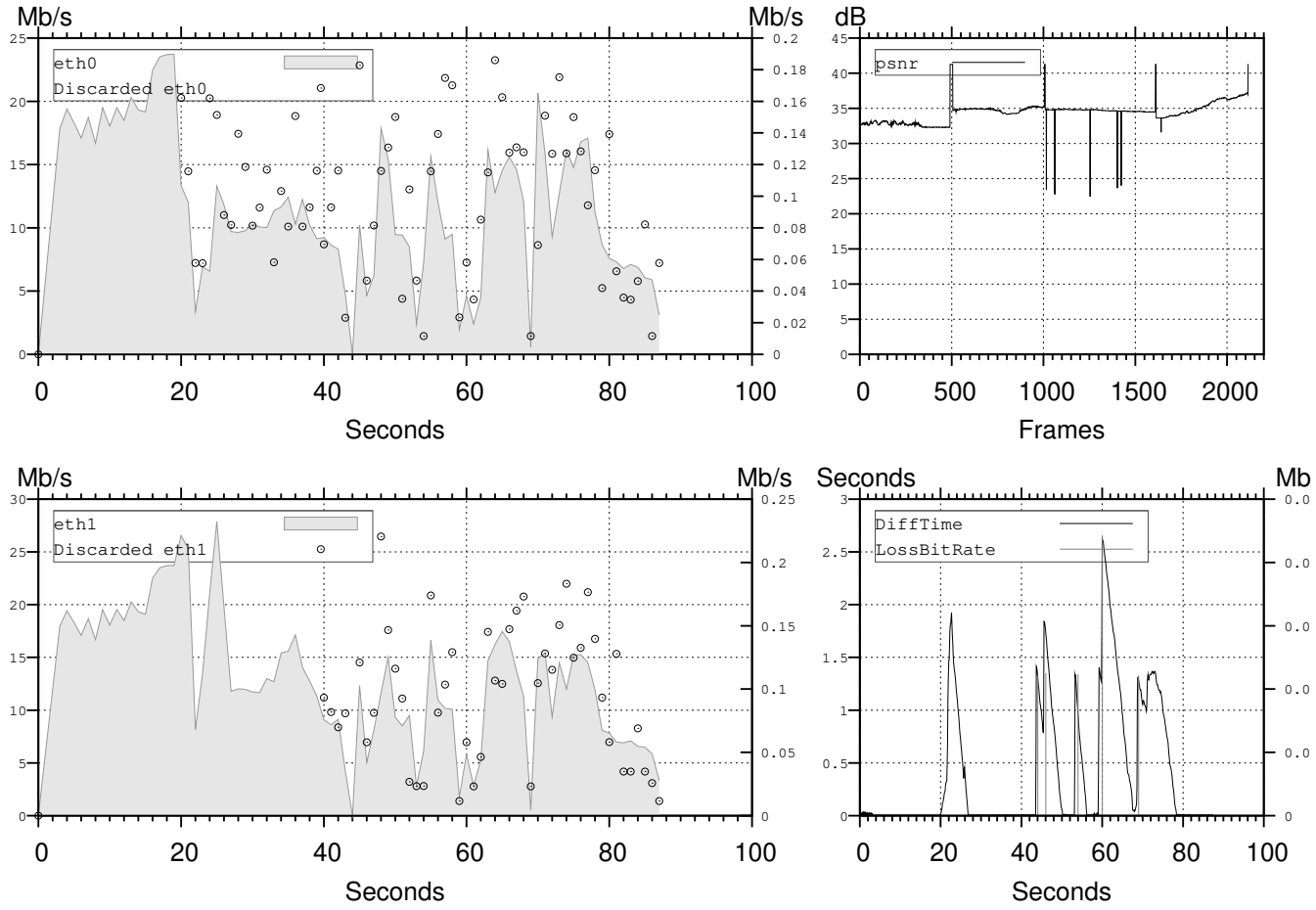


Figure 3.3: Uniform 10^{-2} 30 Mbps CMTMultiUnreliable.

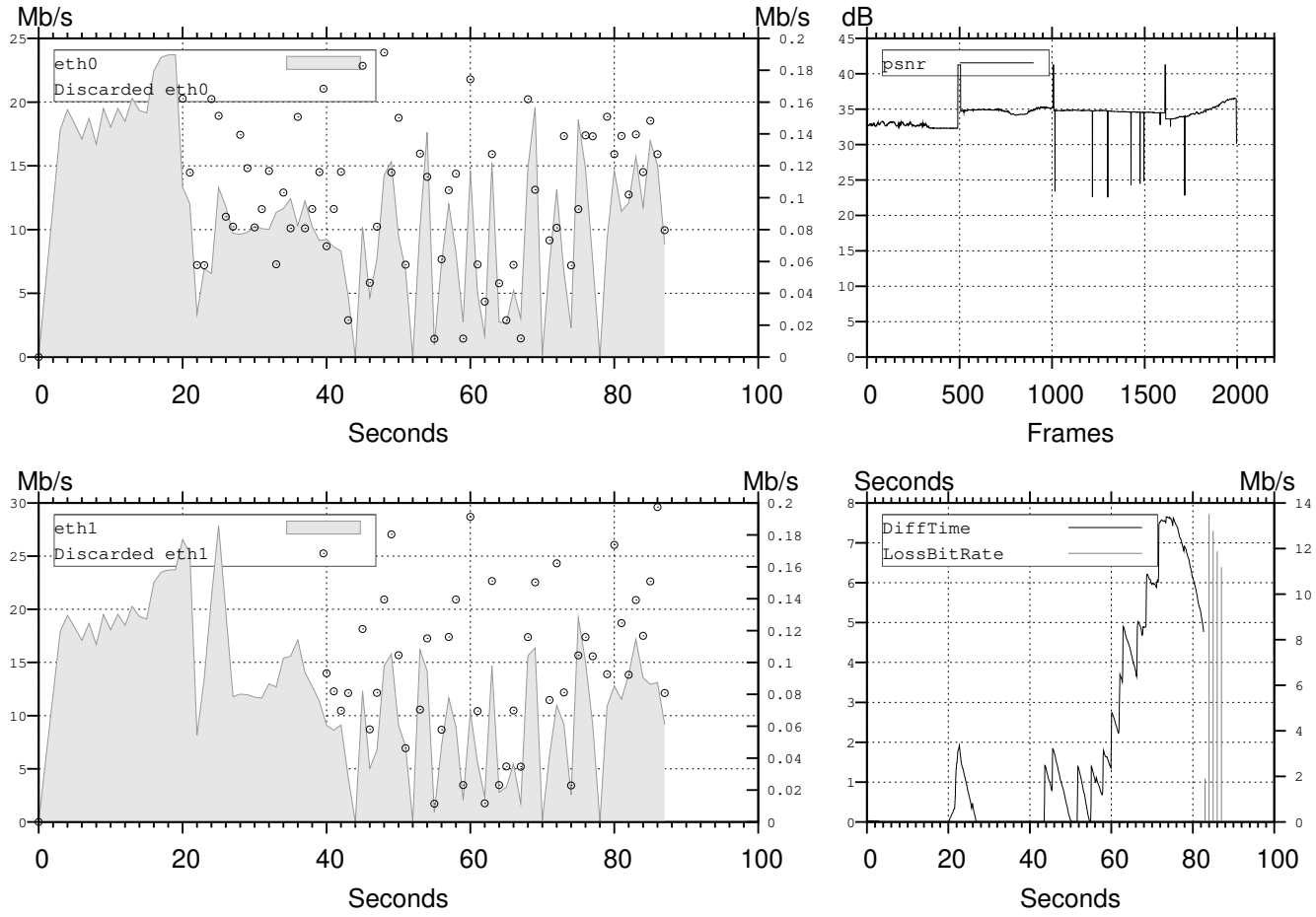


Figure 3.4: Uniform 10^{-2} 30 Mbps CMTMultiMixed.

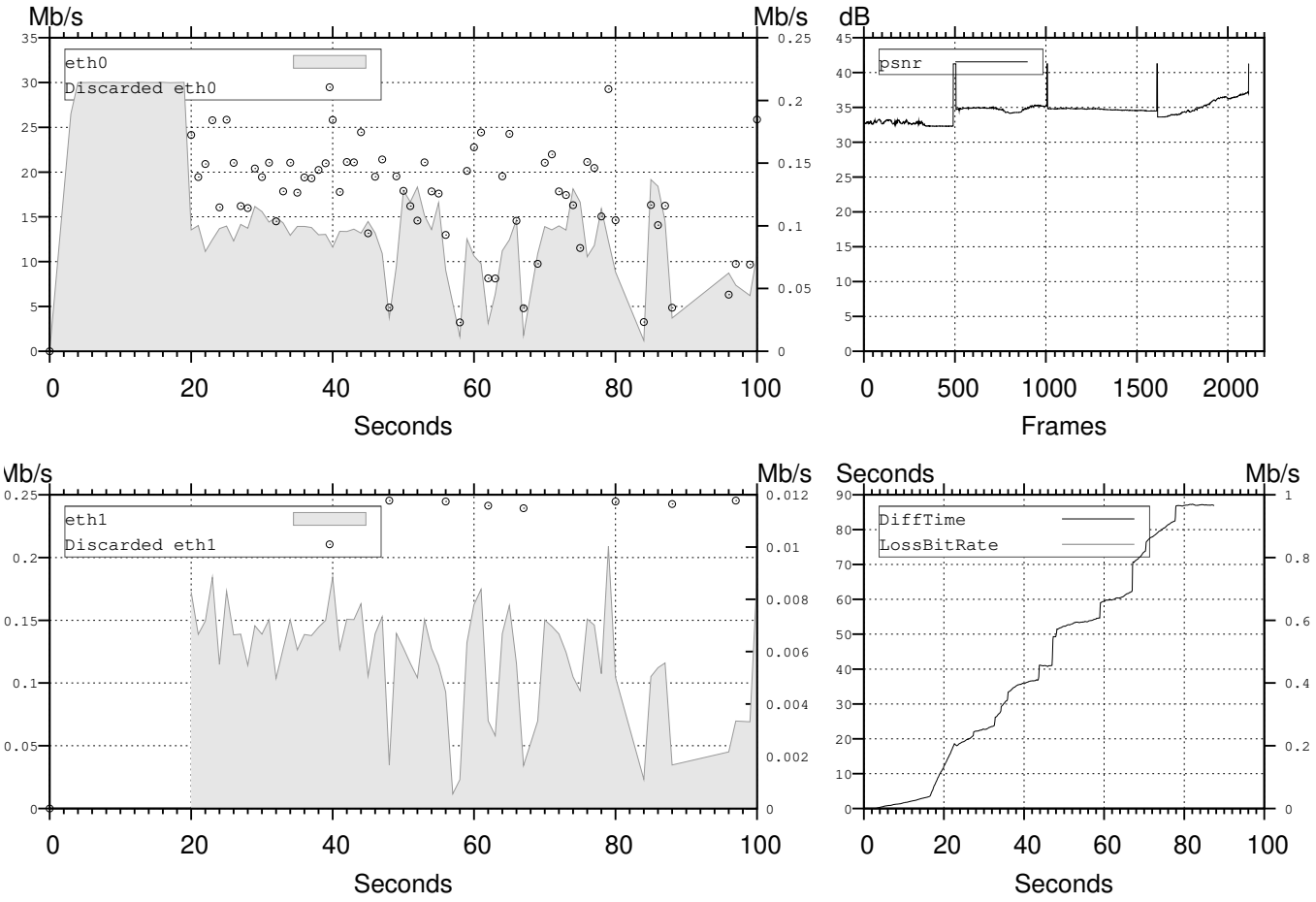


Figure 3.5: Uniform 10^{-2} 30 Mbps SCTPMultiReliable.

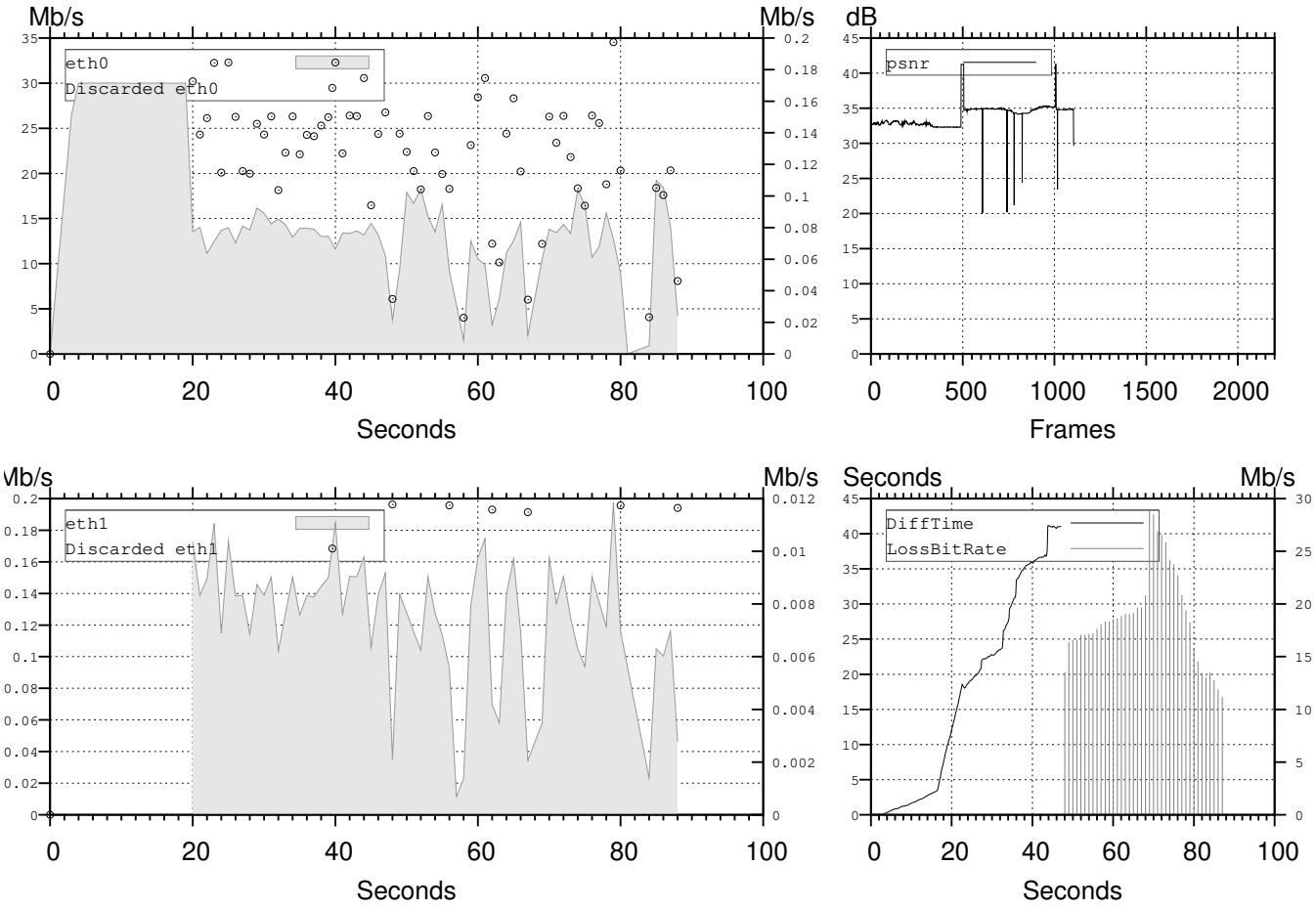


Figure 3.6: Uniform 10^{-2} 30 Mbps SCTPMultiUnreliable.

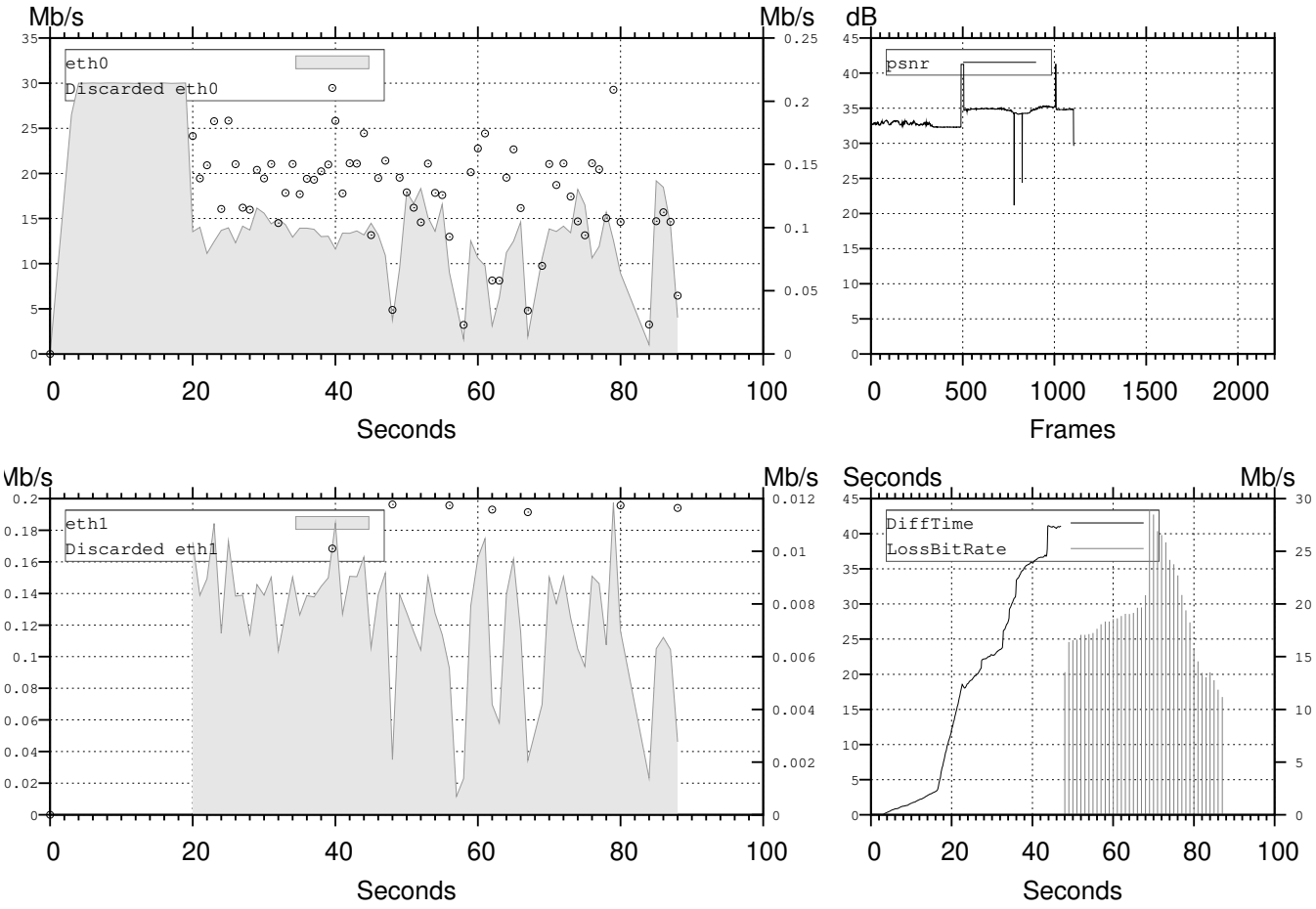


Figure 3.7: Uniform 10^{-2} 30 Mbps SCTPMultiMixed.

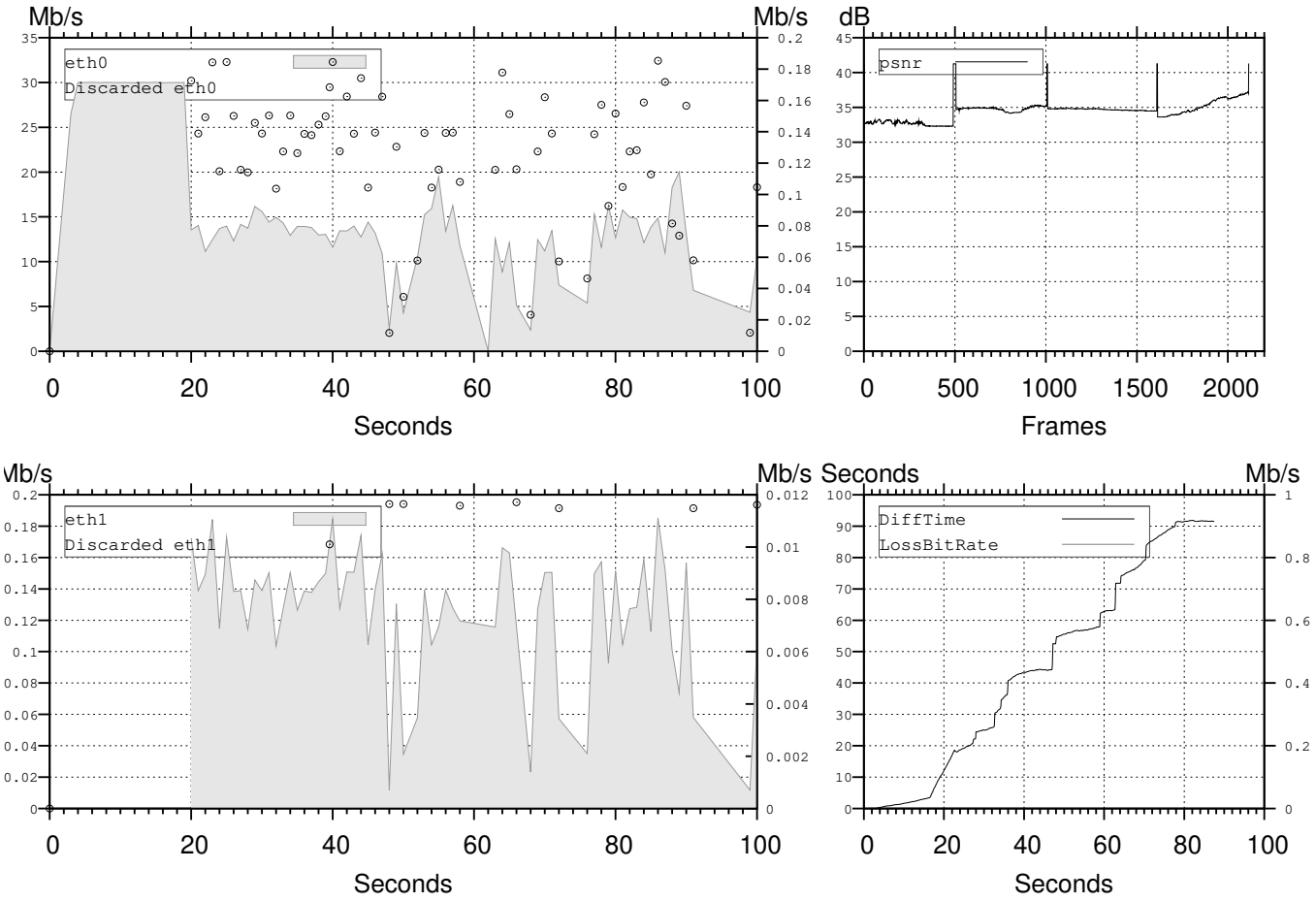


Figure 3.8: Uniform 10^{-2} 30 Mbps SCTPReliable.

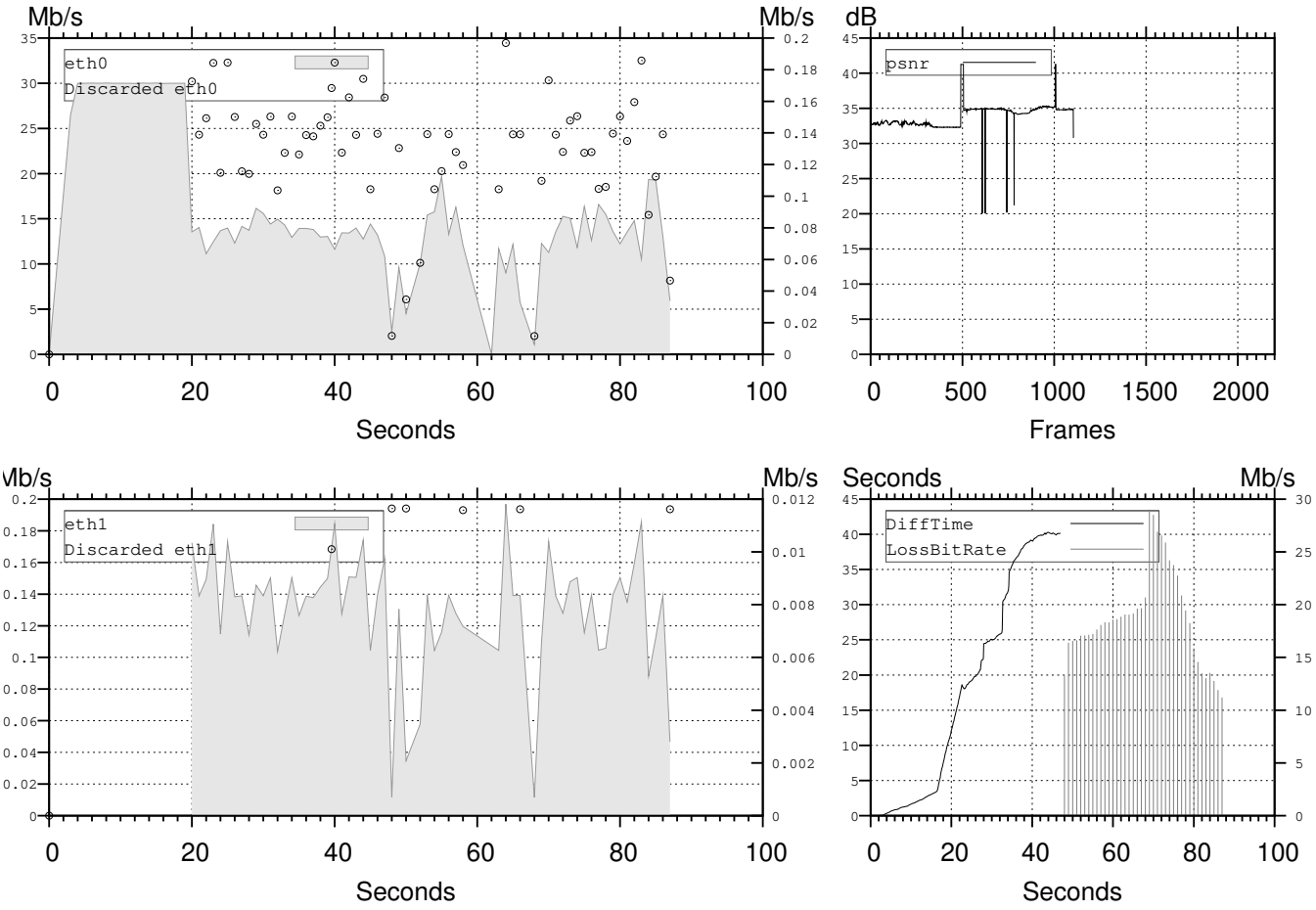


Figure 3.9: Uniform 10^{-2} 30 Mbps SCTPUnreliable.

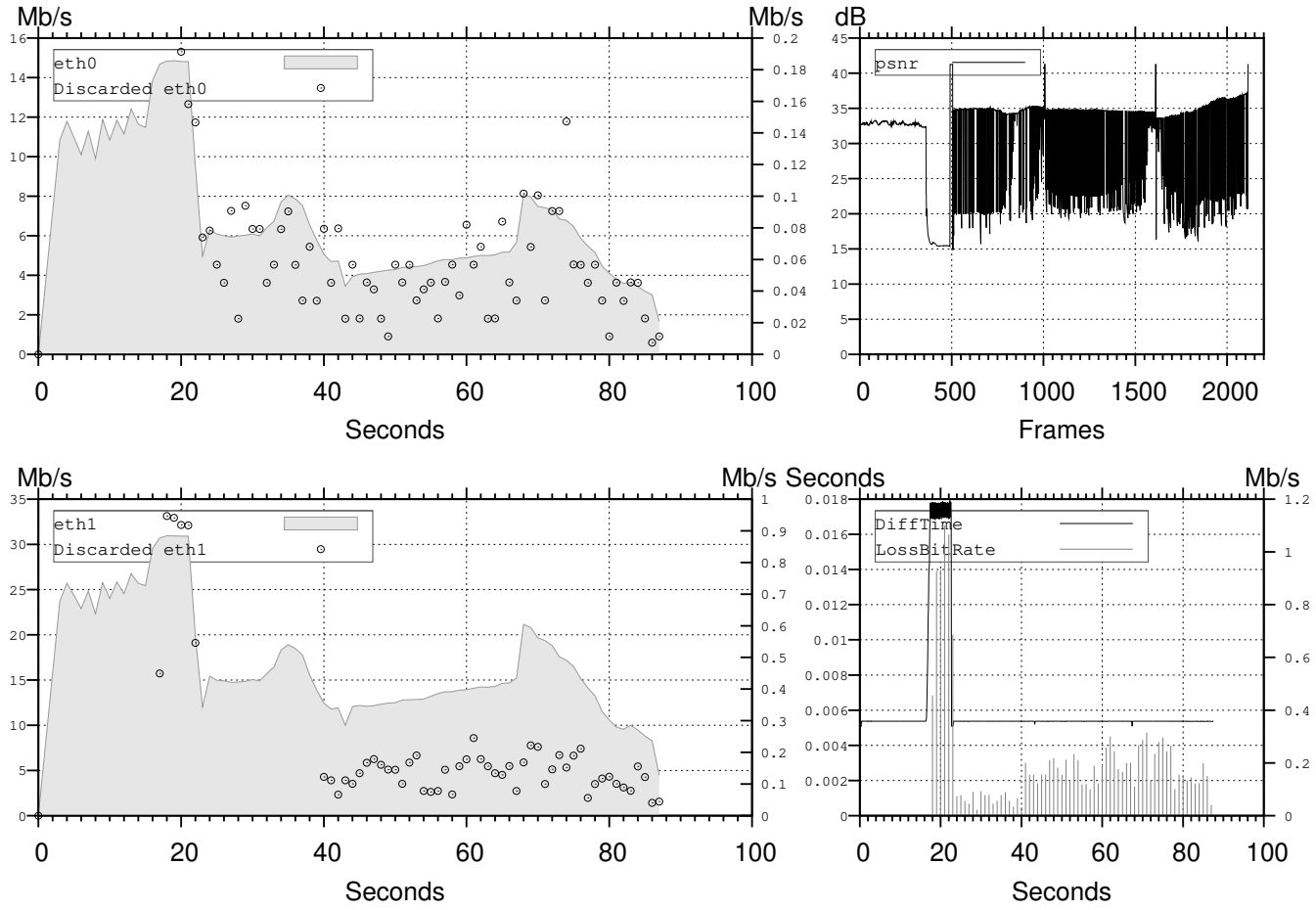


Figure 3.10: Uniform 10^{-2} 30 Mbps RTPMulti.

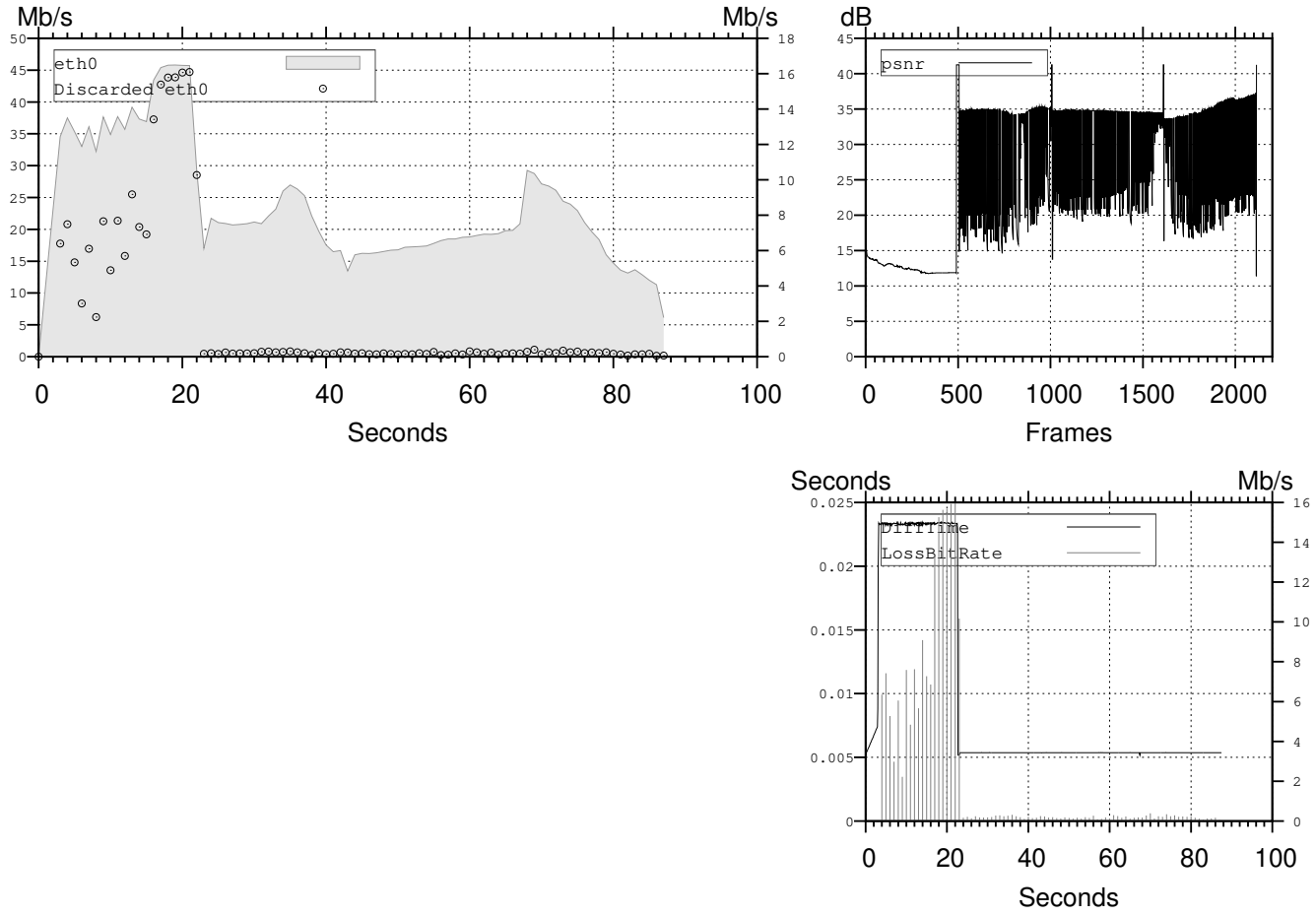


Figure 3.11: Uniform 10^{-2} 30 Mbps RTP.

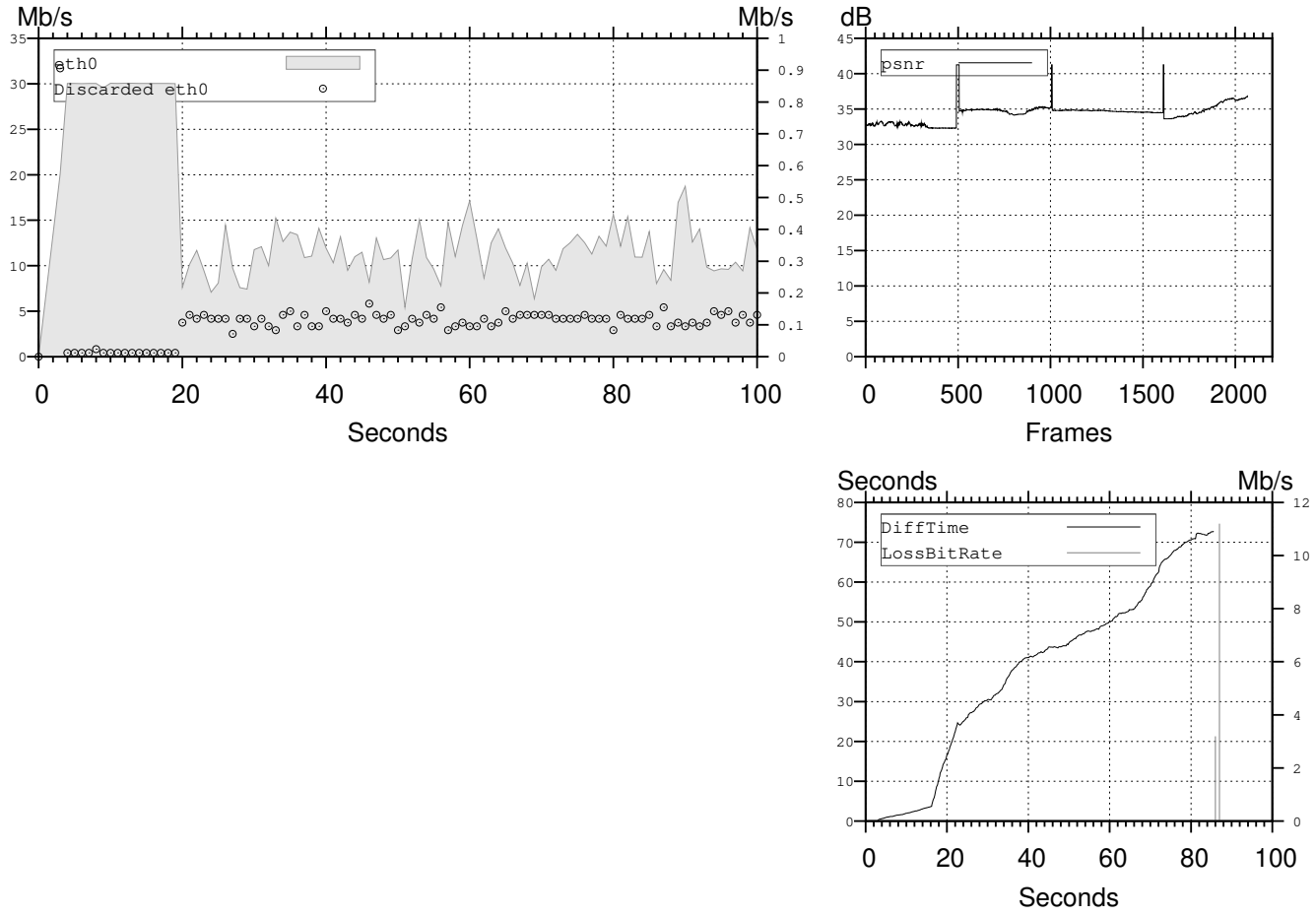


Figure 3.12: Uniform 10^{-2} 30 Mbps TCP.

3.4 Conclusions

SCTP has been one of the most promising protocols for multimedia communications and is still employed as part of conferencing systems. Its massive deployment has only been delayed by the sudden increase in the last-mile bandwidth capabilities that have reanimated TCP thanks to the adoption of the more comfortable HyperText Transfer Protocol (HTTP) protocol. Still the capabilities offered by CMT-SCTP are not offered to the best of my knowledge by any alternative and even though nowadays connections have increased their available bandwidth, employing multiple connections has still some relevance for other use cases (not bandwidth limitations) like economical saving options. H.264/SVC never reached the masses but has been employed in production scenarios and what is more important, has been superseded by its next generation homonym Scalable High-Efficiency Video Coding (SHVC).

The presented results show empirically how mixing SCTP streams with H.264/SVC layers is a natural and profitable approach for video delivery in networks with high error rates. This study could be enhanced by tuning the configurable SCTP parameters such as the reliability level or the path and association retransmission limits, among others. The incorporation of other extensions such as the Stream Control Transmission Protocol Potentially Failed (SCTP-PF) [4] offering a non-duplicate transport service with tunable loss recovery linked with the importance of a layer for the actual operation point. Enhancing the video characteristics by employing High Definition (HD) streams and at least temporal scalability would have been also desirable and the limitations imposed by software decoding vanished thanks to processor evolution and software decoder maturity, not to speak about hardware coding.

Finally, a real implementation of these transmission techniques could permit to contrast the simulation results obtained in this study.

Chapter 4

Video transmission in the Future Internet (FI)

Video streaming as a flow of time tagged (RTP) datagrams (UDP) has been slowly replaced by a less logical connection oriented methodology (TCP). The adoption of Real-time Messaging Protocol (RTMP) first and HTTP based video streaming techniques later was backed up by the increase in network bandwidth. At the same time, new paradigms have appeared.

The FI introduces new concepts, solutions and architectures that impose new challenges to the existing services. Clearly, most FI architectures imply a clean-slate adoption, therefore their evaluation should not only be focused in bulk transmission capabilities but also taking into account application payloads. In that sense, video streaming is one of the more resource hungry payloads while also being one of the more popular. The adoption of HTTP as a basis for transmitting almost anything, regardless its adequacy or not for certain traffic, supported by the increasingly available networking bandwidths, makes it the ideal candidate to start with this application evaluation. Therefore, with the support of the ANA4IoT project [103], three promising and rather independent architectures are evaluated. Heterogeneity Inclusion and Mobility Adaptation through Locator ID Separation (HIMALIS) as the representative of clean-slate architecture to break IP ossification, Content-Centric Networking (CCN) as an overlay to evolve the 'Net' into the Information Centric Networking (ICN) model and finally the Internet of Things (IoT) which imports restrictions from the deployment and the hardware back to the network and its logic.

4.1 Description

The main objective of the ANA4IoT [103] experiment within the OpenLab project has been to determine the adequacy of different architecture proposals for the Future Internet when applied to typical scenarios of the IoT. At the same time, and as expected outcome, valuable feedback was obtained that may help the designers and developers of both the selected architectures and the infrastructures used to run the experiments to improve them. The analysis is focused on HIMALIS [83] and CCN [71], which are two outstanding architectures that offer different networking views for the Future Internet. The HIMALIS architecture is based on end-to-end communications but resolves the drawbacks found in the current Internet by separating global and local routing, so the general routing scalability is improved, and by separating identifiers from locators, so achieving indirection to seamlessly support mobility and multi-homing. The CCN architecture offers a totally different view of the network. It is designed following the ICN approach and, thus, places information pieces (content) as the central element of the network, making clients declare their "interest" on content pieces and providers to offer and deliver them to the intermediate network elements which, in turn, collaborate to deliver the requested content pieces to those clients.
Using prototype implementations of the aforementioned architectures a batch of tests (sub-experiments) were performed that exercised some specific capabilities required both by IoT scenarios and, in general, by the Future Internet. To get approximate results to those obtained in the real world and when running those architectures in real networks, the tests were carried out with different and heterogeneous topologies built on top of GAIA [12] [104], Heterogeneous Experimental Network (HEN) [105], PlanetLab Europe (PLE) [106]. In some sub-experiments, interconnecting those architectures to get richer observations and measurements was intended. The feedback extracted from this project will be also valuable for the different testbeds to know the difficulties that each one introduces to the experimenter.
In this group of experiments how reducing the Maximum Transmission Unit (Maximum Transmission Unit (MTU)) of the network was studied in relation to how affects to different aspects and elements of the deployed architectures. This gave us a profile of their behavior when running in constrained networks, like the typically found and associated to IoT. In particular, different values for the MTU were used and, for each one, the performance of both the global transit network (the Internet) and the access network was demonstrated. This means that the bandwidth, throughput, and latency for each situation and section of the network was obtained.
The main hypothesis aimed to be demonstrated or invalidated with this experiment is that

IoT workloads require specific support from network architectures because of its small packet size. Performance descriptors are not just hurted by the extra headers introduced but also by the behavior of intermediate switching and routing machinery regarding their queuing policies. Demonstrating this will help to improve future designs by, for instance, aggregating traffic or folding packet headers into a minimal flow identification header. Moreover, analyzing the results obtained from this experiment the minimum MTU required for each architecture is determined in order to work properly with IoT workload, which is an important aspect of IoT networks because they have had constrained resources, reflected in the reduced packet size introduced above. Moreover, for each performance descriptor, mainly throughput and latency.
Apart from the network profiling, the results were compared and analyzed to determine the benefits and drawbacks when using different values for the MTU of the network. Furthermore, the effect of network size and topology were studied through the performance descriptors by experimenting with different topologies and network sizes.

4.2 Testbeds and tools

The GAIA testbed, short name of the GAIA Extended Research Infrastructure, is located at the southeast of Spain. It targets the research of Future Internet architectures and comprises several facilities from the University of Murcia and the Spanish government. It offers a vertical infrastructure, composed of a backend with high capacity of data storage, communication, and processing, together with a frontend with an extended set of multidisciplinary testbeds, deployments, and living labs for the ubiquitous monitoring, sensing, and processing. That said, it offers a highly flexible framework for experimentation with architectures and protocols for the Future Internet. In fact, it has been used in many research projects to evaluate their outputs from the communications and telematics point of view.
PLE is the European arm of the global PlanetLab system, the world's largest research networking testbed, which gives users access to Internet-connected Linux virtual machines on over 1000 networked servers located in the United States, Europe, Asia, and elsewhere. PLE is being currently developed by the OneLab initiative. The PlanetLab Europe Consortium has 150 signed member institutions: mostly universities and industrial research laboratories, each of which hosts a minimum of two servers that are made available to the global system. These institutions are home to near 1000 users. On a typical recent day, around 250 were connected to on-going experiments.

HEN (Heterogeneous Experimental Network) is a testbed designed for conducting a wide range of network experiments, from congestion control, large-scale routing, Denial-of-Service and others to experiments in mobile systems and sensor networks. To achieve this, the core of HEN will consist of about 80 to 100 computer nodes, each having 4 Gigabit Ethernet experimental interfaces and one Gigabit Ethernet management interface used for net-booting the node, controlling the experiment, and backing up the results. In addition to the computer nodes, a number of routers and other networking equipment will be included in HEN as experimentation equipment. The testbed will also include a large amount of wireless equipment, including 802.11 and sensors.
Regarding experimentation tools the conclusions are that, from the tools offered as part of the OpenLab project, the Network Experimentation Programming Interface (NEPI) [107] is the one that best fits with the requirements. It permits to abstract the experiment definition from the experiment execution in a very simple an easy way. It currently supports PLE, and will support other platforms as it gains support for OMF. With such new version, those experiments that involve many testbeds can be directly integrated through NEPI.

4.3 IoT video in FI

4.3.1 Generating IoT video

In order to evaluate how video is consumed within an IoT network we have created some video resources limiting the maximum slice size of the H264/AVC [36] streams to 81 bytes. This means that this slices could be directly transported over the typical constrained network protocols used by IoT, such as 6LOWPAN [108], without the need of introducing fragmentation with the consequent packet aggregation at the receiver. In addition, we encoded the videos trying not to exceed 250 kbps of bit-rate to cope with the top speed specified by the IEEE 802.15.4 [109] standard.
The video used has been encoded using the JSVM 9.19 reference software [110]. In addition, limiting the codec to intra frames was decided (frames encoded using information from within themselves) because the less powerful things would only be able to keep into memory one decoded frame, either to be encoded or to be displayed. For the same reason, the frame size has also been limited to CIF. In case that the thing would be native video capable, this is, the thing has some specific hardware to encode/decode video such as a camera already capable of producing encoded video, the thing would not need to store the frame in its own memory and would not need to do the encoding/decoding job but in any case it is regarded that prediction, motion estimation, and compensation procedures could lead

to battery draining (even with specific hardware) which is something to be avoided. Just like that, the frame rate of the videos has been limited to 5 frames per second.
That said, the video selected in the experiment is a concatenation of the very well known bridge and bridge far test videos [111] with a length of 4100 frames, which give us 820 frames at 5 FPS. This results in 164 seconds of video.
These videos were selected for their slow motion which makes them similar to videos obtained by a surveillance camera or similar environments in which low power things could probably get involved. To encode the video with different bit-rates different static QP (Quantization Parameter) have been used producing average bitrates of 38, 58, and 79 kbps. The average PSNR values are 28.9331 Y, 30.2404 U and 31.2660 V for luma and both chroma respectively. As can be seen in Figure 4.8 the PSNR is increased from frame 400 onwards. At that point is where the second concatenated video starts.
Although adaptation is one of the key features of DASH, to carry out the experiments specific bit-rates per iteration have been used. This will ease the data analysis. The different techniques to be developed to decide the adaptation logic are completely out of the scope of this project.

4.3.1.1 Dynamic Adaptive Streaming over HTTP (DASH) encapsulation

To create the DASH streams, the "libdash" [112] implementation and DASHEncoder from ITEC-UNIKLU were used. The DASHEncoder software is intended to read the raw input, encode it with x264 [113] and generate the DASH video with its slices. In this case, an already encoded stream is provided as an input. Therefore, the DASHEncoder sources have been slightly modified avoiding the x264 encoder part and all the related code and substituting it by a link to the encoded file.
The reason why the unmodified software was not used as it is, is that to the best of my knowledge the x264 implementation did not accept the maximum slice size parameter at the moment that the experiments were carried out, which was used to simulate the IoT video source.
On the other hand in order to be able to use reduced MTU, simple URL compression methods were used to avoid exceeding the MTU size. In order to do it, the operating system hosts file was tuned to redirect a only one letter host name to the web server in CCN cases. Unlike in CCN, in HIMALIS it is a bit different as the requests must go always to the local adapter so localhost is always used.
As it is shown in Figure 4.1, the DASH chunk sizes vary depending on the part of the video involved. As can be observed the first 200 files have over 4.5K size while from file

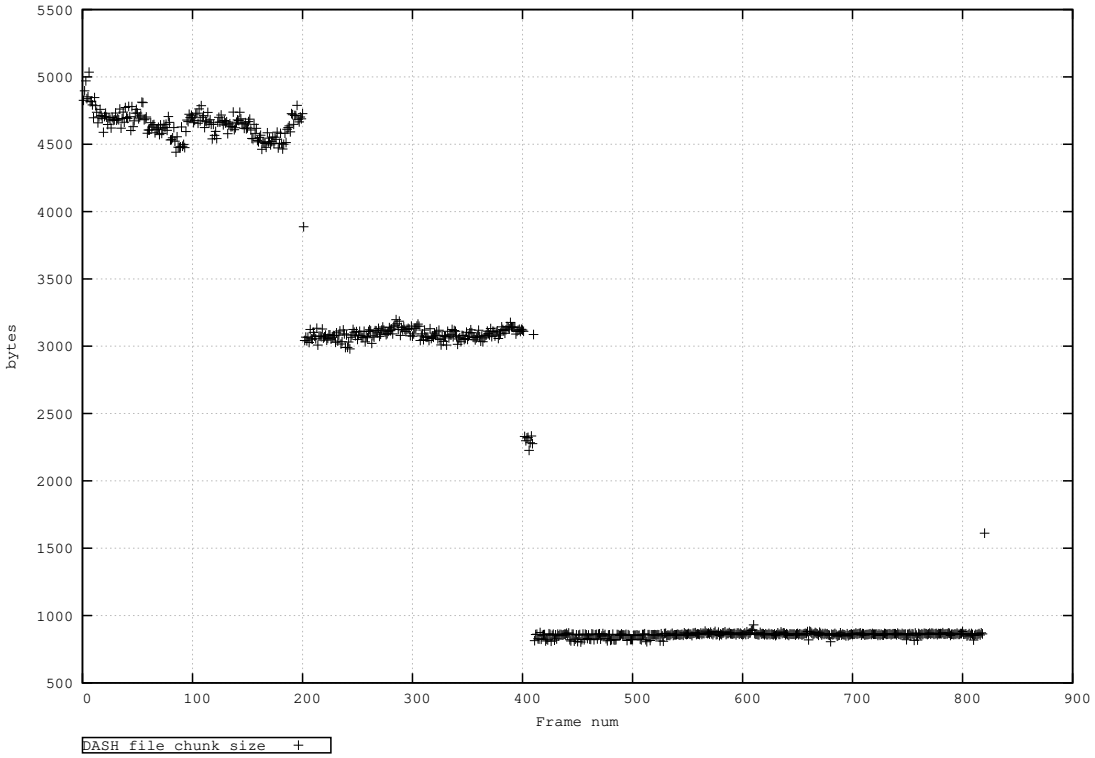


Figure 4.1: DASH chunk file and size relation.

400 the file size is reduced drastically to 1K. It is clear taking a look at Figures 53 and 54 the relation between the PSNR and the DASH chunk size.

4.3.2 HIMALIS

In this section, HIMALIS is introduced and evaluated as the FI architecture solution for IoT enabled video in a multi-domain environment. The HIMALIS architecture is deployed on top of the PLE distributed testbed. The deployment and the later experimentation on top of the architecture is scheduled and performed by means of Network Experimentation Programming Interface (NEPI), ensuring the repeatability and validity of the results.
Deploying HIMALIS on top of PLE ensure the validity of the results, not only because of the possibilities in terms of repeatability that a public and open testbed offers but also due to the fact that PLE is deeply geographically distributed across Europe. NEPI on the other hand, not only helps in terms of facilitating repeatability to other researchers but also in the extraction of the here presented results. Thanks to NEPI the testbed can be clean-slate deployed on each experiment, in addition, synchronization mechanisms for launching each part of the experiment are provided, e.g. so that the client does not try to start the connection before the server is launched. Finally, NEPI offers with the means to retrieve the distributed experimentation data, such as application logs, that are analyzed to present the results.
DASH is the application with which the HIMALIS architecture is evaluated, differentiating this study from others where less representative applications like simple Internet Control Message Protocol (ICMP) messages or bulk transmission. Using DASH video streaming provides information about the feasibility and performance evaluation for the two more representative technologies being used and likely relevant for the foreseeable future like HTTP and Video. The third future technology not covered directly by DASH is IoT. To cover that spot in the experiments, the videos encapsulated into DASH comply with what an IoT network is able to transport in terms of packet size and bit-rate. In addition, the performance analysis of the architecture has gone further in the direction of IoT with the reduction of the MTU to values typical in these kind of networks.
The HIMALIS software was provided by the National Institute of Information and Communications Technology (NICT, Japan). The software is split in two different planes. The signaling plane leverages on web services and allows node and connection management. The data plane is c++ based and is triggered by the signaling plane to notify the GW nodes the required forwarding characteristics serving as well as endpoint software.

4.3.2.1 Experimentation summary

Achieving successful HIMALIS deployment and data transmission on top of PLE has a certain degree of randomness. A successful experiment is considered when the full DASH stream is transmitted from the server to the client. A minimum of 4 successful experiments have been considered to continue with the next experiment parametrization. The number of 4 comes from the experience in terms of timing. Also from that experience, the PLE slice providing the list of nodes was trimmed without those nodes that were producing recursive failures and weekends were prioritized for experimentation hence the good results of some of the experiments as detailed in Table 4.1. The table specifies the number of experiment launch performed in column *# Launch* while *# OK* denote the number of successful experiments achieved. From those experiments not belonging to the later, *# PlanetLab NOK* and *# HIMALIS NOK* account the failures due to PLE problems and HIMALIS initialization or data transmission errors respectively.
There is no clear relation between the failures and the achievements with the number of launches and the MTU which was predictable since the signaling and deployment is not affected by the reduction in the MTU.

	# Launch	#PlanetLab NOK	# HIMALIS NOK	# OK
Domain1-Domain5 1024 bytes	53	34	15	4
Domain1-Domain5 512 bytes	59	32	23	4
Domain1-Domain5 128 bytes	16	11	2	4
Domain1-Domain5 50 bytes	74	46	22	6
Domain1-Domain4 512 bytes	15	8	2	5
Domain1-Domain4 128 bytes	13	0	3	10
Domain1-Domain4 50 bytes	14	5	2	7

Table 4.1: HIMALIS deployment and DASH experiment summary.

4.3.2.2 HIMALIS Scenario Deployed on PLE

The scenario deployed on top of PLE is that of a pentagon (as shown in Figure 4.2) representing a backbone ring with one local domain per vertex. Each local domain consists of a HNR/DNR/GW node that holds the HIMALIS' control and data planes functionality. A node acting as a switch to which the nodes acting as clients are connected through *vtun* tunnels, as well as the HNR/DNR/GW does.
It must be highlighted that although in the topology two nodes might be neighbors one from the other, in terms of geographical location the nodes might be hundreds miles away

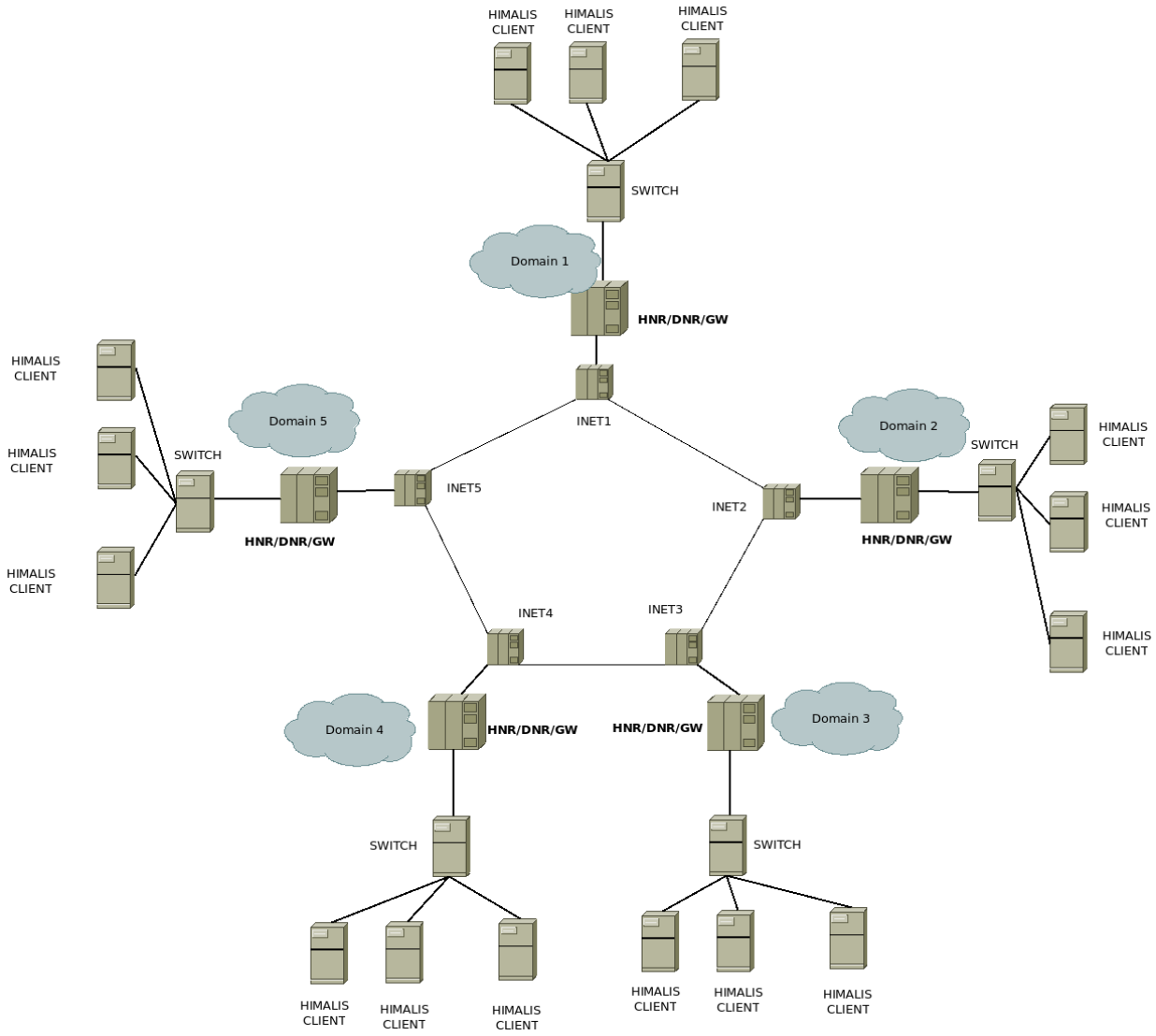


Figure 4.2: Himalis PlanetLab deployed scenario.

from each other. It is therefore possible that the Round Trip Time (RTT) between two clients of neighboring domains might be bigger than the RTT from two clients with multiple jumps in between them. The reason for this variability is the randomness introduced by NEPI on the node selection algorithm for the deployment phase. The experiment defines the number of nodes to be used and NEPI extracts them from a poll of available nodes, which in turn is requested from the PLE slice associated to the researcher. Then it is up to the experiment to assign a role to each machine which in this case is not based on geographical location.
The HIMALIS implementation deployed for these series of experiments provides non-connection oriented (UDP) data tunneling on top of the HIMALIS architecture, hence the data transmission is unreliable. To overcome that limitation, a rather simplistic approach of a flow control has been implemented, see Figure 4.4 to avoid data loss and synchronize data transmission. Each HIMALIS Packet consists of 44 bytes of HIMALIS fixed header and 5 bytes used for app signaling, including 1 byte message code, 2 bytes sequence number, 2 bytes last sequence number to be received as shown in Figure 4.3.
The application layer flow control introduced above is not network agnostic and in addition both, the DASH client and the DASH server which in turn are an HTTP client and server correspondingly. Hence, an adapter that speaks TCP/HTTP on one side and UDP/HIMALIS on the other side is needed. This adapter is also in charge of introducing the application specific flow control header and/or remove it on arrival. On one side the HIMALIS HTTP Adapter Client receives HTTP requests and forward them to the corresponding HIMALIS HTTP Adapter Server. The HIMALIS HTTP Adapter Server receives the request and performs a complete HTTP transaction with a standard HTTP server sending back the result. The flow control mechanism retries up to 3 times to recover a lost application layer packet before leaving it as unrecoverable. In fact the video player would discard any DASH chunk that arrives after the current playing instant so it is worthless keep trying to download such chunk.
To clarify the entities involved in the communication and the protocols being used and transported on each entity the Figure 4.5 was introduced.

4.3.2.3 DASH streaming - Base Case

To evaluate the feasibility of DASH video transmission over HIMALIS with IoT restrictions, first the base case for standard DASH video transmission needed to be performed. To that end, transmission of DASH video with a packet size of 1024 between two neighbor HIMALIS domains was performed. From the deployment shown in Figure 4.2, the domains 1 and 5

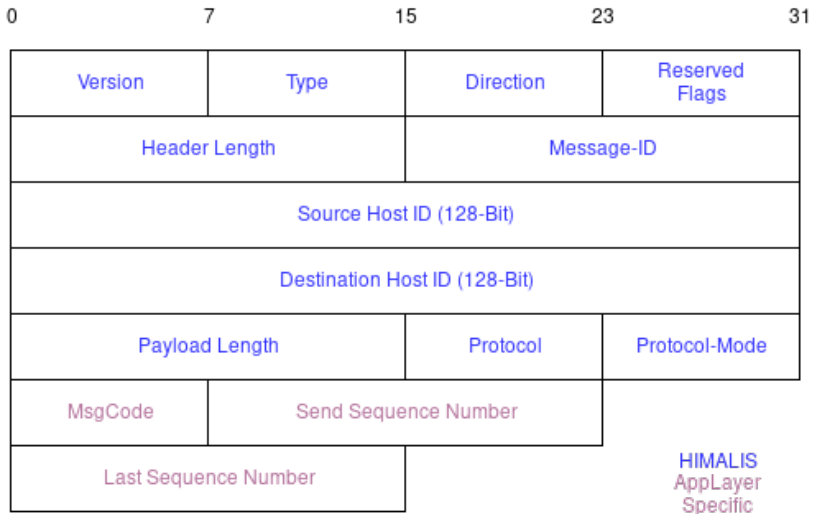


Figure 4.3: HIMALIS and flow control header detail.

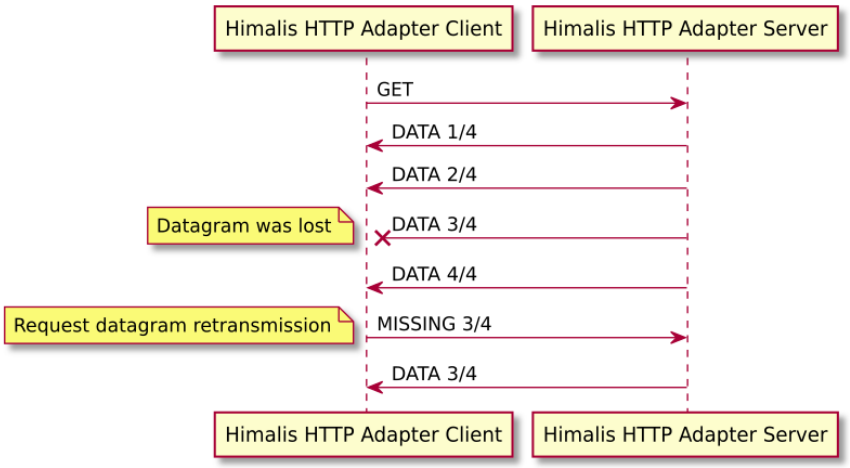


Figure 4.4: App layer transmission protocol used on top of HIMALIS

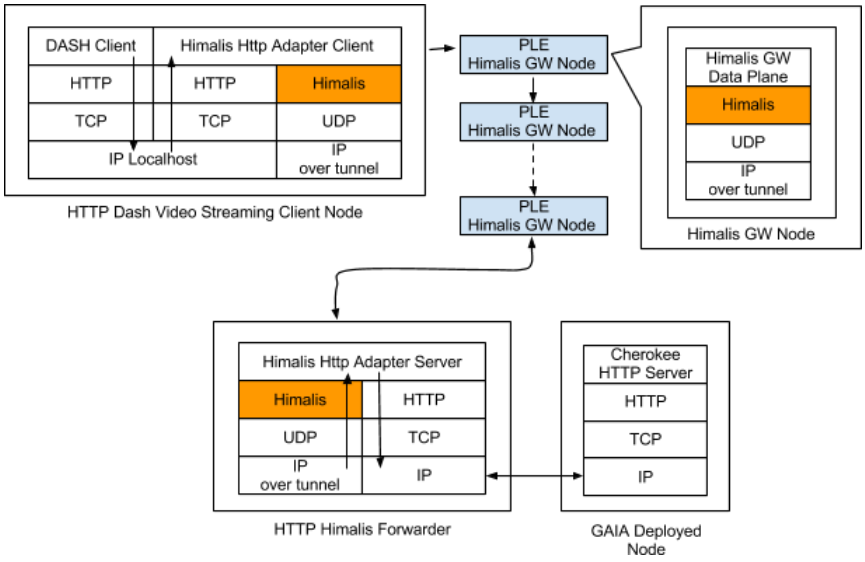


Figure 4.5: Himalis deployment entities and relations

are used. The biggest down rate is performed from domain 5 to domain 1, meaning that the HTTP adapter server is collocated in one of the HIMALIS client nodes in domain 5 while the HTTP adapter client and the client itself are deployed in domain 1. It is worth to note that the HTTP server containing the video bit-stream is located in Gaia testbed in the University of Murcia, the communication between the domain 5's HTTP adapter server and the server itself is performed through the Internet without tunneling or any other means.

Figure 4.6 represents the retrieval time of 4 fruitful experiments. The term fruitful here means that HIMALIS scenario was deployed from scratch on top of PLE, the control plane was triggered correctly so that the associations between the domains and each domain entities were performed correctly and that finally the video played from the client. In all the experiments it is easily seen that there is a steep increase in the time needed to download the chunks from the server. For a small amount of data, the retrieval time is increased which corresponds exactly with the retrieval of the second part of the video in which the sequence *Bridge Far* takes place. At that point and as can be seen in the Figure 4.1 the relation between the size for each chunk is reduced drastically which means that more retrieval processes need to be issued hence producing more overload per data byte transmitted. In fact, although not so noticeable, from 1000 ilobytes onwards there is already a small change in the slope which corresponds to the frames between 200 and 400 also on Figure 4.1.

In addition to the data retrieval time, Table 4.2 shows information related to the installation

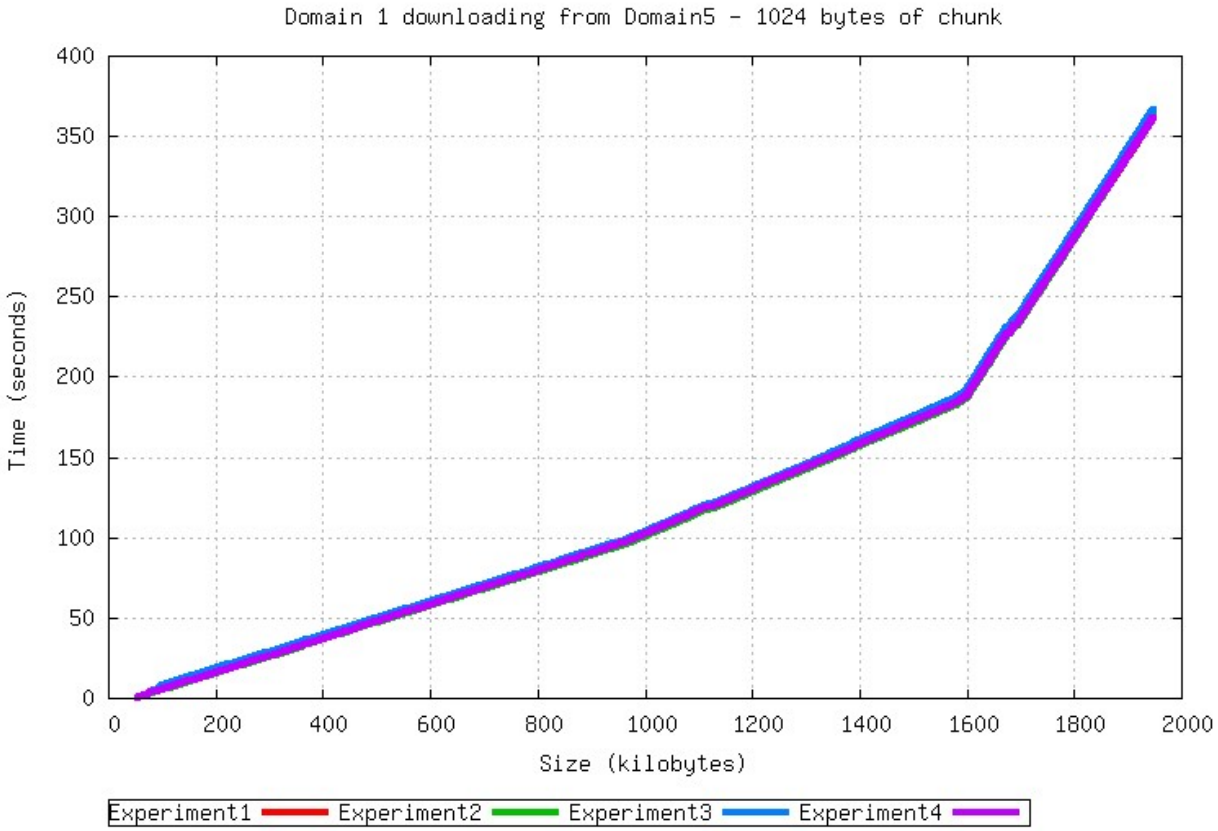


Figure 4.6: DASH over HIMALIS transmission in neighbor domains.

time of all the entities involved in the experiment and the connection time for the infrastructure to finally be ready to transmit data. The achievement of the ready state depends on the distribution of the assigned PLE nodes as well as the processor and memory load of those nodes. The HIMALIS signaling plane takes some time to be stable, we perform reconnect retries until we obtain a successful connection. Depending on the load of the HNR/DNR/GW machine and the Client machine (as it is the local tomcat the one in charge to communicate with the remote tomcat), differences in times may occur, as is seen in Table 4.2.

Install	Connect	Data Retrieval	Size
625.039718 sec	40.094038 sec	366.095788925 sec	1994004 bytes
626.834500 sec	40.062966 sec	361.223042294 sec	1994247 bytes
626.999327 sec	40.090127 sec	362.135886942 sec	1994247 bytes
639.830867 sec	13.356959 sec	361.417529530 sec	1994247 bytes

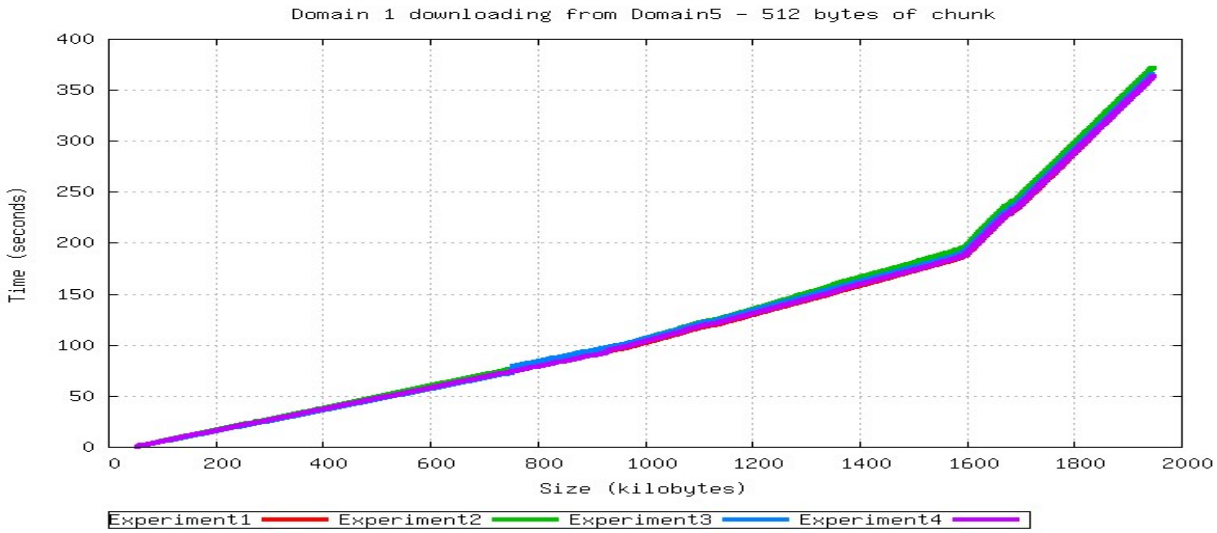
Table 4.2: DASH over HIMALIS transmission in neighbor domains timetable.

4.3.2.4 Video for IoT

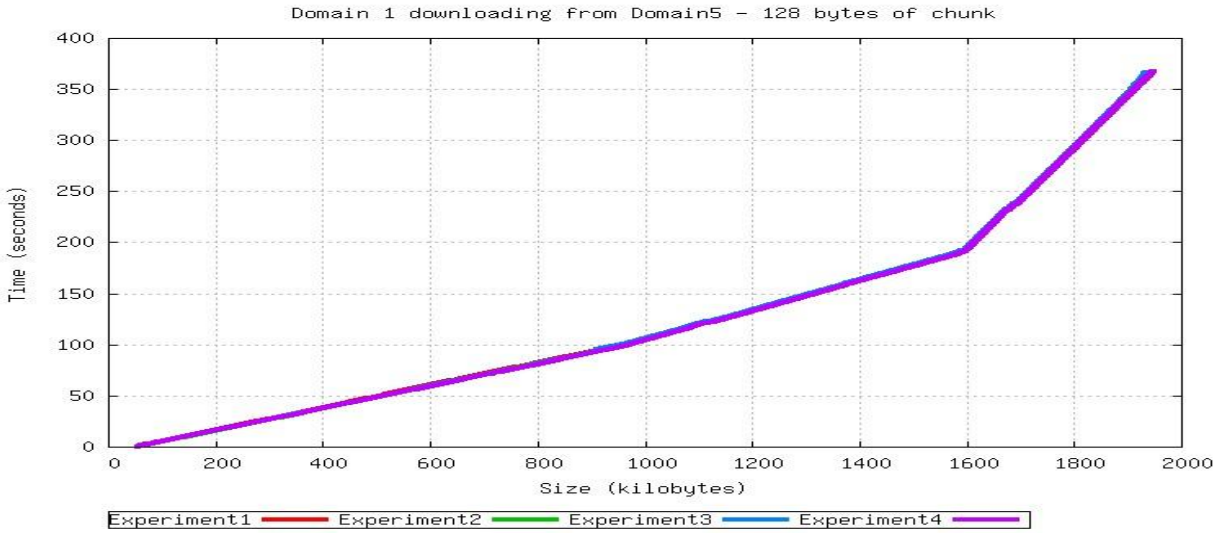
Once the base case for transmitting IoT video encapsulated into DASH over HIMALIS have been performed, next step is to investigate the effect of the underlying network MTU size restrictions in the same use case. For these series of experiments, the restriction is applied only to the data plane and not for the signaling. First of all, the data plane packets would usually be the ones to reach the edge of the network where IoT devices are deployed. Consider a HNR/DNR restricted to that kind of networks is not considered probable. At last, from the experience gained in the previous experiments, deploying the scenarios has been hard enough with proper signaling capabilities.

4.3.2.4.1 Neighbor Domains First series of experiments are performed as an extension of the experiments introduced in section 4.3.2.3. Neighbor domains 1 and 5 and therefore used for the transmission and MTU is reduced to values of 512, 128 and 50 bytes respectively.

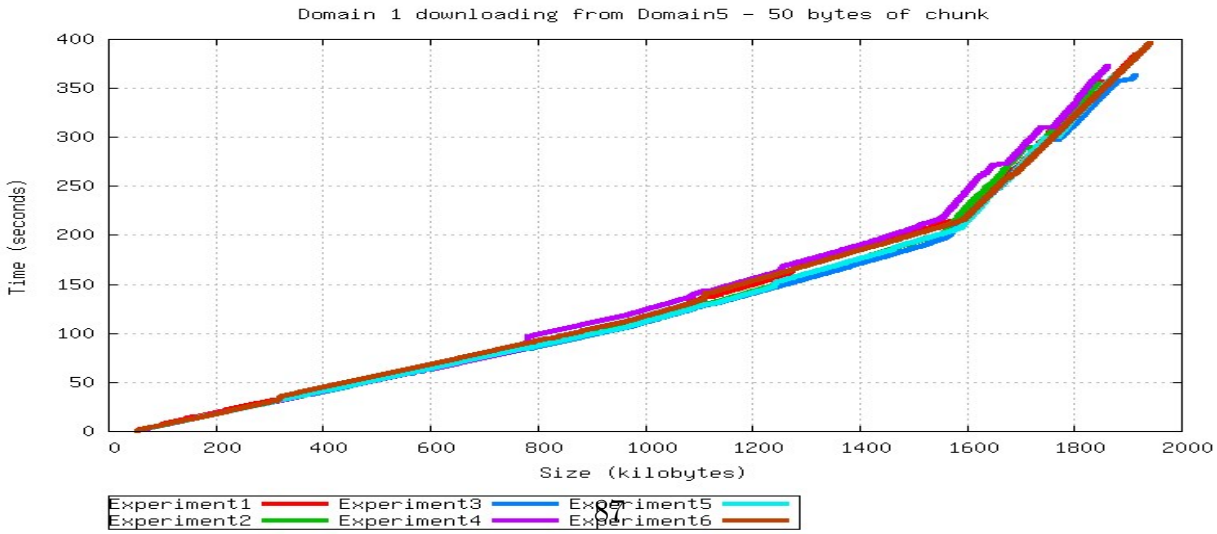
The first experiment is a continuation of the experiment described above. The video is retrieved from a node in Domain 5, which in turn, retrieves the content from the web server located in GAIA, to a node in Domain 1. These nodes despite the nomenclature are neighbor domains as there is only one HIMALIS hop between them, no other GW is involved between the GWs being used. Figure 4.2 shows a representation of the scenario.



(a) Domain 1 to Domain 5 transmission with 512 bytes MTU



(b) Domain 1 to Domain 5 transmission with 128 bytes MTU



(c) Domain 1 to Domain 5 transmission with 50 bytes MTU

Figure 4.7: MTU variation study in HIMALIS neighbor domains

MTU	Install	Connect	Data Retrieval	Size
Experiment1 512 bytes	629.7680 sec	39.8204 sec	362.9083 sec	1993739 bytes
Experiment2 512 bytes	610.4218 sec	39.8790 sec	371.8765 sec	1993771 bytes
Experiment3 512 bytes	610.5264 sec	39.8979 sec	365.3348 sec	1992872 bytes
Experiment4 512 bytes	617.7914 sec	39.9150 sec	362.9633 sec	1993865 bytes
Experiment1 128 bytes	619.6756 sec	40.0530 sec	367.1119 sec	1994165 bytes
Experiment2 128 bytes	621.7051 sec	39.8445 sec	366.1240 sec	1980308 bytes
Experiment3 128 bytes	639.7902 sec	39.9862 sec	366.3618 sec	1980400 bytes
Experiment4 128 bytes	628.6526 sec	39.8701 sec	367.7630 sec	1994180 bytes
Experiment1 50 bytes	613.2309 sec	40.0539 sec	396.2406 sec	1989440 bytes
Experiment2 50 bytes	610.3639 sec	40.0590 sec	372.4797 sec	1908269 bytes
Experiment3 50 bytes	600.3610 sec	40.0801 sec	383.6383 sec	1961807 bytes
Experiment4 50 bytes	613.3122 sec	40.0250 sec	363.3388 sec	1960619 bytes
Experiment5 50 bytes	613.8812 sec	40.0553 sec	354.4483 sec	1889848 bytes
Experiment6 50 bytes	599.5804 sec	40.0678 sec	362.7165 sec	1921674 bytes

Table 4.3: DASH over HIMALIS transmission in neighbor domains MTU variation study timetable.

Figures 4.7a, 4.7b and 4.7c represent the analysis of download time analog to the one shown in Figure 4.6 but for a limited MTU of 512, 128 and 50 bytes respectively. As can be easily observed in the figures, the results for 512 and 128 bytes are really similar to those of the base case and only for 50 bytes restriction can be observed a real difference which is even more noticeable on the last part of the video source where the number DASH chunks is increased with an average reduction in size. It must be noted that a chunk is not considered downloaded until all the app layer flow control fragments (See Figure 4.3) arrive the destination. Despite parallel transmission would be possible, in these experiments another DASH chunk is not downloaded until previous chunks have finished which, corresponding with the worst case in terms of parallelism, is the worst approach a client could adopt, again, the idea is to approach to the worst case understanding that results could be enhanced by means of proper pipelining in the client side.
On the other hand, it was expected that the overload produced by fragmentation would have greater impact on the overall transmission which would be increased due to the lack of pipelining. Nevertheless, the transmission of the app layer flow control fragments is parallelized affecting the transmission only if a packet is loss therefore forcing the retransmission and consequently delaying the start of next DASH chunk download.
In addition to graphical assessment provided by the figures, Table 4.3 details information about the retrieval time as well as deployment and signaling time as was already done in

Table 4.2. For MTU of 512 bytes a mean of 365.77 seconds was obtained while retrieving the data with a standard deviation of 4.22. For 128 bytes the mean is 366.84 with a standard deviation of 0.74. The variability is lower on the second case although in average the amount of time expend was higher. For 50 bytes the mean is increased to 372.14 and also the standard deviation roses up to 15.44.
Looking at the amount of data retrieved for all the experiments in Table 4.3, one can observe that for 50 bytes experiments there is a general downward trend. This trend affects negatively to QoE since missing packets result in video decoding issues. A common practice in video decoders is to duplicate previous frame in case of non decodable frame, the effect for the viewer is still image which is in general better than an empty image. As an example of this effect, an analysis of the PSNR for Figure 4.7c 'Experiment 5' and 'Experiment5 50 bytes' from Table 4.3, the experiment with more losses, is shown in Figure 4.8. The differences with the original PSNR marked with the red cross are not big partly because of the frame duplication mechanism introduced above. As was already stated in Section 4.3.1, the video is like the one obtained from a surveillance camera and because of that, anything that is not moving is exact to the previous frames. In this case in the video the only movement are the waves from water as well as the walkers over the bridge. Since copies of previous frames are introduced in case of missing frames, the difference in the PSNR is not too high in value but from the point of view of human senses one sees how the walkers would jump backward when frames are lost and previous ones are introduced.

4.3.2.4.2 3-domain setup In order to evaluate the influence of inter-domain HIMALIS routing and signaling, the neighboring domains experiment is extended by introducing another domain in between them, thus having 3 domains involved and therefore having the same amount of gateways. For this series of experiments the communication is established by Domains 1 and 4 while Domain 5 acts as relying party.
Looking at Figures 4.9a, 4.9b and 4.9c its easy comparing with Figures 4.7a, 4.7b and and 4.7c that there is no effect in having one relying party in the communication, the results are very similar with what has been shown for the neighboring domains. Only remarkable fact is that in this case the stability achieved in the testbed was higher, therefore more successful results were obtained. Only a few cases where HIMALIS was successfully deployed and the communication between the two endpoints could be established are worth mentioning, 'Experiment2 128 bytes', 'Experiment6 128 bytes', 'Experiment6 50 bytes' and 'Experiment7 50 bytes'. Looking in detail at the experimentation log, the conclusion is that communication was impossible after a certain point in the communication or synchronization failure on the app layer flow control mechanism occurred . For example

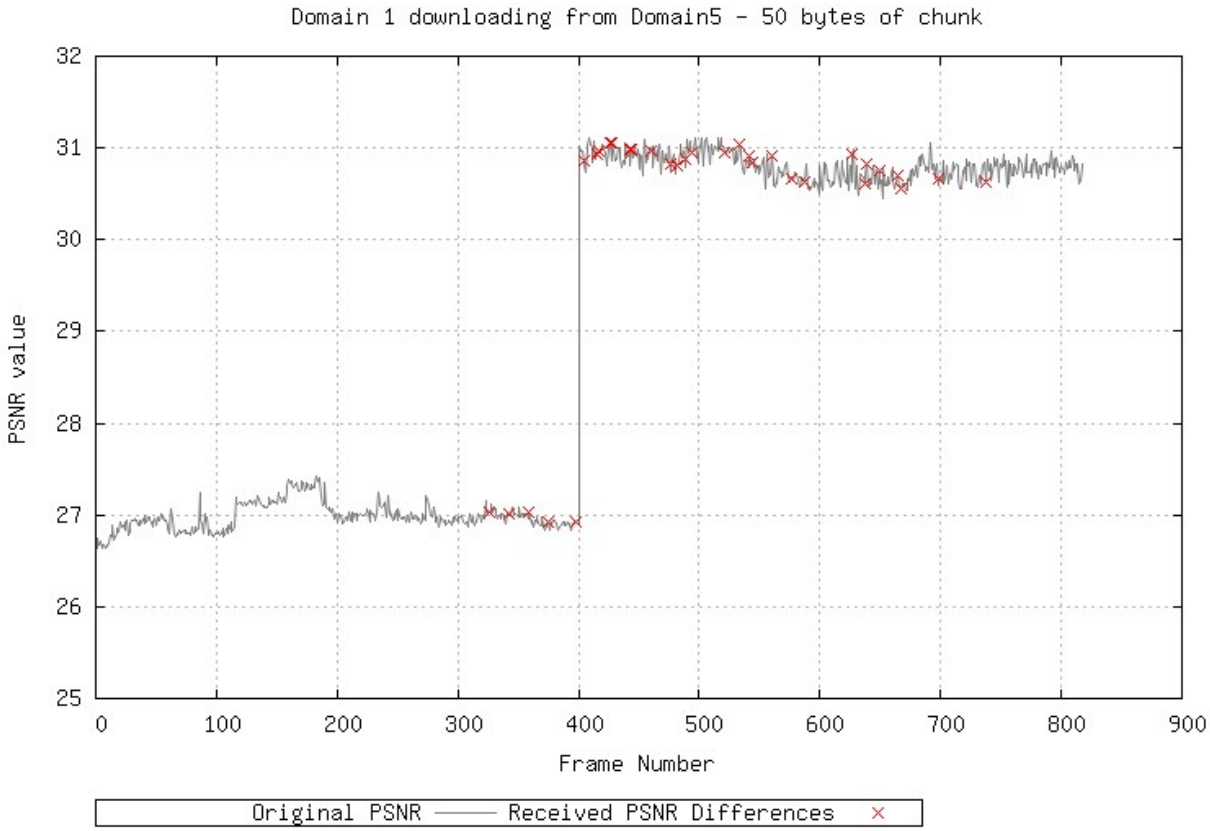
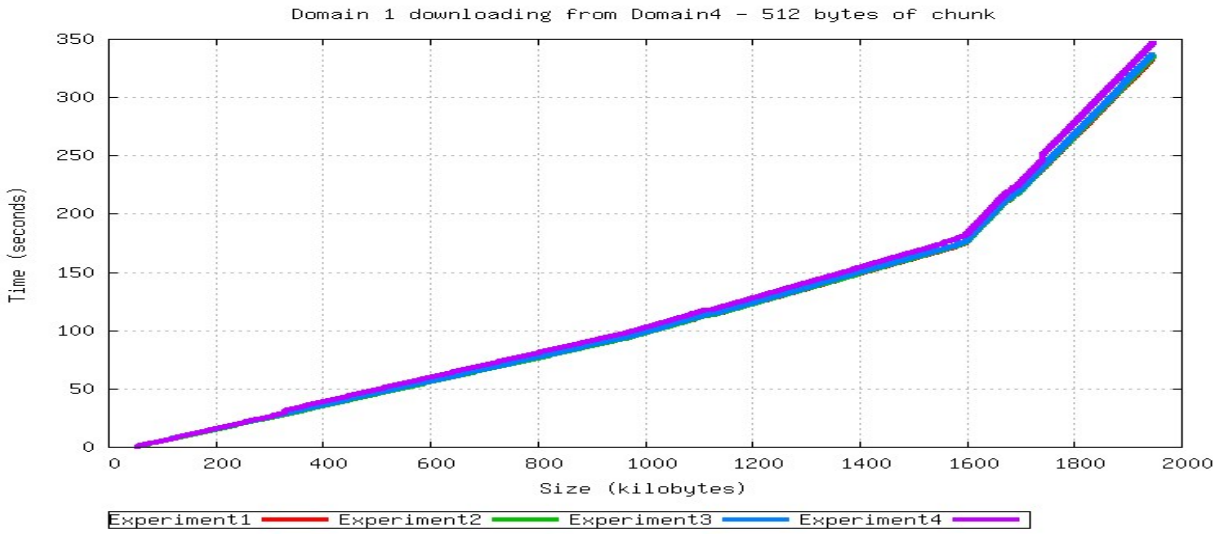


Figure 4.8: Luma PSNR values for original and retrieved video.

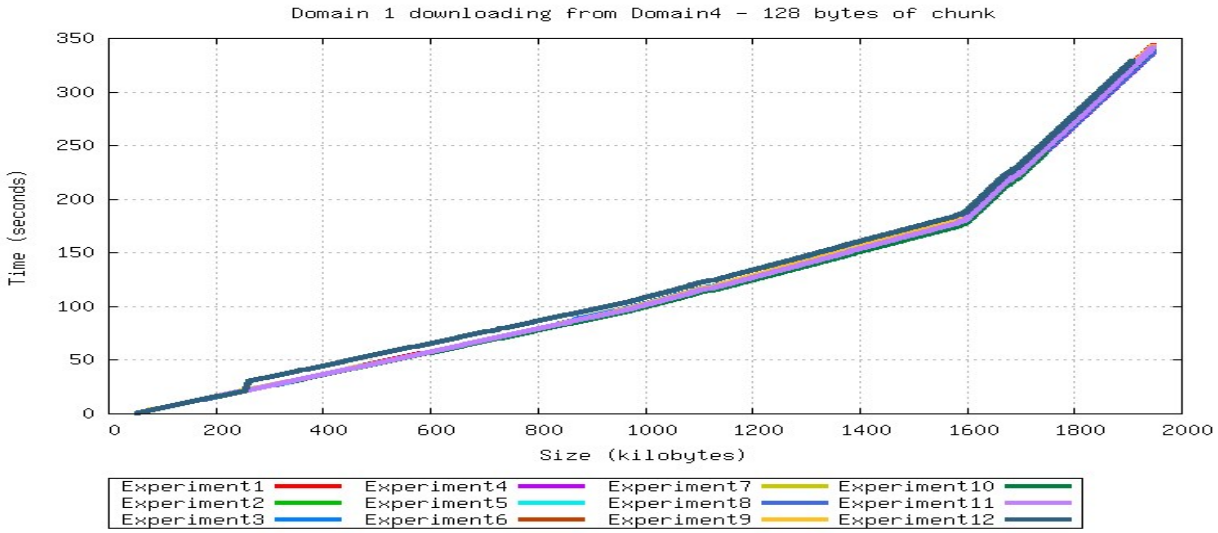
for 'Experiment6 50 bytes' the client is able to download chunks up to # 180/800, after that, all the retrievals timeout until the experiment is finally aborted, nevertheless the requests are arriving at the HTTP adapter server but this entity is unable to decode them due to state machine at bad state.
The former ratifies the problem of not offering a connection oriented service on the HIMALIS architecture placing the weight of transmission error resiliency on top of the application level shoulders. The prototypes employed for this experiments were not aimed and produced as final pieces of software, hence the small amount of error recovery. It is also important to remind that these experiments were executed without human intervention fulfilling one of the objectives introduced, experiment automation and repeatability.
Looking at the results from all the experiments, it is clear that it is feasible to transmit IoT video encapsulated with DASH on an IoT network employing a FI architecture as HIMALIS. Nevertheless, the results in terms of timing are not within desirable values since the amount of time expend for a 3 minute video is far above 5 minutes. Despite the bad results, it has to be taken into account the highly distributed nature of PLE plus the usage of tunnel based network overlay and the draft condition of the network adapters employed in the experiments. Adapter request pipelining could have offered higher rates at the cost making more difficult the debug task.
Conclusion is that there is no drawback that avoids from using HIMALIS over IoT architectures in general and in particular to transport video. It would be desirable having TCP over HIMALIS tunnels. That would have avoided the need of introducing the signaling used onto UDP which leads unavoidably, due to human errors and/or network errors, to racing conditions in which the software does not behave as expected, although this happens in very few cases as has been demonstrated.

4.3.3 CCN

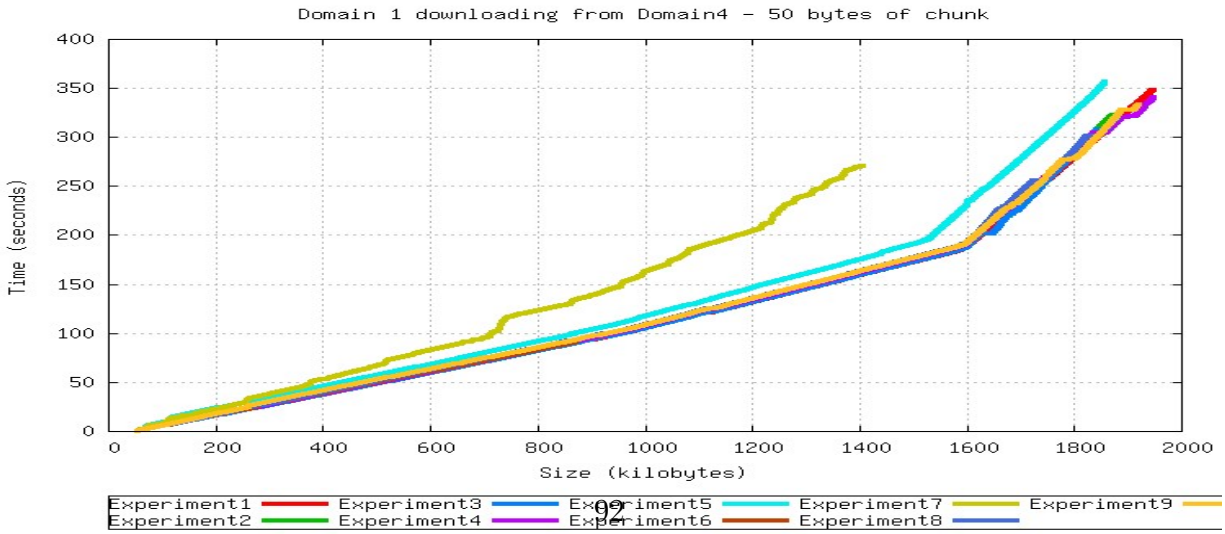
Also in the context of OpenLab and taking advantage of the experience gained with the evaluation of HIMALIS, the evaluation of another key concept for the FI, ICN was performed. Among the architectures available leveraging on the ICN for content distribution, the CCN is one of the more widely deployed and extended.
The objective of these series of experiments is the evaluation of a complete CCN architecture on top of a geographically distributed and widespread deployment thanks to PLE. Similarly to the work presented in Section 4.3.2, DASH is employed as the mechanism to encapsulate, split and stream videos. The use of DASH offers a simple deployment and distribution mechanism while maintaining desired features such as bit-rate



(a) Domain 1 to Domain 4 transmission with 512 bytes MTU



(b) Domain 1 to Domain 4 transmission with 128 bytes MTU



(c) Domain 1 to Domain 4 transmission with 50 bytes MTU

Figure 4.9: MTU variation study in HIMALIS separated domains

mtu	Install	Connect	Data Retrieval	Total Size (bytes)
Experiment1 512 bytes	639.4993	40.0778	333.2514	1993328
Experiment2 512 bytes	966.5911	40.1735	335.0967	1994247
Experiment3 512 bytes	607.4486	40.0381	336.6219	1993306
Experiment4 512 bytes	614.1696	40.4283	346.5841	1994247
Experiment1 128 bytes	612.8764	40.039	342.7470	1994247
Experiment2 128 bytes	625.9332	40.1679	150.1093	1422921
Experiment3 128 bytes	593.2233	40.0120	341.5366	1994247
Experiment4 128 bytes	626.5330	40.0550	336.3641	1994247
Experiment5 128 bytes	628.2406	40.0330	338.3113	1994247
Experiment6 128 bytes	606.4055	40.0480	65.7491	699630
Experiment7 128 bytes	613.0728	40.0200	337.7257	1994247
Experiment8 128 bytes	626.4426	40.0310	336.7822	1994247
Experiment9 128 bytes	612.9821	40.3020	342.2818	1994247
Experiment10 128 bytes	609.2002	40.0340	340.0761	1994247
Experiment11 128 bytes	613.1250	40.0350	340.9835	1994247
Experiment12 128 bytes	612.8836	39.9980	328.3043	1954349
Experiment1 50 bytes	620.248361	40.0400	348.4795	1994173
Experiment2 50 bytes	619.563842	40.0408	322.0301	1916082
Experiment3 50 bytes	629.838211	40.0650	313.1267	1920756
Experiment4 50 bytes	613.575704	40.1459	340.9295	1995474
Experiment5 50 bytes	624.284409	43.1638	356.0917	1902898
Experiment6 50 bytes	619.942278	40.0850	90.9474	892073
Experiment7 50 bytes	605.701077	39.8110	270.9041	1440640
Experiment8 50 bytes	627.097452	40.0719	321.1288	1927776
Experiment9 50 bytes	619.893201	40.0470	332.8402	1965636

Table 4.4: DASH over HIMALIS transmission in separated domains MTU variation study timetable.

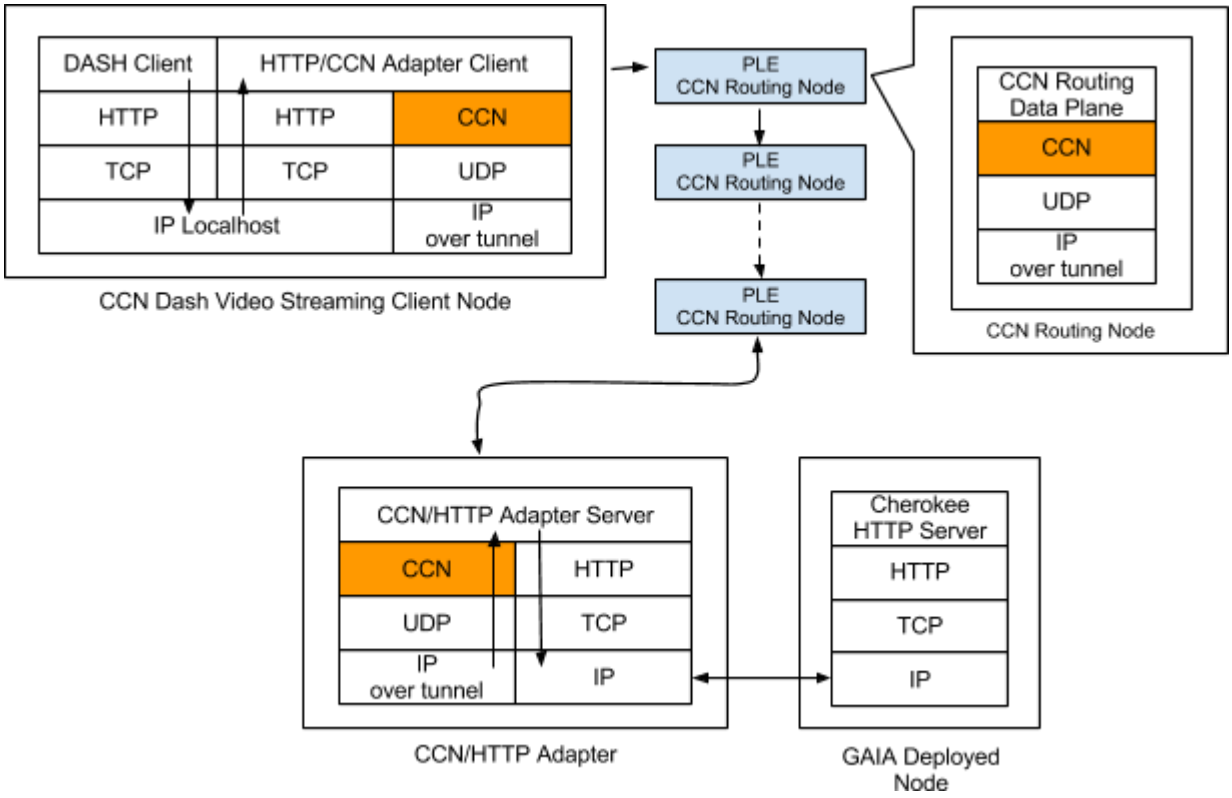


Figure 4.10: CCN deployment entities and relations

adaptation which is foreseeable as necessary in networks such as IoT with highly variable and constrained bandwidth characteristics. Again, the use of HTTP/CCN adapters is leveraged to provide with the means to adapt the current Internet TCP/HTTP approach to technologies of the FI.

After unfruitful experiments leveraging on ccnx [114] adapters, the client and server CCN adapters were implemented using python language and the library pyccn. This library is provided by UCLA, and offers Python bindings for the C version of with the the goal of speeding up the development by having an OOP style interface to CCNx but without the overhead of Java. Occupying only around 100 lines each of the adapters, they are capable of performing correct translation between CCN and HTTP and easily configure certain parameters of interest for the experiments, such as the CCN chunk size.

The adaptation between HTTP and CCN are provided by two adapters corresponding to each side of a typical HTTP connection, server side (CCN to HTTP adapter) and client side (HTTP to CCN adapter). Figure 4.10 shows how the DASH client is connected to an adapter ('HTTP/CCN Adapter CLient') listening to connection in localhost transforming the incoming HTTP requests to CCN interests and therefore converting the CCN data

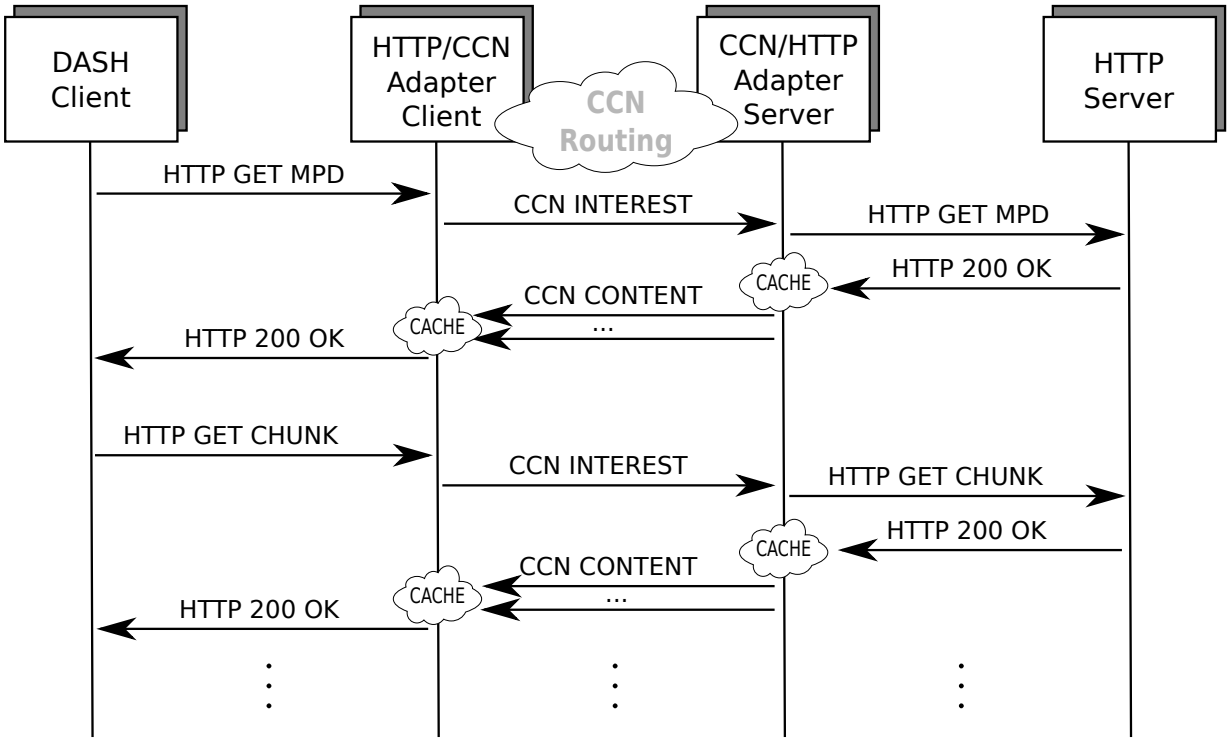


Figure 4.11: DASH/CCN network exchange sequence

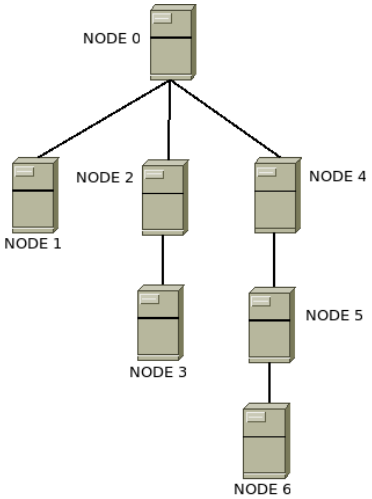
packets to HTTP responses. The generated interest packets are hence forwarded to the next node in the CCN topology which is labeled in the schema as 'CCN Routing Node' which in turn would redirect the packet to another 'CCN Routing Node', if needed, finally arriving to the 'CCN/HTTP Adapter Server' which reverses the process started by the 'HTTP/CCN Adapter Client' performing the HTTP request to the server and reshaping the HTTP response into CCN data packets, a more abstract view of the process is shown in Figure 4.11.

The CCN Signaling Plane is built on top of Tomcat running web based services. 'CCN routing node's Nodes are registered through the former Signaling plane. Also connection establishment. The web service generates an 'idl' configuration file that is provided to the Data Plane as well as the adapters. As part of this experimentation a library to generate native CCN packets was produced.

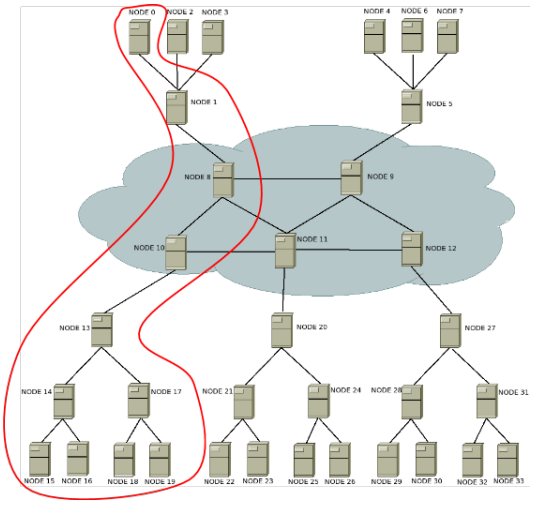
4.3.3.1 CCN Scenario Deployed on PLE

This section describes the topologies deployed in PLE to perform the experiments. There are two topologies, the topology shown in Figure 4.12a that has independent paths with different number of hops from the leaf nodes to the root node and the topology in Figure

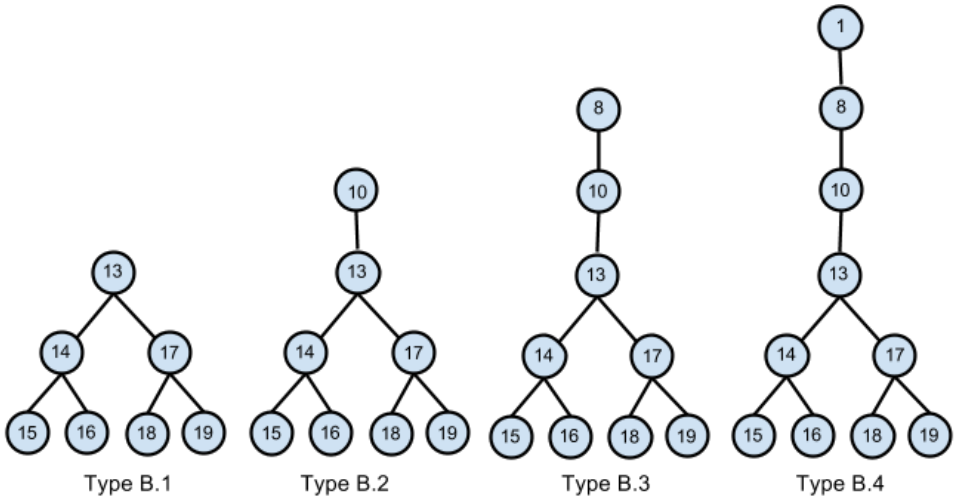
4.12b that offers different number of hops were the nodes in the path coincide as shown schematically in Figure 4.12c. These topologies allow the study of different parameters such as the number of jumps and their effect in the video retrieval time, the effect of chunk size, the network latency and or the packet loss ratio.



(a) CCN simple scenario



(b) CCN full deployed scenario with paths employed for experimentation



(c) CCN full scenario schematic video of the different selected paths.

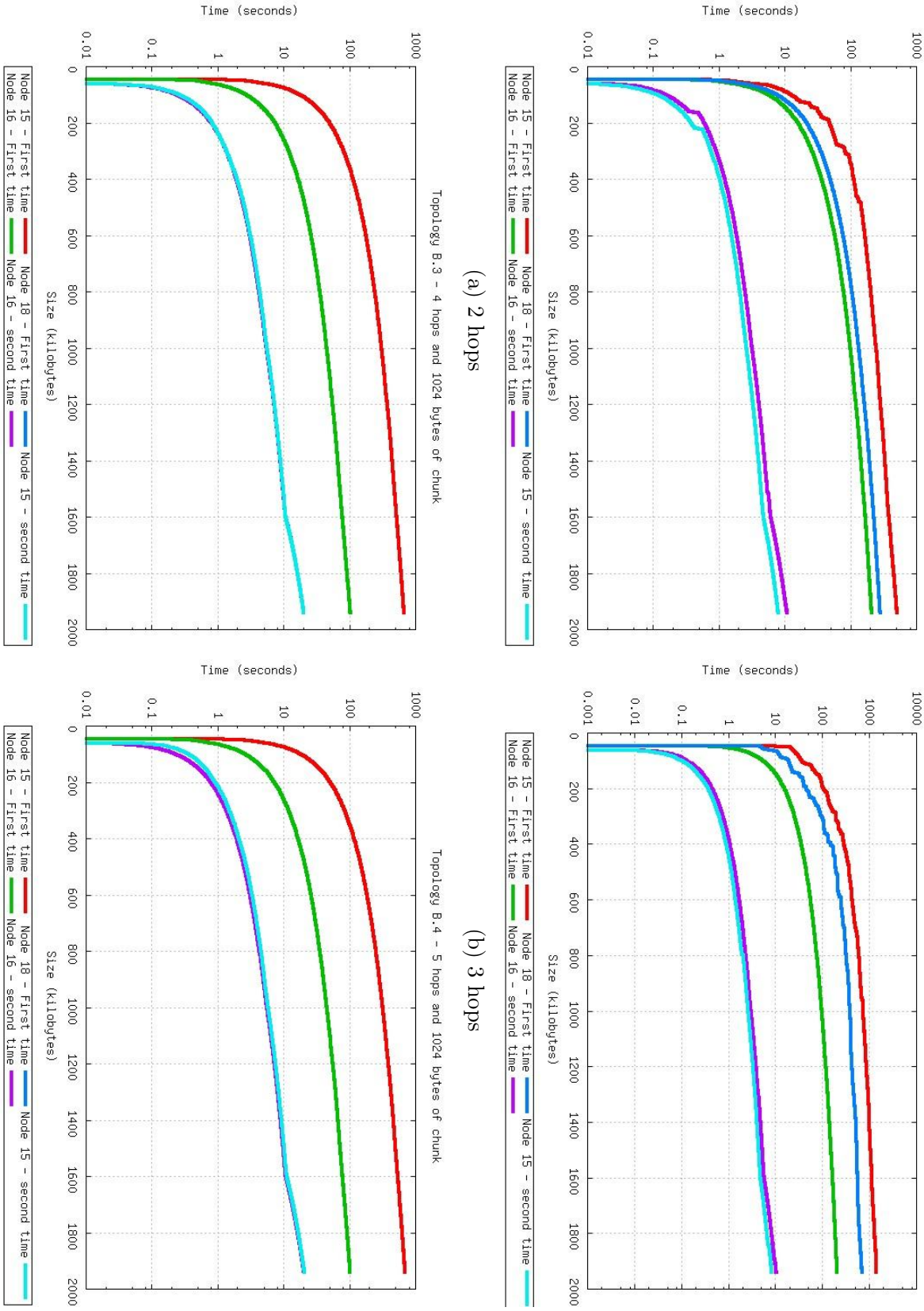
The topology described in Figure 4.12b includes a central backbone of five interconnected nodes that are necessarily traversed by interests sent by clients (leaf nodes at the bottom of the figure) trying to get to the content providers (root nodes on the top of the figure). Each line leaving the cloud represents a whole subdomain in CCN terminology with different subnetwork, hence routing is performed through the mechanisms provided by the CCN protocol.

4.3.3.2 DASH streaming - base case

Evaluation of DASH on top of CCN is performed by employing the topology in which the nodes in the path coincide (Figure 4.12b). The experiments are performed by deploying the 'HTTP/CCN Adapter Server' one node further each iteration so that the obtained network deployment is that of Figure 4.12c Type B.1, Type B.2, so on and so forth. Obtaining as a result the behavior of CCN from 2 hops up to 5 hops. Remaining parameters of the experiments are statically configured not affecting the evaluation. As a side note, the MTU size for these series of experiments is of 1024, a large enough value to be considered Internet standard and small enough so that no underlying network (such as the vtun used for PLE deployment) affect the measurements.
For the topology described in 4.12c Type B.1 described as the 2 hops case, the clients involved will be nodes numbered as 15, 16 and 18 retrieving successively the video in that precise order. Finally, clients 15 and 16 will repeat the retrieval to observe in-network caching effect.
Figure 4.13a shows the results of the streaming process. Node 15 and 18 have the highest values since they need 2 jumps to reach the content. Node 16 enjoys a significant reduction in download time, the deployment situated the node nearer from the common node originating that the RTT is significantly lower. Node 15 second request results resemble those of Node 16 second request where in-network caching is applied and content is served by common ancestor in Node 14. Similarly to the results of HIMALIS in section 4.3.2, it can be observed that the last 400 data bytes produce a proportionally higher increase in the download time. As introduced before, the cause for that increase is the growth in number of chunks with the reduction of size per chunk as can be seen in Figure 4.1.
Results for 3 hops are presented in Figure 4.13b. There is a clear similarity to the results for 2 hops but a small tendency to Node 18 results to be nearer to those of Node 15 can be envisioned, as well as the unification of times for the second round of downloads. Those two tendencies are confirmed by the results shown in Figures 4.13c and 4.13d corresponding to 4 hops and 5 hops respectively.

4.3.3.3 Reducing Maximum Transmission Unit

Similarly to the study made for HIMALIS where the MTU was reduced progressively to analyze the influence of that parameter in the transmission of DASH video on top of HIMALIS, same experimentation was performed for CCN. In this case, the scenario used for the experimentation is the one presented in Figure 4.12a. In this case the topology is fixed and different hops to a common node focusing the experiment variability in to the



(a) 2 hops

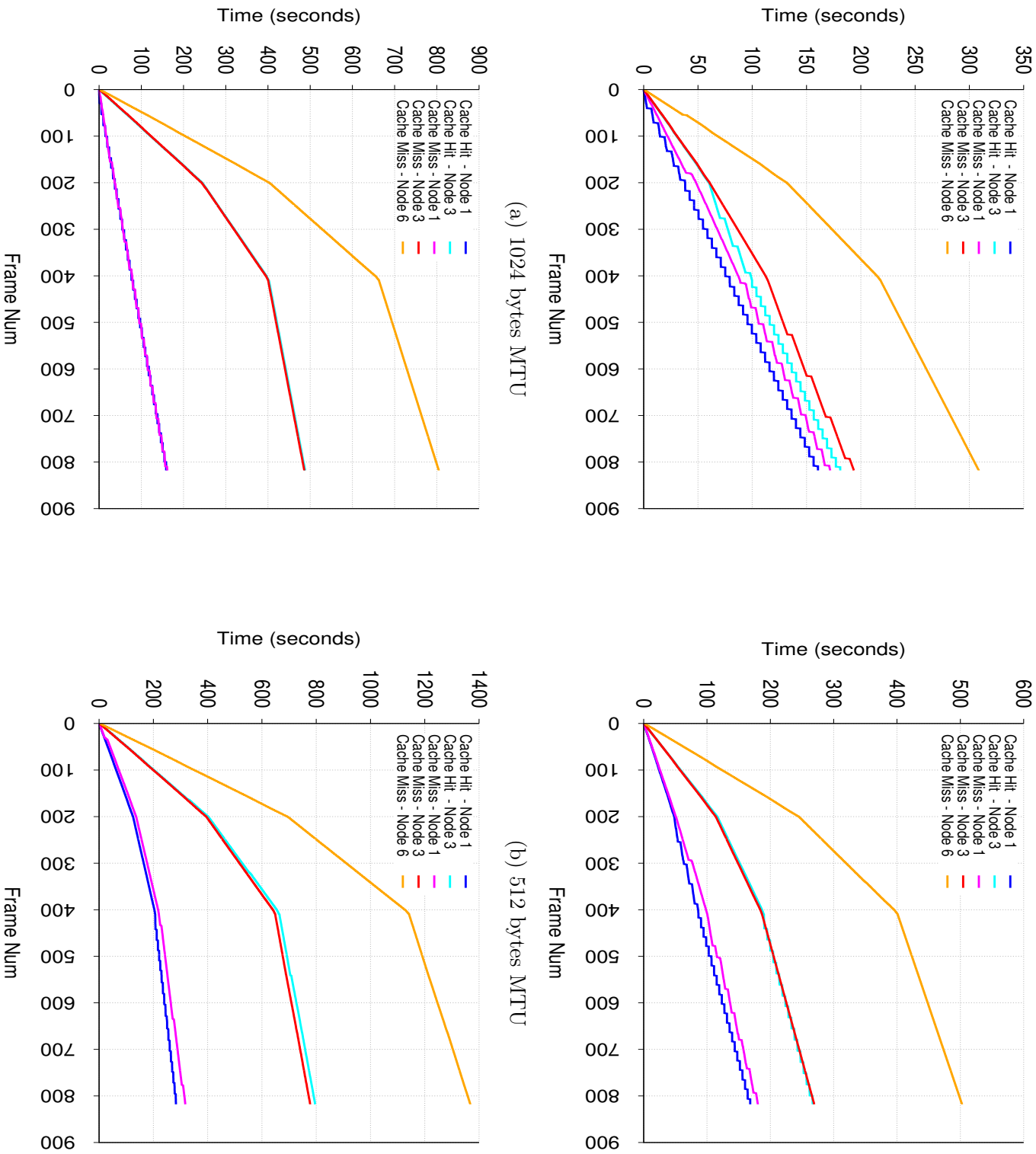
(b) 3 hops

(c) 4 hops

(d) 5 hops

Figure 4.13: CCN transmission results with 1024 bytes MTU

MTU size.
Thanks to the designed topology the experiment for each MTU value is performed by retrieving the content from nodes 1, 3 and 6 respectively with empty in-network cache that produces results for 1, 2 and 3 hops respectively, finally repeating the retrieval from nodes 3 and 6 to show the in-network caching (populated in the first round) effect.
The chunk sizes for these series of experiments go from 1024, complementing the results from previous subsection, to 32 bytes, reaching the boundaries of IoT transmission standards (e.g. 81 bytes in the case of 6LOWPAN [108]). Within the boundaries of what is considered IoT, 64, 48 and 32 bytes were experimented, going far beyond the theoretical limits of what CCN supports causing special adaptation during the experiments.
Figures 4.14a to 4.16 represents the time progress of clients at Nodes 1, 3 and 6 after two consecutive requests for MTU sizes varying from 512 bytes to 32 bytes.
During the first part of the experiment, the nodes 1, 3, and 6 (as shown in Figure 4.12a) will sequentially request the video stream for the first time, what has been denominated 'Cache Miss'. Then, the nodes 3 and 1 will request the video stream again retrieving the cached version, what has been denominated as 'Cache Hit'. This way, it can be observed and measure how video delivery is affected by CCN caching.
On a first round of experiments the performance of CCN for IoT HTTP without any timing restriction applied, meaning that the whole stream is downloaded without taking into account any play time like a bulk download. Tables 4.5, 4.6 and 4.7 list the results for the bulk download of the video sources encoded at 38, 58 and 79 kilobit per second (kbps) respectively. In general cache hit values are lower that their cache miss counterparts, nevertheless the differences are not sufficient to determine that the CCN in-network caching imply a direct benefit for the client not taking into account the saving in terms of bandwidth for the network operator. From tables it is clear that the reduction of the CCN MTU size implies a loss in performance increasing the download time exponentially. There is also a direct relation between the number of CCN hops and the retrieval time. In general, for any bit-rate and/or MTU size can be observed that the difference between the cache miss and the cache hit time for a certain node is smaller in terms of the total download time.
In Figures 4.15, 4.14 and 4.16, the relation between the elapsed time each client node spent to download the 58 kbps video version corresponding to Table 4.6 is shown.
As already stated, with the figures can be visually confirmed that the increase in download time is proportional to the decrease in MTU.
For each case it is also possible to confirm the relation between the number of hops between the client and the 'CCN/HTTP Adapter Server' and the increase in the retrieval time.
All the graphs present a variation in the angle to a more horizontal one from frame 410



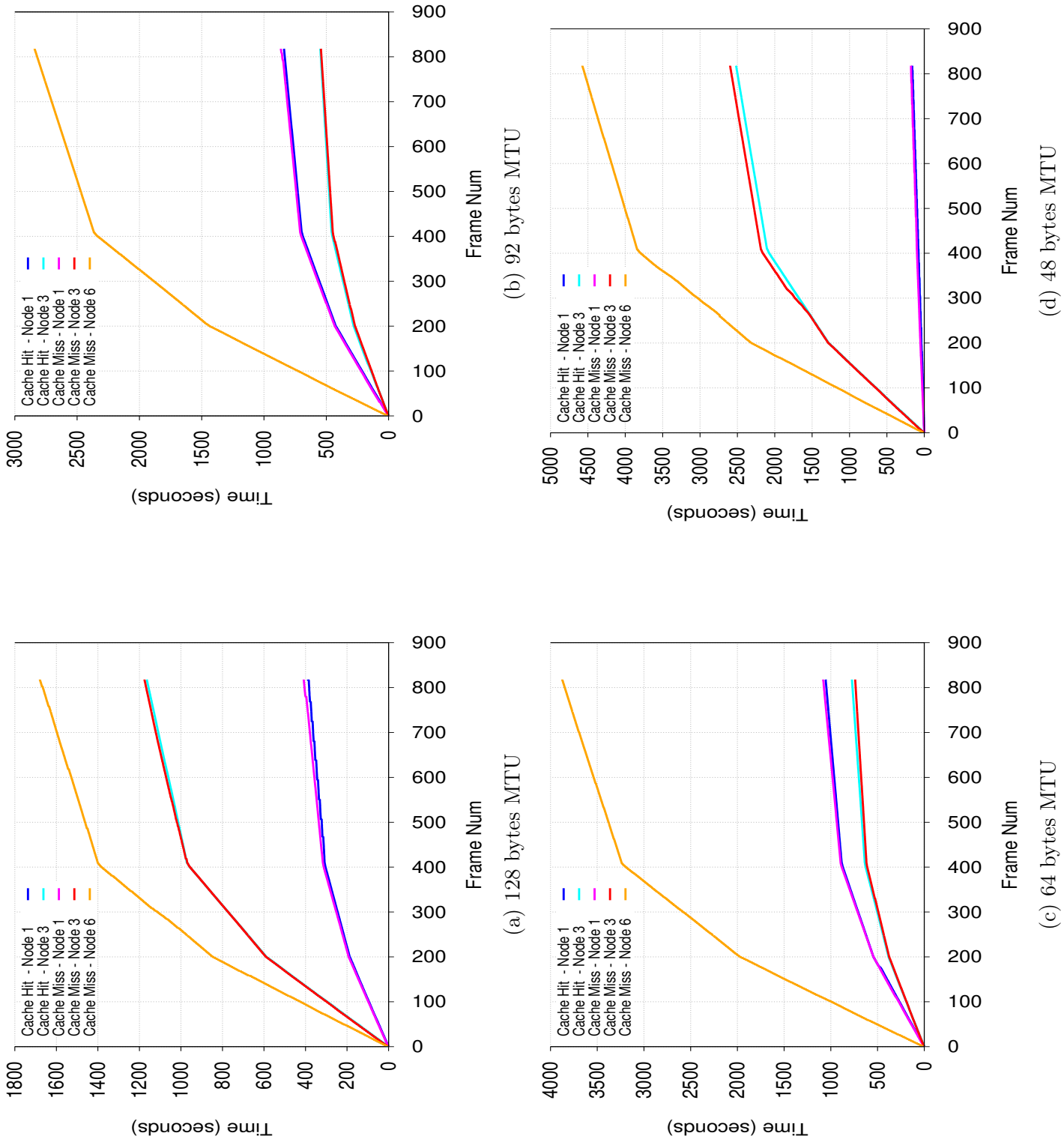
(a) 1024 bytes MTU

(b) 512 bytes MTU

(c) 256 bytes MTU

(d) 192 bytes MTU

Figure 4.14: Results for basic topology (I)



(a) 128 bytes MTU (b) 92 bytes MTU (c) 64 bytes MTU (d) 48 bytes MTU

Figure 4.15: Results for basic topology (II)

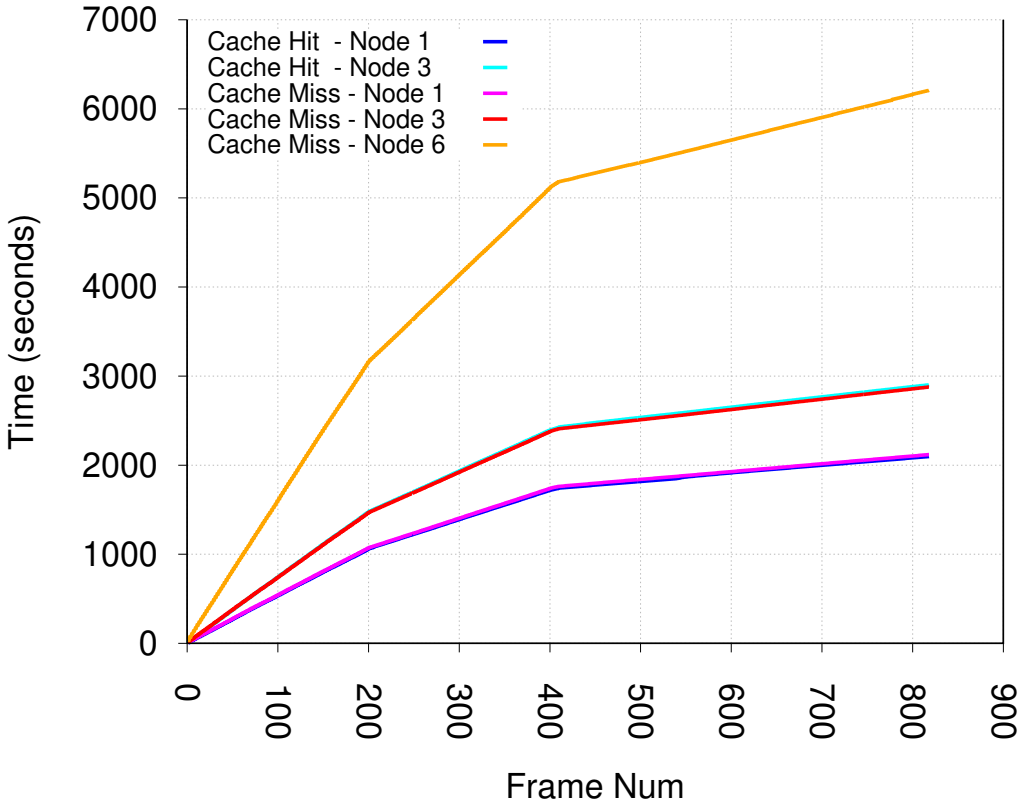


Figure 4.16: 32 bytes MTU

Bytes	N1Miss	N1Hit	N3Miss	N3Hit	N6Miss
32	1686,0579	1665,3442	3407,6152	3392,2056	3676,5575
48	668,7148	658,0254	1547,9900	1555,4722	3268,7163
64	805,0028	752,0781	1325,8520	1421,4504	2530,6700
92	379,1102	359,1160	1092,7789	1099,6510	1723,3172
128	306,9185	278,9557	734,1414	798,5298	1137,7130
192	257,4231	232,9028	297,1269	324,3383	758,9249
256	276,7760	238,0714	372,0721	385,3181	573,7049
512	203,9962	168,3308	214,3915	228,7382	292,8262
1024	167,3246	162,0324	165,8882	162,3756	205,8480

Table 4.5: Experimentation with 38 kbps bitrate

Bytes	N1Miss	N1Hit	N3Miss	N3Hit	N6Miss
32	2118,1373	2097,1697	2877,0511	2902,7176	6209,2617
48	1628,9004	1594,7934	1031,5177	1029,8626	4323,3920
64	1081,4020	1055,4288	739,2264	774,2104	3873,9569
92	864,1175	839,6645	543,2547	547,4849	2841,0136
128	689,2486	619,7700	579,5825	661,6352	1699,3918
192	451,7747	381,5972	666,6065	703,0290	1293,6931
256	361,5066	309,5498	486,8731	480,0760	989,8218
512	247,8281	210,3649	204,0162	232,4632	459,0574
1024	160,1961	160,1581	214,0908	209,5092	329,0974

Table 4.6: Experimentation with 58 kbps bitrate

Bytes	N1Miss	N1Hit	N3Miss	N3Hit	N6Miss
32	3319,0994	2976,7420	4836,4743	FAILED	9085,2073
48	1310,2709	1265,1468	1905,6432	1946,8194	6186,4586
64	1645,4683	1636,9633	1339,5937	1614,4556	4829,9400
92	1141,9833	118.8758	7120.2064	729.1005	3085.2014
128	181,3390	159,6204	1607,7552	1590,2523	2898,6299
192	563,3972	491,3653	787,9775	845,9573	1540,1011
256	418,9757	381,1457	526,5776	529,0240	1392,3111
512	296,6672	256,7014	242,0550	278,1475	685,3320
1024	166,8538	159,7497	289,3108	277,0339	427,8385

Table 4.7: Experimentation with 79 kbps bitrate

onwards corresponding to the reduction in the DASH chunk size. At frame 410 there is a decrease on the frame size originated by the second part of the video 'Bridge far'. In general, the first part of video 'Bridge' has more moving elements still visible after the high submitted compression (waves, pedestrians, ...) meanwhile in the second part those objects are in most cases visually discarded by compression therefore reducing the amount of data to be encoded hence limiting the frame size. The performance of each node is independent of the others and has to be seen as an independent even originated by the randomness of PLE. It has to be taken into account that each figure is a full new deployment were all the nodes get reallocated depending on the PLE status.
The former values served the purpose of studying how the CCN network would behave with a small MTU but without taking into account the application on top, video. It is true that the data source was IoT video and that influenced on the file type produced but it was not taken into account how the download would affect a video streaming client. That last variable is taken into account in the next series of experiments.

4.3.3.4 PSNR study

In order to take into account the lifespan of a frame into a video streaming application, the client was configured with different timeouts per chunk. The timeout trigger make the video chunk ignored for playing therefore stopping the download for that precise chunk. For this scenario the number of hops parameter is ignored since the focus is on the network packet size restriction in addition to the timeout. Only Node 1 of the topology was employed for these series of experiments.
As a consequence of the timeout per chunk, some of the chunks are not downloaded. When a chunk is discarded, the player replaces it with the previously downloaded one (as may video decoders do). Duplicating last frame has two advantages, in terms of QoE since the user observes a still image which is (unless there was a scene change) more natural than a full color frame and in terms of this experiment allowing to make a frame by frame comparison of the PSNR values providing with and objective assessment of the perceived quality. Picture 4.17 shows an example of the difference 4.17c between a frame (#603) replaced with the previous one 4.17a and a frame successfully downloaded 4.17b although to human eye the differences are imperceptible.
The first iteration of the study was performed for the lowest bit-rate (38 kbps) with 1, 3 and 5 seconds timeout. The results showed that 1 second timeout was inadequate for the scenario since most of the chunks were not downloaded. One of the reasons for this result (which was partly expected) is the average RTT of PLE which is measured around 300ms.

(a) Cache Miss frame 603. (b) Cache Hit frame 603. (c) Differences frame 603.

Figure 4.17: Visual result frame 603 with a bitrate of 79kbps.

Tables 4.8 and 4.9 represent the average PSNR values obtained in Cache Miss and Cache Hit cases for node 1 with each network packet size restriction retrieving the 38 kbps video version as well as the number of chunks retrieved. In order to make easier the analysis of the results, two extra columns for the δ values in terms of average PSNR value and number of retrieved chunks were appended. Negative δ PSNR values indicate an increase in the video quality which means that the result was nearer to that of the original video, while an increase of the δ number of chunks implies an increase of the number of downloaded chunks for the cache hit case. Looking at the tables can be clearly noted that the use of cached data has a positive influence on the perceived quality since the δ PSNR values are always lower than the cache miss scenario.

Table 4.8: PSNR 3 seconds timeout study with 38 kbps bitrate

	Cache Miss	# chunks	Cache Hit	# chunks	δ PSNR	δ chunks
0092	28,8025	405	28,8231	410	-0,0206	5
0128	28,8012	537	28,8195	609	-0,0183	72
0192	28,8088	467	28,8164	608	-0,0076	141
0256	28,8037	504	28,8231	613	-0,0194	109
0512	28,8029	612	28,8541	695	-0,0512	83
1024	28,8342	684	28,9044	814	-0,0702	130

Both tables (4.8 and 4.9) indicate that there is a direct relation between the MTU and the timeout. To obtain a successful service when the former is decreased the later must be increased which implies a smaller QoE since a bigger buffer time is required. On Table 4.9, rows corresponding to 48 and 64 bytes MTU show a reduction on the number of successfully downloaded chunks while the average PSNR stays at a higher value for the

Table 4.9: PSNR 5 seconds timeout study with 38 kbps bitrate

	Cache Miss	# chunks	Cache Hit	# chunks	δ PSNR	δ chunks
0048	21.1791	283	28.5491	277	-7.3700	-6
0064	28,8081	407	28,8295	404	-0,0214	-3
0092	28,8001	407	28,8342	410	-0,0341	3
0128	28,8093	409	28,8342	410	-0,0249	1
0192	28,8050	472	28,8231	613	-0,0181	141
0256	28,7987	600	28,8229	618	-0,0242	18
0512	28,8057	468	28,8231	612	-0,0174	144
1024	28,8559	695	28,9062	818	-0,0503	123

cache hit scenario in comparison with the cache miss. For the 48 bytes case, the problem was the inability of the client to download the first 7 frames producing a drift on the PSNR calculation (the total number of frames was lower than that of the original video). For the 64 bytes case the chunks lost were of the second part of the video which as was already commented has smaller frame size and each chunk carries less information, nevertheless losing more chunks on this last part of the video is producing a better PSNR value but that doesn't imply an increase on the subjective QoE perception.

Although the results are promising, it has to be noted that some MTU values are missing for each table and in particular 32 bytes experiments were completely unsuccessful. As a consequence of these results and to entrust the assertion that the combination of 32 bytes MTU and a timeout restriction below 5 seconds was not capable of transmitting the desired bit-stream, a series of experiments with higher timeout values for the 32 bytes MTU case were performed choosing increases of 2.5 seconds in the timeout as tentative values therefore experimenting with 7.5, 10 and 12,5 seconds. From these series of experiments a result of a minimum of 10 seconds timeout per chunk was needed to successfully stream 32 bytes reduced MTU streams. The obtained PSNR for the cache miss case is 28.8329 db and the corresponding cache hit value is of 28.8085 db. The values can be considered good unless the final retrieval time is taken into account. The download time for each case is of 1046.87 seconds and 1043.31 seconds respectively. In comparison the experiments with 92 bytes MTU and 3 seconds timeout achieve values around 400 seconds which already was double the time of the source video duration.

Taking into account the obtained results, two extra experimentation rounds were performed but with increased bit-rate (58 and 79 kbps) not exceeding the bandwidth of a typical IoT scenario also restricting the MTU to that of these scenarios between 48 and 128 bytes. An

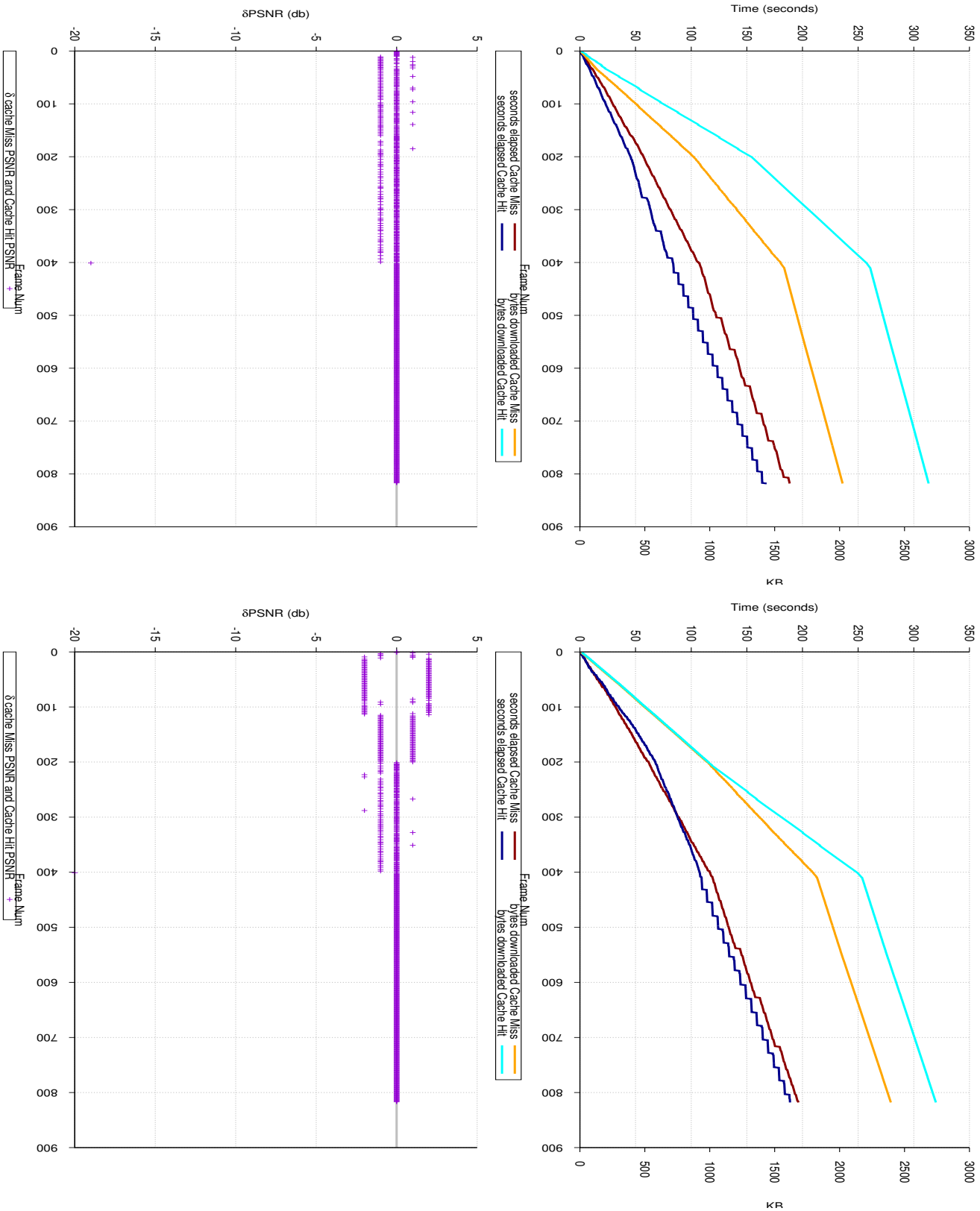
Table 4.10: PSNR 5 seconds timeout study with 58 kbps bitrate

	Cache Miss	# chunks	Cache Hit	# chunks	δ PSNR	δ chunks
0048	30,0619	691	30,1837	803	-0,1218	112
0064	29,8381	359	29,8830	348	-0,0449	-11
0064	30,1323	742	30,1973	817	-0,0650	75
0064	30,1353	747	30,1981	818	-0,0628	71
0092	30,0070	400	29,9999	385	0,0071	-15
0092	30,0134	408	30,0328	406	-0,0194	-2
0092	30,0099	405	30,0325	405	-0,0226	0
0092	30,1741	792	30,1981	818	-0,0240	26
0128	30,1910	810	30,1981	818	-0,0071	8
0128	29,9986	428	30,0198	574	-0,0212	146
0128	30,0100	406	29,9090	359	0,1010	-47
0128	30,0087	473	30,0216	613	-0,0129	140

increase on the bit-rate shifts the weight on the network with the side effects of an increase of the QoE and/or a reduction of the coding complexity, allowing the codec to produce higher bit-rates while maintaining the video quality allows the possibility to remove costly procedures in the encoding in case that the encoding is performed by software. The PSNR study for these two rounds of experiments is shown in Tables 4.10 and 4.11 and some visual results are also shown in Figures 4.18 and 4.19 corresponding to 48 and 64 bytes MTU respectively. Each figure shows two plots, the one on the top showing the relation between the downloaded size (not all the chunks are downloaded therefore the size varies) and the time consumed and the bottom showing the δ PSNR in which the represented points below the x-axis 0 represent visual enhancements for the cache hit case while points above the axis imply a reduction in the perceived quality. All the presented values show an increase in the perceived quality while there is a reduction in the expended time for the cache hit (as expected) thus demonstrating that in-network caching is a desirable capability for the FI networks.

Despite the randomness of the PLE node selection of the experiments, the results provide with a good approach of what could be achieved by encapsulating IoT video in DASH and distributing it with an ICN approach leveraging on CCN while the transmission channel remains within IoT parameters.

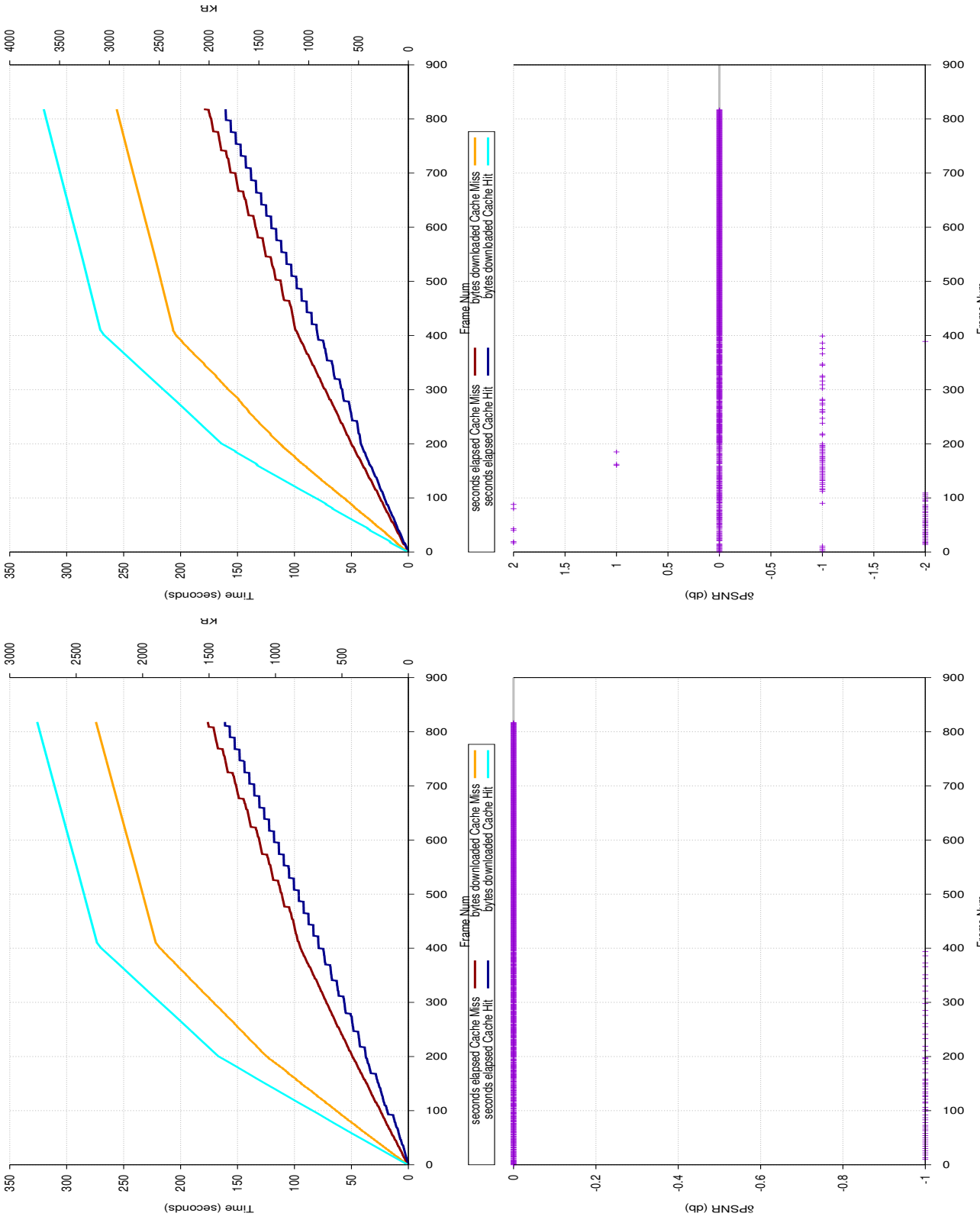
These series of experiments have successfully demonstrated that IoT video can be easily encapsulated into DASH streams providing with easier deployment without requiring any



(a) 48 bytes, 5 seconds timeout, 58 kbps

(b) 48 bytes, 48 seconds timeout, 79 kbps

Figure 4.18: Results of CCN with limited packet size and timeout 48 bytes MTU.



(a) 64 bytes, 5 seconds timeout, 58 kbps

(b) 64 bytes, 5 seconds timeout, 79 kbps

Figure 4.19: Results of CCN with limited packet size and timeout 64 bytes MTU.

Table 4.11: PSNR 5 seconds timeout study with 79 kbps bitrate

	Cache Miss	# chunks	Cache Hit	# chunks	δ PSNR	δ chunks
0048	30,9516	654	31,0540	713	-0,1024	59
0048	30,5292	305	30,7512	330	-0,2220	25
0048	30,9659	667	31,0742	730	-0,1083	63
0064	30,4641	300	30,4765	302	-0,0124	2
0064	31,0744	723	31,1878	809	-0,1134	86
0064	30,4198	293	30,4334	293	-0,0136	0
0092	30,5083	306	30,9169	393	-0,4086	87
0092	30,3773	286	30,3529	280	0,0244	-6
0092	30,4487	300	30,4516	296	-0,0029	-4
0128	30,5010	324	30,9535	410	-0,4525	86
0128	30,9153	458	30,9192	611	-0,0039	153
0128	30,9234	407	30,9535	410	-0,0301	3

unavailable technology and with the extra benefit of taking advantage of all the caching mechanisms developed for the WWW during the last years. In addition, the experiments have demonstrated that CCN is capable of transmitting content under different network packet sizes restrictions. This capability is in particular really interesting to IoT due its usual restrictions. The constraints introduced by PLE suppose an extreme case and deploying this system on a real IoT network, could only lead to better performance due to the reduction of RTT. It is clear that the junction of IoT, DASH and CCN is, in consequence, a good first approach for video on *things* in the environment of Future Internet.

4.3.4 CCN on top of HIMALIS

Taking advantage of the aforementioned experiments for both FI architectures, the natural next experiment to be carried out was to collocate CCN on top of HIMALIS. The HIMALIS identifier based connectivity system substitutes the CCN routing system while the CCN layer avoids the necessity of employing such identifiers in the application layer relying on Uniform Resource Identifiers (URIs) for referencing to the content as is usually done in ICN architectures. Figure 4.20 shows a diagram of how the combination works and the messages exchanged by the various entities in the system.

The integration of CCN with HIMALIS produces a very desirable outcome for the former. The ICN paradigm benefits from the HIMALIS's multi-domain dimension, federating two

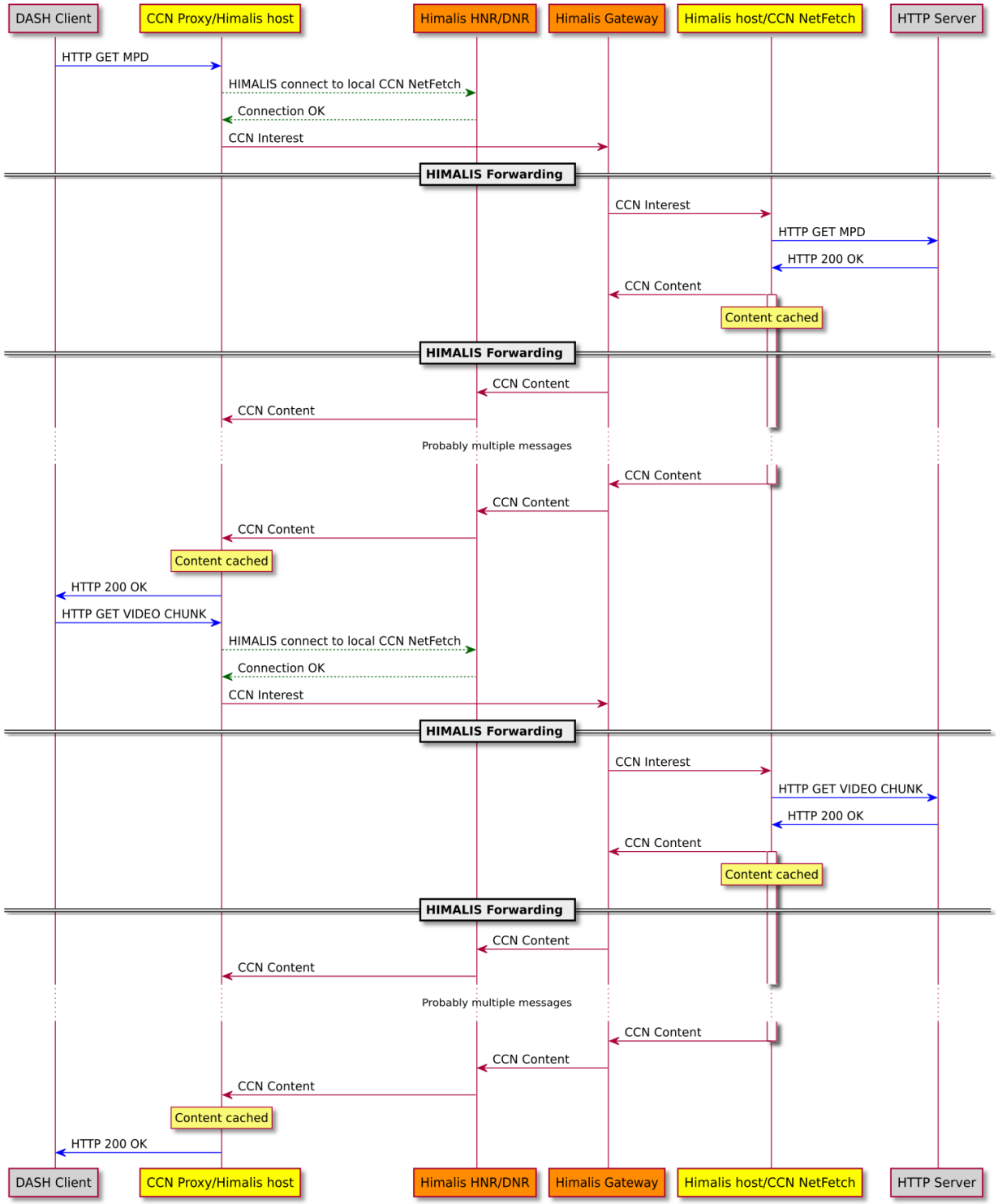


Figure 4.20: CCN over HIMALIS flow diagram

(or more) HIMALIS domain which on top run the CCN architecture for content distribution and in particular for video streaming over HTTP. The integration produces implications in terms of authorization, security and network performance since now there will be, at least, two ICN instances sharing content and, at least, two networking domains allowing connectivity between each other. One of the more interesting issues pointed out in the CCN bibliography is the control over the in-network cached content by the content provider and what is more important how to charge the rightful users.

Figure 4.21 shows the retrieval of the Media Presentation Description (MPD) file in a federated CCN over HIMALIS scenario. The term Local makes reference to the client's administrative domain and Remote to the provider's domain. Obviously, these terms do not imply geographical distribution, therefore a request to the Local 'CCN NetFetch' might imply traversing multiple domains. It is not intended with this figure to describe precisely all the messages neither all the combinations.HIMALIS itself would employ more messages both in the signaling plane and the data plane just to rely a simple packet. Also the DNR and HNR entities which have been deployed collocated on the same host could be split in multiple hosts leading to extra signaling over the network which now is carried on within localhost.

To get an idea of how such integrated architecture would behave and taking profit of the NEPI scenario descriptions of previous experimentation, some experiments that simulate the packet exchange have been performed. Although the executions of CCN and HIMALIS are isoolated from each other, the results are analyzed together. The experiment therefore consists on a DASH client retrieving the content from an HTTP server, both entities being collocated on different HIMALIS nodes. Figure 4.22 show the experiment results, including the data from a 'raw' HIMALIS transmission (green line), the CCN cache miss (blue line) and hit (cyan line) values and finally the integration of CCN on top of HIMALIS with cache miss (red line) and hit (brown line).

It must be noted that Figure 4.22 shows the time employed by HIMALIS to 'warm-up' the infrastructure which is around 40 seconds, therefore all HIMALIS based data do not start at the axis. For the CCN over HIMALIS case, the connection times are added to the initial time, this extra overload would probably be reduced by the fact that all the CCN routing and topology could be flattened since now from the CCN point of view all the destinations would be one jump away since it is HIMALIS in charge of the underlying connectivity. On the other hand, it would be desirable and easier to centralize the cache structure to obtain as many cache hits as possible.

Finally, it is worth to mention that the benefits given by the coalition between HIMALIS and CCN regarding content delivery are mainly related to the hybridization of the network.

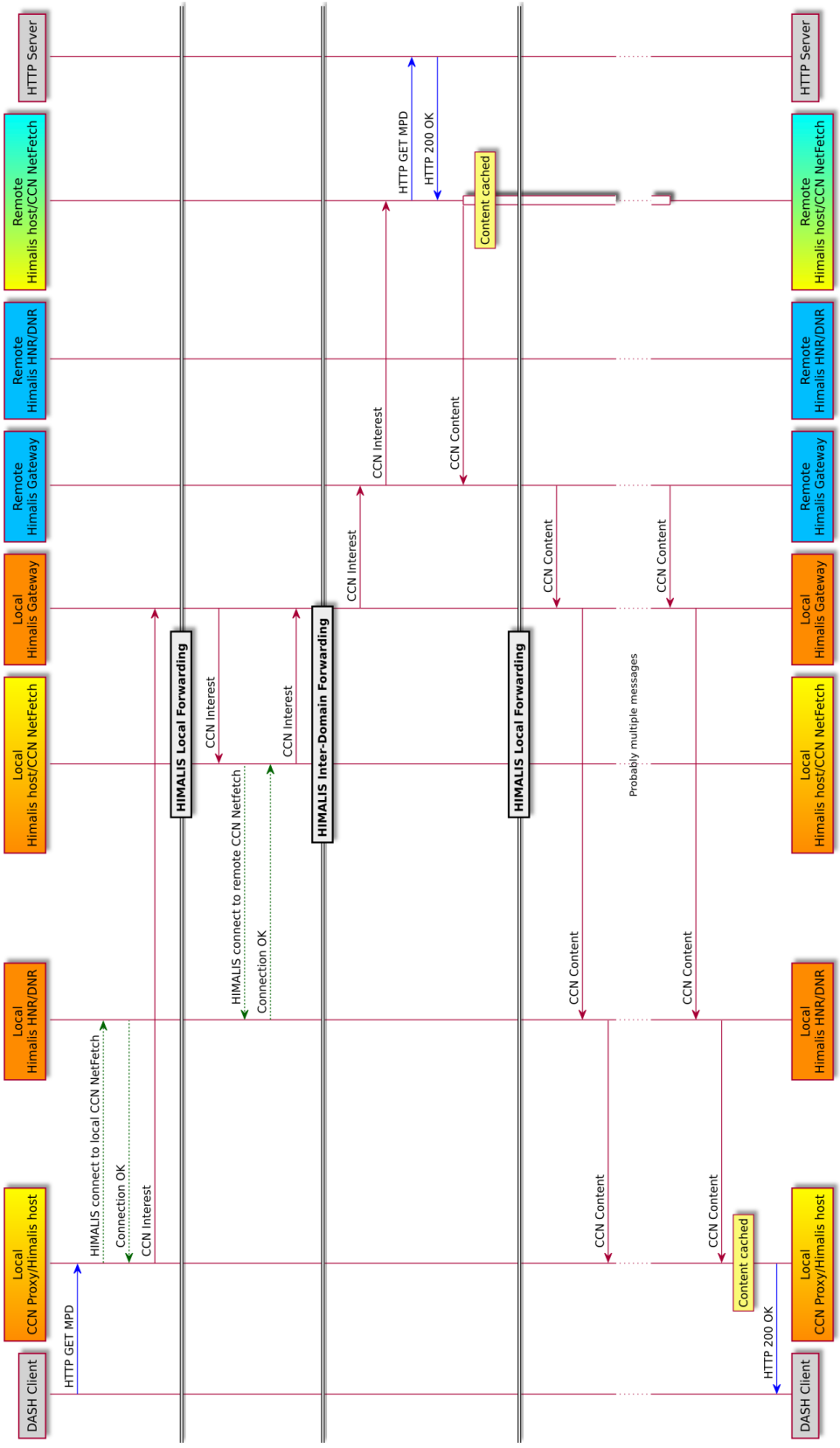


Figure 4.21: CCN over HIMALIS flow diagram, domain federation specific

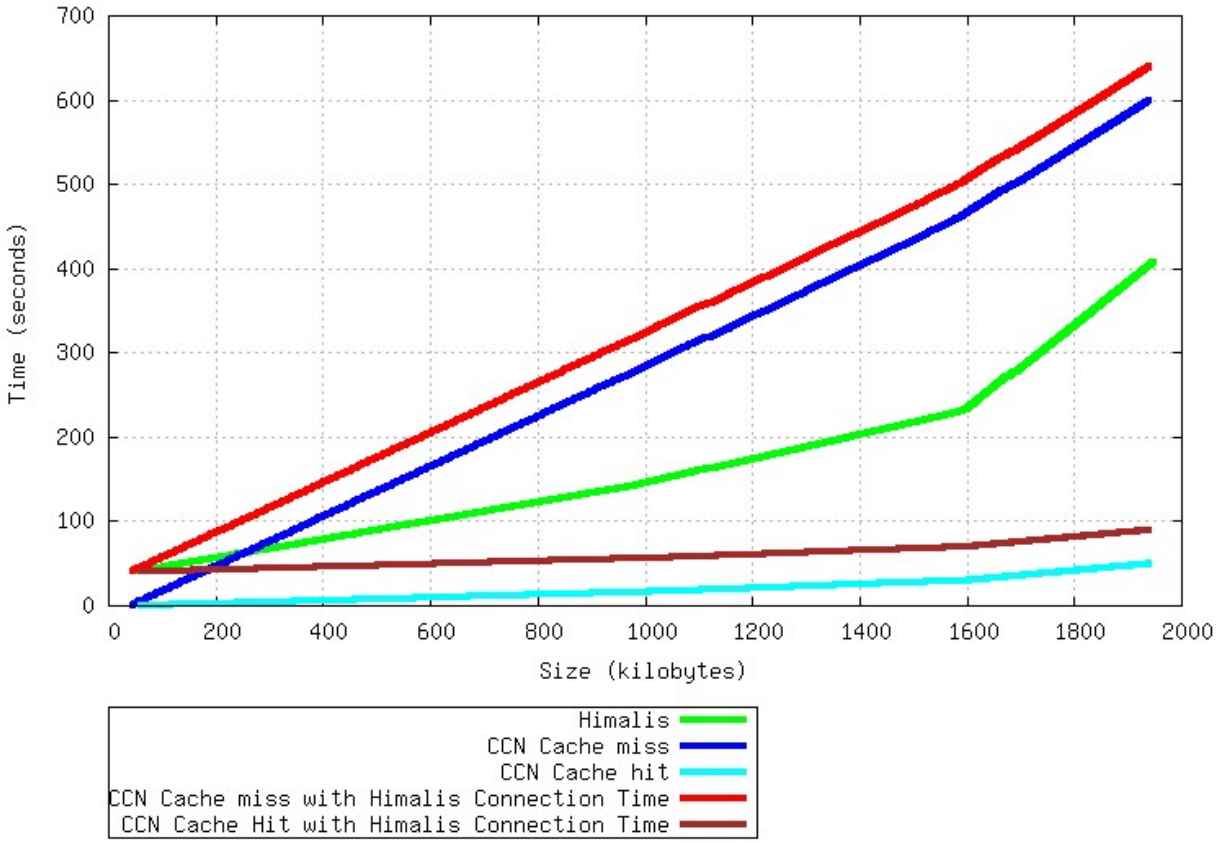


Figure 4.22: CCN over HIMALIS federation results.

While CCN works very well when transferring and caching content, it does not work so well with end-to-end communications. This is where HIMALIS comes into play. Also, HIMALIS provides a good underlying infrastructure to decouple CCN from IP and thus obtaining increased scalability in the network offering a clean-slate approach in terms of geo-localization of the network addresses breaking with the so-called ossification. Moreover, CCN traffic may be directly translated to HIMALIS traffic in the intermediate gateways, enabling the existence of CCN-only clients or servers. This is feasible because HIMALIS separates identifiers from locators, so content may be transmitted by using its name as HIMALIS identifier without worrying about the underlying locators and the corresponding transmission operations.

4.4 Conclusions

This chapter has explored various FI architectures and analyzed their effect on video streaming by deploying them on top of worldwide distributed testbeds.
From the point of view of the architectures being inspected, HIMALIS is a more connection oriented approach while CCN relies on a ICN design and leaves underlying connectivity to other architectures (legacy or even HIMALIS as has been shown in this chapter). Although it is true that HIMALIS seems to be more versatile, it is also true that the difficulties in its deployment are quite higher than those of CCN, nevertheless CCN introduces other complexities related to the in-network caching, so choosing one or another or a combination of both depends pretty much on the application layer.
Regarding the application layer, the transmission of video using the so well known DASH protocol over the two aforementioned technologies has been carried out and demonstrated, in addition, the inclusion of IoT capable encoded videos in the experimentation introduced a third FI technology in the experimentation results.
The MTU reduction affected differently to each architecture. The bigger impact on CCN shown in the obtained results can be related to the fact that HIMALIS is connection oriented which in turn implies having a faster packet loss recovery mechanism integrated as part of any communication being carried out on top of it. However, a caching system should be integrated in these series of experiments on top of HIMALIS to obtain a fairer comparison with CCN.
From the results it is clear as expected that the higher the MTU the better, however it is also clear from the results that for IoT traffic traversing core networks some aggregation techniques must be considered to release the load on the network nodes. In particular for

video, the http-based streaming techniques seem to be a good approach in that sense. Finally, the experimentation on top of worldwide distributed testbeds and employing the NEPI software to design the deployment of the scenarios and the execution of the experiments provides with repeatability characteristics to the results and representative numbers in terms of distances between the nodes.

Chapter 5

SDN ICNaaS for HTTP Video Streaming

The clean-slate approach of the FI although necessary in a near future seems risky and not easily accepted by the industry. A key enabler for FI, is Software Defined Networking (SDN). SDN was born in the academia but is gaining progressively more attention from industry, therefore making it more feasible in the short-term. Proposals based on SDN can be evolutive and non-disruptive. The control plane centralization with the early hardware support rocketed SDN as a solution to adopt and evolve the new architectures. The feasibility of SDN for video streaming, content distribution and caching have already been in the research focus. The difference with the previous work in this thesis lies in the limited impact this last proposal applies onto already existing content delivery elements, despite the adoption of SDN and the ICN architecture, the key players remain there just slightly modifying their role in some cases. Three are the main contributions of this proposal, the Information Centric Network as a Service (ICNaaS) in which the provision of independent ICN instances per provider allow the isolation and ease the acceptance of the new paradigm, to provide ICN like communications without new protocols or mixtures of TCP and UDP using well-known techniques such as delayed-binding and finally incorporate elements from nowadays Content Delivery Networks (CDNs) into possible future enhancements like caches thanks to maintaining HTTP during the whole communication. All these contributions are provided by means of SDN.

5.1 Description

Humans are visual beings [115], consequently, we aim for visual communication. Therefore, it is just natural for us transmitting graphical information by any means at our reach. Unavoidably, the appearance of computer networks, text based at first and picture capable later on, and their democratisation and enhancements in terms of bandwidth in the 90s lead to transmit video on top of the Internet which has commonly been known as 'video streaming'.

Video streaming, as any other technology, has evolved from its simpler approach to more complicated forms, answering to the challenges imposed by the environment. The first issues to overcome were processing power and bandwidth limitations. The democratisation of the Internet produced also as a side effect the birth of de facto standards in terms of what could be expected as available for the two former limitations. As a consequence, specific standards for video coding and video transmission were defined by standardisation bodies like Organization for Standardization (ISO), International Electrotechnical Commission (IEC) or Internet Engineering Task Force (IETF), among others.

Traditionally video transmission techniques have been based in a conjunction of session oriented connectivity (mainly TCP) for signaling and datagram oriented connectivity (mainly UDP) for data transmission. The use of datagram oriented connectivity was aimed to maximize the available bandwidth while minimising the delay for each video data piece. This has been true up to the democratisation of high speed links provision by Internet Service Providers (ISPs) in the last mile, as well as the evolution of the computing power. These two milestones have completely changed the way we face video transmission from 'what do we have available' to 'what is the maximum we can afford for'.

Available bandwidth increase has been the most important factor in the recent video streaming evolution. Session oriented protocols (TCP based) are nowadays employed for data transmission, breaking with the historical election of the datagram based ones. The rise of this approach was fostered by the difficulties that firewalls and Network Address Translations (NATs) imposed to the traditional datagram transmission, among others. In addition, it has been demonstrated [43] that having twice the bandwidth available related to the desired bit-rate makes TCP affordable as transmission technique.

Once TCP has been accepted as a desirable solution, it was a matter of time that the universal keystone of the Internet, the HTTP protocol, came into play. DASH [44] [46], previously introduced in this document in section 2.1.3, is the acronym for Dynamic Adaptive Streaming over HTTP. It reuses already existing and consolidated technologies, such as HTTP and eXtensible Markup Language (XML), to enable efficient

and high-quality media delivery through networks. The idea behind DASH is to create redundant metadata, which provides extra functionality with insignificant overload to the network architecture and service provider. Thus, it delegates to the client most of the complexity.

To ease the streaming process DASH splits the media into small chunks of data which are indexed with a so called MPD file. A single MPD file is able to contain different representations for the same content with different characteristics. Therefore, a client can easily select or switch between different versions of the delivered (streamed) file, with different qualities like bit-rate or picture size. The standard does not define how, when, and why a client should take any particular version of the media and this aspect is an open research field right now.

That said, the main advantage of DASH over other (non HTTP based) existing streaming mechanisms is that both the chunks and MPD files can be easily stored in already existing HTTP caching infrastructures. In addition, DASH will straightforward take advantage of almost any optimization that could have been applied to the existing infrastructures such as the CDNs. For this and other applications, the DASH standard defines profiles and allows different modes (live, on-demand, and others) to provide interoperability and suitability for different services.

As an evolution of both, the TCP/IP model and the CDN model, the Content Centric Networking (CCN) architecture brings the benefits to the network level of the move from host-to-host, end-to-end communications to network-guided, content-centric communications. It is designed following the ICN approach and, thus, places information pieces (content) as the central element of the network, making clients declare their "interest" on content pieces and providers to offer and deliver them to the intermediate network elements which, in turn, collaborate to deliver the requested content pieces to those clients. CCN protocol bases its operation on two message types. The *Interest* message is used to request data by content name and it can even identify a specific chunk of content to retrieve. On the other hand, the *Content Object* message is used to supply data. The messages of this type contain, besides data payloads, the identifying names, mandatory cryptographic signatures, and identifications of the signers (called the publishers) along with other information about the signing. Formally, a *Content Object* message is an immutable binding of a name, a publisher, and a chunk of data.

The approach followed in CCN departs from a clean slate where all the advances in the Internet infrastructure from the last years are discarded for the sake of breaking the IP ossification and applying named routing and in-network caching. For that to happen, adapters are needed so that for the customer applications the changes are transparent.

On the other hand, reusing the existing CDN actors such as caches and the efforts and evolution of those networks elements in a transparent way without completely breaking the ossified network seems to be a good preliminary approach to the deployment of other more disruptive ICN solutions. On that behalf, a proposal for directing traffic from clients to caching elements and servers based on the content and not on the addresses by the communications peers is done.
Finally, SDN as one of the key enablers for the FI has been also envisioned as a possible solution to achieve the ICN deployment in the real life. The use of SDN provides with the means to redirect application level traffic, replace addressing where needed and react to network events dynamically. Although SDN is in some circles also understood as a clean slate, which is partially true for the link layer, in others is envisioned as the tool that is allowing a non-disruptive apparition of new solutions which discard the pre-established architectures.
In this section a Content and Information-Centric Distribution Networks as a service is defined, extending the XaaS ecosystem by employing SDN related technologies. The goal is to provide the means to offer content providers with differentiated caching services provided with enough dynamism to cope with user changing behavior offering the best quality of experience. We aim for a non-disruptive approach in which the actors and software entities involved in the final service provision and consumption are not modified.

5.2 ICNaaS Concept and Motivation

Content distribution services are traditionally implemented through a mix of techniques like HTTP redirection, Domain Name System (DNS) load distribution, anycast routing, and application-specific solutions, among others. As a result, a complex distributed system is in charge of redirecting users' requests to clusters of network caches. Such decision, i.e. the right cache(s) that must serve the content, is usually made based on the communicating endpoints (IP addresses) involved. Contrary to this, the ICN paradigm advocates for delivering requested resources based on their name and independently from the data transport [82]. This can potentially increase the efficiency and scalability of content distribution, but it typically requires the deployment of state-of-the-art protocols like CCNx [71] as introduced previously.
Authors in [50] highlighted some future directions that are directly related to the ICNaaS proposal:

- Service Ortiented Architecture - Or Service composition highly motivated by user

preferences.

- Dynamic Content - Content that is generated on the fly.
- An adaptive CDN for media streaming - Hosting on demand media content.

On the other hand, in the last few years we have witnessed the rise of SDNs and the high momentum they have gained [54]. By means of a logically centralized controller that maintains the global view of the network and exposes a programmatic interface, SDN offers huge opportunities for network programmability, service automation, and simplified management.
Until now and to the best of our knowledge, ICN and CDN alternatives are focused on providing one service for all the customers, meaning that unless different companies offer the service, the resources are the same for all the customers and personalization is to that end limited. In general, the information is cached in the nearest network node in the case of CCN with its replacement policy and in the CDN in the nearest cache with its replacement policy.
We consider that an enhanced service to be provided by ISPs would be enabling the content provider to arrange its own ICN within the ISP premises. With that and the adoption of ICN initiatives, the ISP benefits of uplink bandwidth consumption reduction in cases in which the content provider is not within the service provider's network and also diversification of market by offering CDN services, clearly differentiated from the competence, thanks to the integration on the network. As a consequence of adopting SDN, the CDN can be easily and dynamically rearranged, the provider himself could interact with the system. Also the caching mechanism can be modified on demand as well as the content to cache assignment algorithm. The content provider could potentially select where the data is stored based on the data itself and not only on the consumer location. In addition, the proposed system avoids contacting the provider each time some content is requested and is the network itself who resolves the request. Meanwhile, the client receives its content transparently, if desired by the provider, or after any bootstrapping action that could be also configurable, like payment or successful authentication among others.
In addition and thanks to the transparent approach of the proposed system, any existing caching entity, either hardware or software can be leveraged for the sake of performance. So even if it is true that virtualization technologies and Mobile Edge Computing (MEC) can play a role in the deployment of ICN in a near future by deploying software caches on the edge of the network, it is also true that can not compete with hardware appliances in some scenarios.

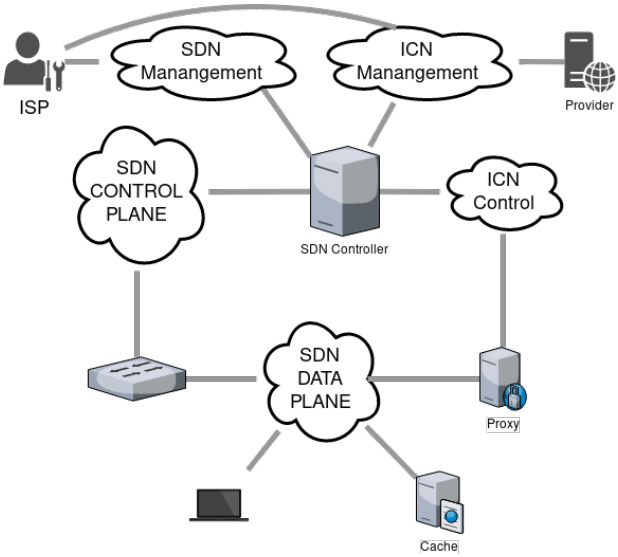


Figure 5.1: Controller interfaces

To clarify what is the environment of a SDN controller providing ICNaaS by having an application sitting on top of the controller or an entity collaborating closely with such controller, Figure 5.1 is provided.
A typical SDN controller environment is composed primarily of the, so called in the related bibliography as, control plane (SDN CONTROL PLANE in Figure 5.1) and the data plane ('SDN DATA PLANE' in Figure 5.1). The data plane makes reference to the switches and the links between them to which the controller does not (usually) have direct access (unless in-line control plane is used). The control plane is the usually private network that provides access to the administrative network, on top of which OpenFlow Protocol (OpenFlow) messages are exchanged.
Usually, a SDN controller offers an external interface in the form of REpresentation State Transfer (REST) Application Programming Interface (API) and/or web interface as well as cli commands. That is represented as 'SDN Management' in Figure 5.1.
Apart from the usual interfaces employed and offered by a SDN controller, the ICNaaS proposal provides with two new interfaces related to ICN, the 'ICN Management' interface to be employed by the provider to set up and operate its ICN instances as well as the ISP to modify the service as a whole. And the 'ICN Control' that is employed by the proxy to communicate with the ICNaaS application usually sitting on top of the SDN controller. More details of these two layers can be found in 5.3.1.1 and 5.3.1.2 respectively.
Adopting ICN for HTTP services has a major drawback, the Uniform Resource Locator (URL) needs to be inspected to decide the final destination of the communication. For that,

in general two approaches are followed, introducing an adapter that transforms the request once the HTTP containing the URL is received to a domain specific request (Interest in the case of CCN), or introducing a proxy in the middle that acts as a man-in-the-middle similarly to how CDNs capture customer requests having two TCP sessions on each side of the proxy, one for the customer and one for the media source (the provider or the cache). Usage of proxy is advocated due to the interest in maintaining the HTTP communication between the man-in-the-middle in charge of extracting the URL and the media source.
In the proposal, traffic traverses the network as it leaves the source. In that sense, the traffic leaving the customer would be directed to the provider IP address with the mac solved by Address Resolution Protocol (ARP) and with source IP address the customer's. Although the SDN will redirect the TCP SYN message to the proxy it won't be accepted unless iptables and other tricky mechanisms are employed, the same would do for the traffic from the proxy to the media source. Taking advantage of the SDN network IP and Media Access Control (mac) rewriting are employed. Speaking about OpenFlow, this means that the minimal version acceptable for this traffic treatment is 1.3. And, off course, the rewriting needs to be reverted by rewriting source IP and mac on the way back.
In my vision, the same content provider could desire different caching algorithms based on the type of content being requested (web pages vs multimedia content), the user requesting the content (premium user differentiation) or any other future strategy.
In Figure 5.2 a multi-site SDN enabled ISP network is represented. Customers of the network are represented with laptop figures and content providers are situated apart of the ISP network but could be also part of it, this representation of the providers does not tie them to a single server but as a single management entity. The SDN Controller with its primary role is represented as well. At the bottom of the figure the ICN overlay is represented. On that overlay, three different ICN instances are represented. The simplest ICN instance ('ICN instance 1' in the figure) corresponding to Provider1, another rather simple ICN instance denominated 'ICN Instance Basic' which is related to Provider2 and intended to support users with basic contract with Provider1 and finally 'ICN Instance Premium' also related to Provider2 which is intended to offer high availability and performance to provider's premium users. Note that the terms Provider's user and ISP's users are differentiated although they are the same so collaboration between ISP and provider is foreseen as interesting for both sides.
Having multiple ICN instances not only allows customized behavior per provider but also provides with the means to have customized behavior for different customers and/or contents for the same provider, as has been represented in Figure 5.2 with the ICN Instance premium.

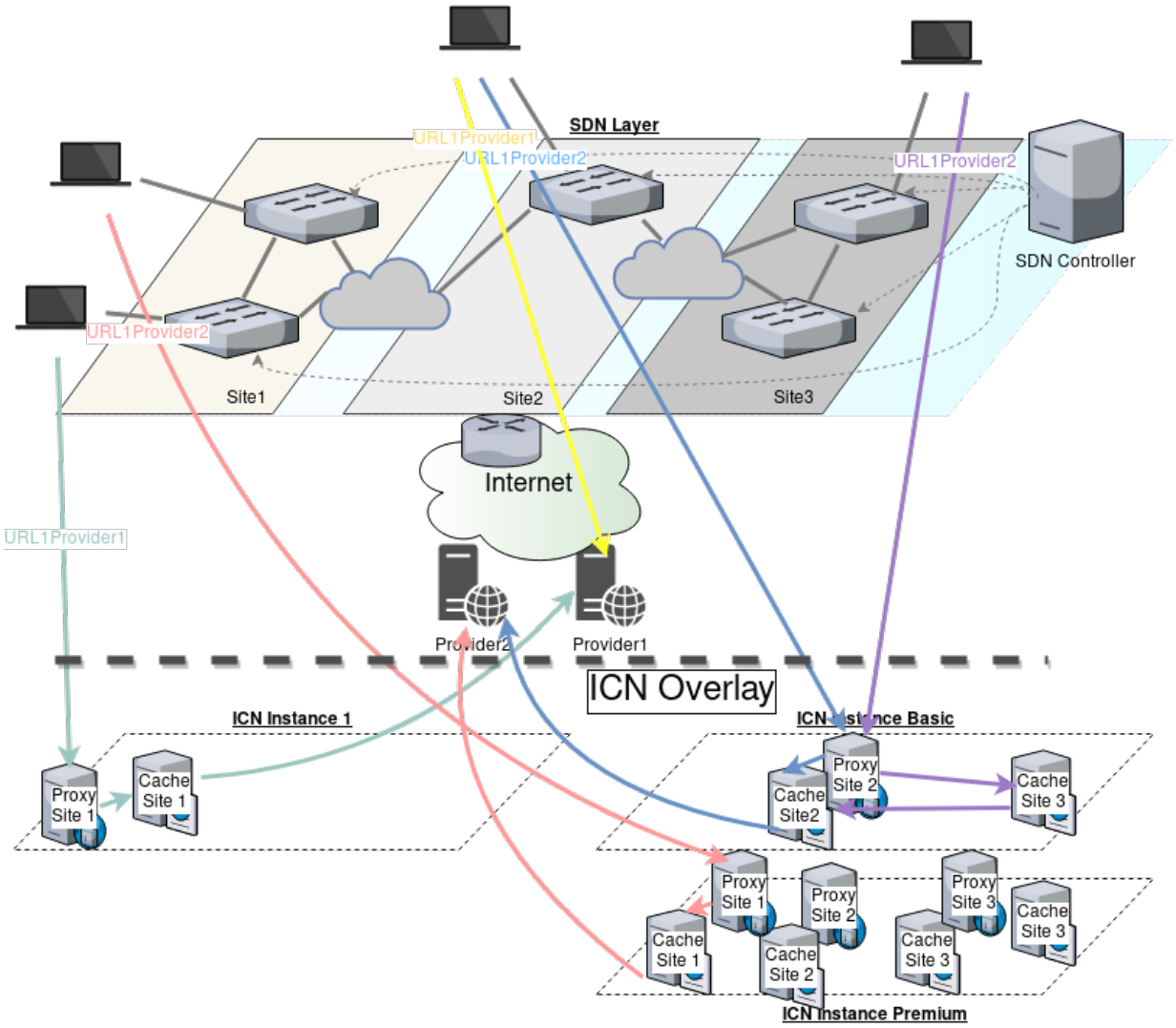


Figure 5.2: Leveraging SDN for ICNaaS

Let's focus on how the ICNaaS is delivering the content for each provider and client. 'Client1' wants to access 'URL1' of 'Provider1' via HTTP. The request arrives to the SDN network and is redirected to the 'Proxy Site1'. At this step the first decision has been made, which proxy is going to serve which clients, unfortunately and because of the backward compatibility with HTTP we are not still in the position to inspect which content is going to be requested, so the decision must be made in an ossified manner for each ICN instance but being possible to be different for other ICN instances. After finishing the TCP handshake, the HTTP request is received and thus the 'URL1' is acquired and delivered to the 'ICN controller' (which is can be sitting on top of the SDN controller or at least tightly related to it). The 'ICN controller' makes a decision to which cache the request should be directed if any, as can be seen in the request to 'Provider1' from 'Client3' the request could be directed to the source based on billing, regional policies or whatever decision is codified on the caching policy algorithm. The proxy request is then redirected to the 'Provider1' to obtain the desired resource. The term redirect here employed means programming the SDN network to make the packets flow from one point to the other of the network plus adapting each packet to the destination characteristics to be accepted by the Operating System network heap of the receiver.
'Client3' is also customer of 'Provider2' and when 'URL1' is requested to the former it is redirected to 'Proxy Site2' from the 'ICN Instance Basic'. Similarly to in the previous case the request is served by 'Cache Site2' after accessing to 'Provider2' web server. 'Client2' on the other hand is also client of 'Provider2' but is subscribed to a premium account and thus is redirected to 'ICN Instance Premium'. How 'Client2' is identified as premium is completely out of the scope but suffice to say that a previous authentication could bootstrap the ICN Instance with the information needed to identify it, this information is nowadays limited to the matching fields of the SDN network (primarily OpenFlow).
'Client4' performs three different requests for different content which is served by the nearest cache ('Cache Site 3'), another served by Site2 cache and finally content directly provided by the provider itself. All these requests have the exact same source network address(IP of 'Client4'), the same destination network address(IP of 'Provider2') and the same destination transport port(TCP port for HTTP). Nevertheless and following the ICN premises they are served by different entities based on the URI in the request. How the destination of the request is decided is something that is addressed in Section 5.4.
Although it is out of the scope how the proxies and caches could be instantiated it is clear that virtualisation could come into place taking advantage of MEC deployments to bring caching elements and proxies closer to the customer.

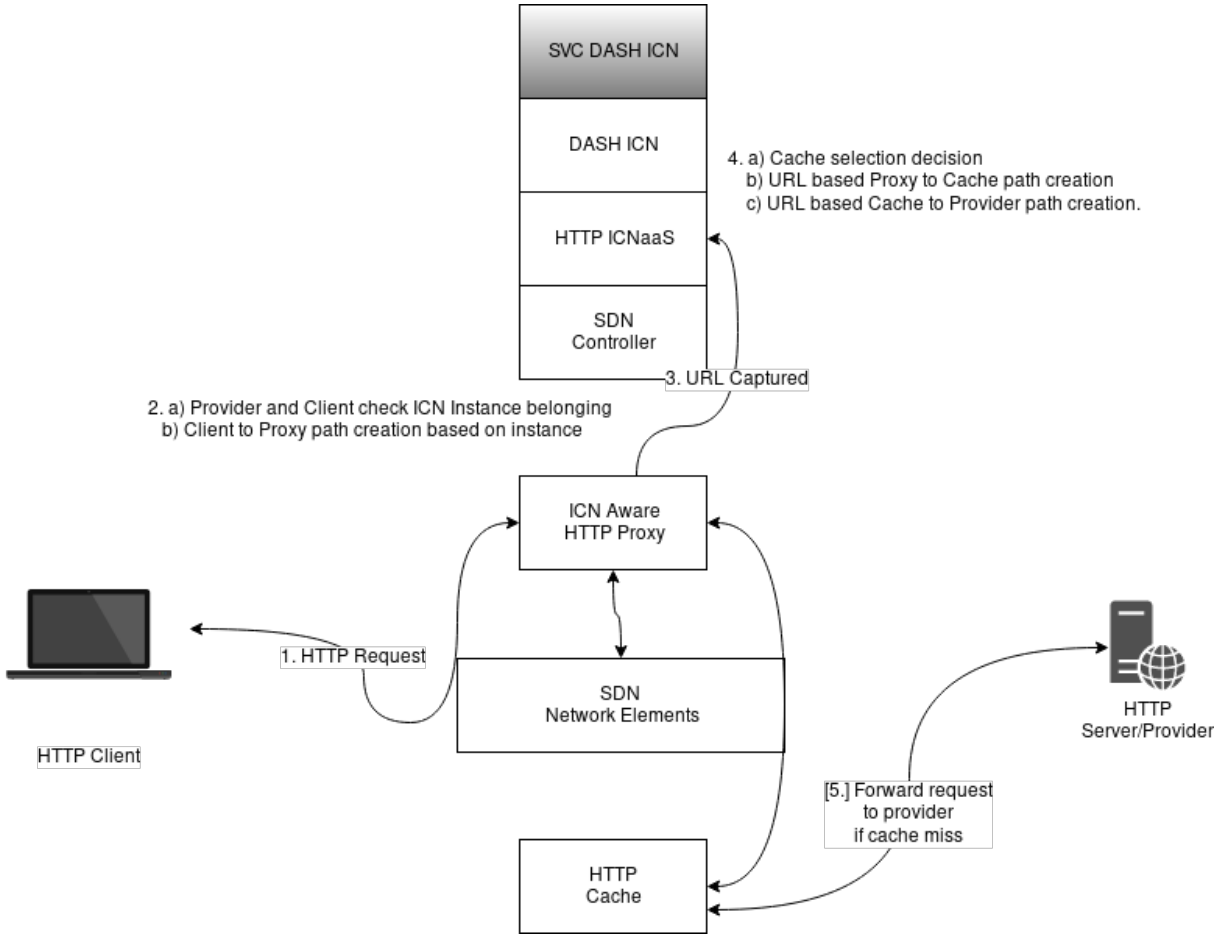


Figure 5.3: ICN over SDN with Proxy

5.3 SDN Controller layering

The SDN Controller intelligence needs to be extended by means of applications running on top of it. From my point of view three extra layers (see Figure 5.3) need to be incorporated to the system to achieve the desired results:

- ICNaaS layer - In charge of providing with the interfaces that allow the provider to instantiate its own ICN, register caches, proxies and if desired select which clients and servers will be involved in the ICN instance.
- Protocol Specific - In our case the application is DASH video with its particularities.
- Data specific - Different video codecs can be used on top of DASH and in particular H.264/SVC with its particularities regarding scalability and the possibilities this technology offers from the point of view of network optimisation and storage.

5.3.1 ICNaaS layer

The ICNaaS layer provides with the means for a content provider to create and manage ICN instances. Related to those instances some information needs to be stored in order to allow the system to operate taking into account that the technology involved is SDN.
In order to make any communication agnostic of the ossified IP layer that the software entities might be running on top of, the assigned IP address as well as the TCP port in which the deployed element is working are needed. In addition, to direct the flows to the desired location, the network element identifier (i.e, the OpenFlow Datapath ID (dpid) in OpenFlow) and the port number are needed. Since the system will be redirecting traffic to different machines from which it was originally directed, the mac address is also required to be rewritten.
Data to be stored per entity involved in the communication:

- Base URL - The Base URL(s) to which the ICN is attached and will react to.
- Caches - The cache IP address, TCP port and mac, as well as the dpid and port of the network element to which the cache is attached.
- Proxies - The proxy IP address, TCP port and mac, as well as the dpid and port of the network element to which the proxy is attached. Also the IP address of the SDN controller is needed for signaling.

- Content Servers - The content server IP address. It is not expected to have the content server as part of the icn but af an already deployed service available on the Internet, therefore only the address is needed and default network resolution mechanisms would be used.

- Clients - The client filter is completely optional but it is foreseen as interesting being able to determine to which clients the ICN instance will be serving. Filters can be produced by any means included in the OpenFlow matching field but in general IP address ranges with TCP ports and dpids with ports would be envisioned as desirable.

5.3.1.1 ICN management communication specification

The provider needs an interface that provides with the means to manage the ICN instances, the ICN management plane (see Figure 5.1). This interface should offer at least the following actions:

- Create a ICN instance with a name or description. In addition, the caching policy algorithm could be specified. And for that an interface for retrieving the available algorithms should be provided.

- Assign caches to an ICN instance. The minimum information to be provided are the mac and IP addresses of the entity, while TCP port would be desirable and if possible also the location in the SDN network. The location is usually represented by the dpid/port duet.

- Assign proxies to an ICN instance. The parameters required are the same as for the caches.

- Register the provider(s) to which an ICN instance aim optimizing.

5.3.1.2 ICN control communication specification

The 'ICN Control' on the other hands must at least offer the possibility to inform about the URL being requested to the ICNaaS so that the caching policy algorithm is fed, the decision on the content source to be employed is taken and the network is programmed properly before the proxy actually makes the connection attempt. In addition, this interface provides with the means to the ICNaaS to influence on the data plane in a rather complex way, since it is true that a controller can generate traffic in the data plane through the network elements (switches) by employing the control plane protocol (OpenFlow) but in

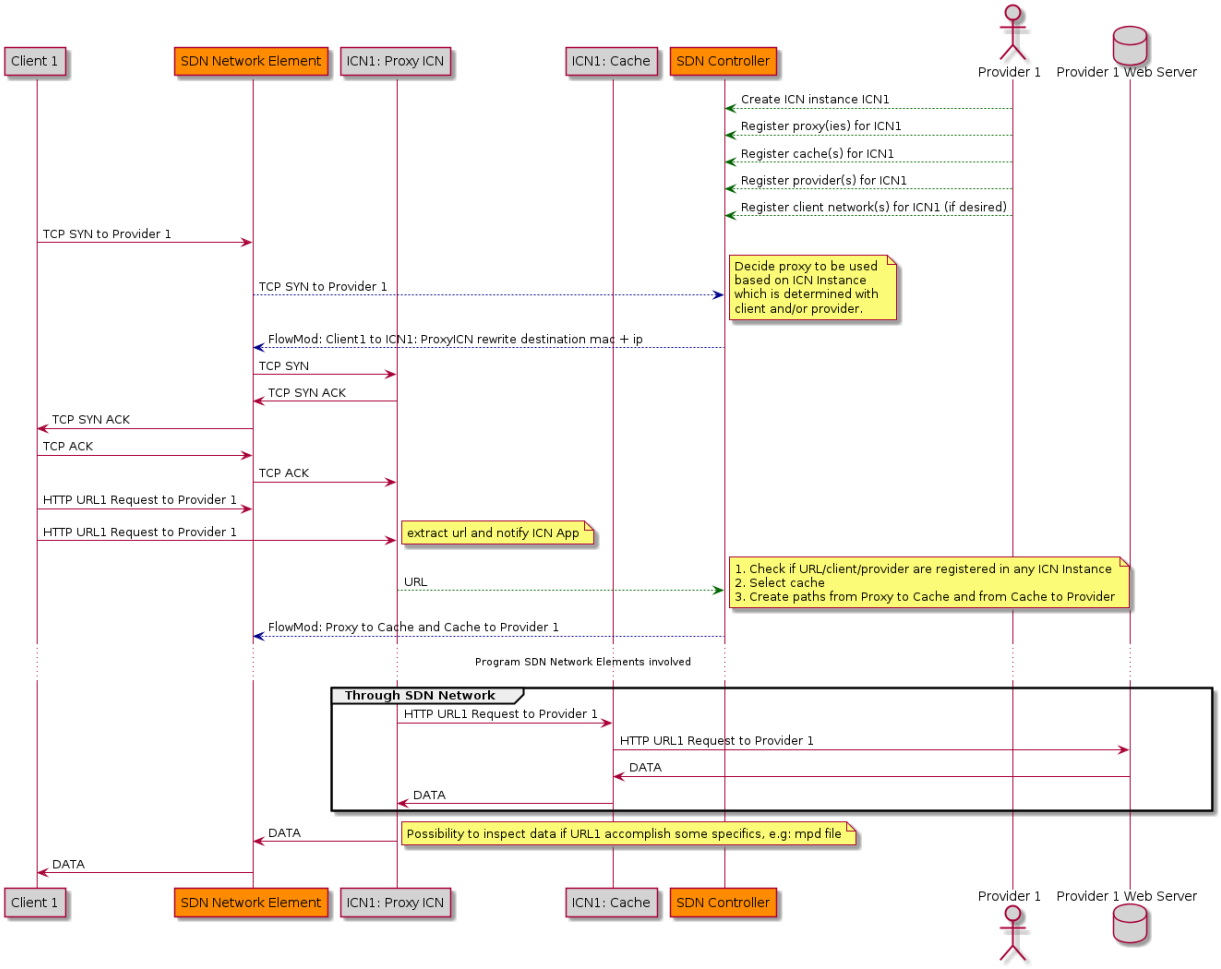


Figure 5.4: Interactions diagram

some cases it is interesting to delegate part of this burden to third parties such as a the proxy.

From the point of view of an ISP exposing internals of their networks such as dpid and ports numbers. Nevertheless information about the location of the caches is needed so that the provider can arrange the caching topology in a meaningful way for the supported service and the intended users. Moreover with the network slicing and virtualization technologies being researched and deployed nowadays, the network elements might probably be virtual instances offered exclusively to that precise provider at that precise moment, reducing the exhibition of the ISP internals.

5.3.2 Protocol Specific Layer

The protocol specific layer is in charge of directing the flows to the ICN elements and to attend the proxy signaling to behave in concordance to the information being requested.
In the case HTTP in general is driven by the inherent nature of the underlying transport protocol TCP. Before any data belonging to HTTP is transmitted there is a connection handshake which, from the point of view of an network element, is composed of three messages [116]. In order to be able to extract the requested URL which is in turn the information requested, the *interest* continuing with CCN terminology, a full handshake process is needed. Once the handshake is finished it is impossible to transparently redirect the client to another data source. That is the point were the proxy comes into play. The TCP splicing [117] or delayed binding [116] is a technique widely used and introduced by proxies to leverage on the kernel the rest of the communication once a milestone has been reached, reducing resource consumption. In this case the SDN controller leverages on the proxy to reduce bandwidth consumption by delegating the TCP operations to the proxy as well as the HTTP inspection.
The connection is directed to the proxy that after finishing the 3-way TCP handshake receives the request with the URL (i.e, HTTP GET). In that precise moment, the obtained URL is provided to the SDN controller DASH layer. With the configuration already provided to the ICNaaS layer by the provider and the url, the next hop in the communication is to be decided.
The caching policies algorithm, that decides which cache or server is going to be directed the communication to or if the communication is going to be dropped, is configurable per ICN instance by the provider and can potentially be changed at any time.

5.3.3 Data Specific Layer - H.264/SVC

The Data Specific Layer is in charge of providing the system with behavior related to the data domain, in the case of DASH video streaming it is related to the video/audio coding. In case the requested URL was a MPD, the ICNaaS sitting on top of the SDN controller downloads a copy and employs it as input for the caching policies algorithm associated with the ICN instance.
If the MPD contained H.264/SVC video definition, some interesting properties of the format can be leveraged. H.264/SVC video provides the system with the means to hierarchically distribute the cached content based not only on the URL as content identifier but also on the scalability level that indicates the relevance of the data chunk for the whole

service. Thanks to the scalability level, the caching policies algorithms can be easily driven to reduce the uplink bandwidth to the provider server, reduce the delay for the user, among others, as is extended in 5.4.
Other Data Specific Layers are envisioned such as for H.264/MVC for stereoscopic video.

5.3.3.1 ICN control communication specification

At this point, the future content could be pushed into the proper caches (prefetching) [94] actively instead of reactively when the requests are performed increasing the cache hit ratio, depending on the policy. To that end, the role of 'cache accelerator' is introduced. Although some appliances could offer with the means to request for content caching, there is also the possibility to actually create fake HTTP requests that will trigger the caching mechanism transparently and vendor agnostic. The term fake here make reference to the fact that these requests are not issued by a customer but by the ICNaaS predicting what will be the behaviour of the customer for the actual network. The introduction of the actual network in the equation is an important step that, thanks to the involvement of the ISP's sdn controller has advantage in the information of the available bandwidth for the customer. With that information, the video versions with a bit-rate higher than the available bandwidth could be directly discarded and thus not prefetched. The cache accelerator can perfectly be implemented as part of the proxy reducing the number of trust relations of the ICNaaS application that sits on top of the ISP SDN controller that is a critical component of the network.

5.4 Caching Policies Algorithms

The caching policy algorithm is in charge of deciding which cache or server is the communication going to be directed to, or if it is going to be dropped.
Although the caching policies should be configurable for the provider, it is also true that the ISP could benefit too from specific policies (e.g, reduce the backbone utilization by leaving the more information the better nearer to the customer). It is foreseeable a pricing system in which the algorithms that benefit the ISP could be awarded with lower billing for the content provider hence the relation between the ISP and the ICN Management in Figure 5.1. Nevertheless this feature is out of the scope of this thesis and is left as future research.
Main factors to decide on which cache a content should be directed to and hence cached is the cache size and the cache location. The objective is to maximize the amount of data

stored in the caching system while bringing it closer to the customer. So there are two main problems, what needs to be cached and where.
Since the focus is on video streaming over HTTP and DASH in particular, the caching decision can not be decided by file extension, unless the particular case of MPD files which is negligible compared to the size of a video stream. But for the case of H.264/SVC some techniques can be applied to prioritize chunks based on the layering scheme.

5.4.1 H.264/AVC over DASH

Usually H.264/AVC streams over dash are encoded with multiple representations each of which has its own parameters as can be seen in Listing 5.1.

```
1 <Representation id="1920x1080p25" codecs="avc3.640028"
2 height="1080" width="1920" bandwidth="4741120" />
3 <Representation id="896x504p25" codecs="avc3.64001f"
4 height="504" width="896" bandwidth="1416688" />
```

Listing 5.1: H.264/AVC representation definition in an MPD file.

Each representation has its own Segments and is independent from each other which means that two clients requesting the same content at different bit-rates will need independent content caching procedures.
It is important to take into account how the Rate Determination Algorithm (RDA) executed in the streaming client is affected by the caching decisions taken on the network. Authors in [118] already took into consideration the effect of cache hit and miss on the RDA producing bit-rate oscillations for DASH. Authors introduce the 'ViSIC' caching system that performs traffic shaping to avoid the changes in the perceived bandwidth in the RDA that produces the oscillations. Thanks to the SDN on which the ICNaaS proposal relies, meters can be used to shape the cache responses for a certain customer avoiding this effect. In this solution the MPD request is duplicated within the ICNaaS application and is processed to use it as input.

5.4.2 H.264/SVC over DASH

Unlike H.264/AVC, H.264/SVC over DASH representations are dependable in the same relation that the H.264/SVC layers depend on each other as can be seen in Listing 5.2 where representation 2 depends on 1 and 0 while representation 1 depends on 0 only.

```
<Representation id="0" codecs="AVC" mimeType="video/264"
 width="1920" height="1080" frameRate="6"
 sar="1:1" bandwidth="2325553">
<Representation id="1" dependencyId="0" codecs="SVC"
 mimeType="video/264" width="1920" height="1080"
 frameRate="12" sar="1:1" bandwidth="3209141">
<Representation id="2" dependencyId="1 0" codecs="SVC"
 mimeType="video/264" width="1920" height="1080"
 frameRate="24" sar="1:1" bandwidth="4019194">
```

Listing 5.2: H.264/SVC representation definition in an MPD file.

Evidently, all the representations depend on the base layer (with representation id 0) which in turn is an H.264/AVC stream. Meaning that in the same case as in H.264/AVC where two clients request different representations of H.264/SVC streams, there is a higher probability of having the content cached, depending on the H.264/SVC encoding options set and the layers obtained, assuming infinite cache sizes, after the first retrieval the base layer will be always cached and thus a 100% cache hit ratio is obtained. However, each representation downgrades the codification efficiency in not more than a 10% [119] and by using H.264/SVC more video clips at different representations can be cached.

Authors in [120] highlight the possibilities of H.264/SVC for transport and caching differentiation. Focusing on caching differentiation, authors state that H.264/SVC aware eviction mechanisms can boost the efficiency of network caching. The idea is to take advantage of the manifest file to avoid eviction of future video chunks once an ancestor is requested, since following chunks will probably retrieved later. The authors also introduce the possibility of prefetching as an enhancement.

Next, there is a list of ideas on how caching distribution could be steered to take advantage of the H.264/SVC characteristics while employing the ICNaaS. The ICNaaS offers the possibility to have different caching distribution algorithms per ICN instance. The eviction policy is restricted to the possible connection from the ICNaaS towards the cache node for signaling the decision, since that is highly dependable on the caching solution it is not desired for an architecture that tries to integrate existing solutions smoothly. Nevertheless, it would be possible to implement it but methods to retrieve the cache size and to signal eviction victims would be needed.

In tree like networks were the cache nodes present a hierarchical structure, considering a leaf node, the caching node nearer to the client and a root node those that are closer to the upstream link and therefore the content provider:

- Cache base layer on root nodes and enhancements in leafs. Minimize trunk traffic after content cached. Maximizes hit for base layer but with bigger delay. Off course, if the tree has a depth greater than one, different enhancement layers can be cached at different levels of the tree.

- Cache base layer on leafs minimizing the delay needed to the reproduction to start if the client accepts start playing with lower quality, that part can not be influenced by the network.

Other approaches can be implemented in almost any network structure, let it be hierarchical, full mesh or any other:

- Collocate N first chunks for every layer on the closest cache. In this case, the playing delay is definitely reduced and the client cache can compensate the delay of having the rest of the stream further away.

- Avoid costly or problematic links by caching the content on a certain position in the network. Although the ICNaaS hasn't got a Path Computation Element (PCE) itself, it may extract information from the controller to decide whether a position in the network, as an example, can reduce congestion.

- Cost based. Caches might have different costs, if storage is SSD or legacy or if it is located in the ISP premises or in a third party. Based on that information the algorithm might take decisions to minimize costs while maximizing Quality of Service (QoS)

5.5 User driven

Historically, multimedia adaptation to network conditions has been performed on the server side or in Media Aware Network Elements (MANEs) that downgraded the content to fit into the actual network condition. In [121] a proposal for a H.264/SVC aware MANE was made that would receive the user preferences for the three possible scalability parameters of an H.264/SVC stream so that the unavoidable adaptation decided by the network was driven by the user.
The proposal was not meant to replace automatic adaptation, but rather to enhance it by enabling the user to steer it. We therefore introduced two steering parameters:

- Layer drop priority.

- Minimum number of enhancement layers for each scalability dimension.

The layer drop priority is expressed as an ordered relation of the scalability dimensions D (for spatial resolution), Q (for fidelity) and T (for temporal resolution). This way the client is able to specify the requirements through the dropping priority of the scalability dimensions. Optionally, minimum values can be set for each of the scalability dimensions. Note that the priority id which is defined as a header field in H.264/SVC [34] has similar semantics, but cannot be used for expressing user preferences, since it is bound to the content.
Besides the steering parameters, the usage of a constrained environment is required to trigger the adaptation. Note that below the focus is on the available bandwidth, however this could in theory be replaced by any other constraint which may trigger an adaptation. The algorithm for the DQT selection, based on the steering parameters, is defined using pseudo code as follows:

Listing 5.3: Algorithm Pseudo-Code

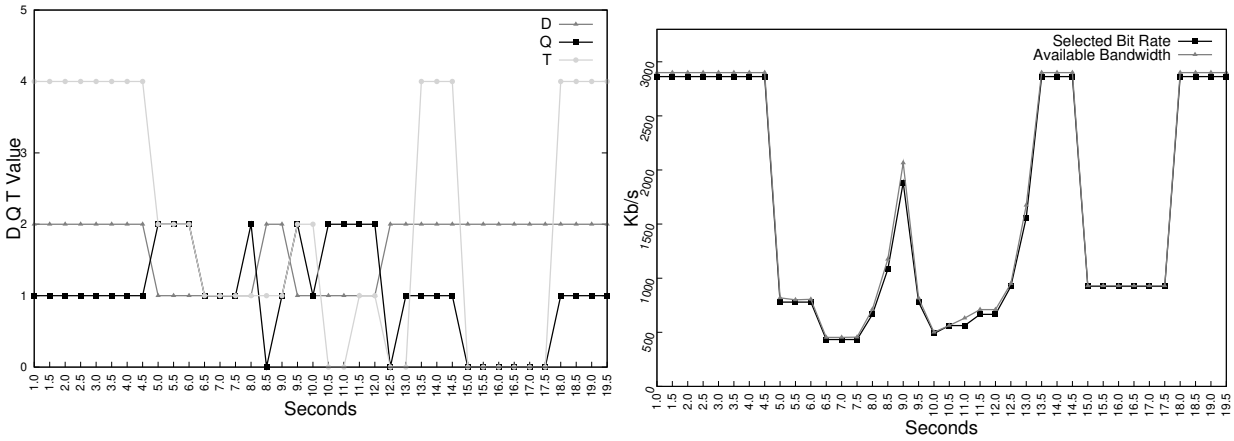
```
Sort scalability dimensions according to drop priority,
from highest to lowest;

while (available bandwidth <
   bitrate of current enhancement layer selection) {
 for each scalability dimension {
  if (num layers of scalability dimension >
      min layers selected by the client) {
   Start filtering the currently highest layer
     from this scalability dimension;
   Stop filtering layers from scalability
     dimensions with higher drop priority;
   SelectionFound = true;
   End for loop ;
  }
 }
 if (SelectionFound == false) {
  Perform best effort adaptation;
 }
}
```

Obviously it would make no sense to adapt the bit-stream if there is enough bandwidth available. Thus, the algorithm is only triggered if the available bandwidth is smaller than the maximum bit rate of the H.264/SVC. In this case layers have to be dropped in order to decrease the bit rate, as indicated by the steering parameters. This is done until a configuration of enhancement layers is found which has a bit rate smaller than or equal to the available bandwidth.
If no suitable enhancement layer configuration can be found, best effort adaptation is performed, selecting the base layer in the worst case.
In order to evaluate the approach the NS-2 simulator [11] was extended to include RTCP feedback and TFRC calculation. Additionally, the different layer selection

approaches presented below were implemented. The simulation setup consists of two nodes, representing a streaming server and a streaming client connected by a link. The bidirectional wire is configured to offer a total bandwidth of 2.9 Mb/s while the H.264/SVC video is encoded with an average bit rate of 2.89 Mb/s.
TCP Friendly Rate Control (TFRC) [122] is used to calculate the available bandwidth T for a certain session as defined in Equation 5.1.This function gives an upper limit to the bandwidth available in terms of bytes per second. The inputs to this functions are the packet size s, the Round Trip Time (RTT) r, the loss event rate p and the TCP retransmission timeout T_{RTO}. The implementation uses a mean of all packet sizes for s and a weighted moving average of the fraction lost for p, as shown in (5.1).

$$T = \frac{s}{r\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{5.1}$$

Having calculated the available bandwidth, the DQT selection algorithm is triggered. Congestion is simulated after four seconds, as can be seen in the Figures 5.5b, 5.6b, 5.7b and 5.8b. There is a delay until the actual adaptation begins, which can be explained with the weighted mean of the fraction lost p in Equation 5.1.
Another possibility to trigger the DQT selection is to manually change the available bandwidth, referred to as user triggered adaptation. This situation was simulated within the interval between seconds 14 and 18.
Four different approaches are compared in the evaluation. The first approach represents a best effort adaptation which selects the layer combination fitting best into the available bandwidth. The second approach sets minimal values for each scalability dimension D, Q and T, but no layer drop priority. The third approach supports layer drop priority but no minimal values. Finally the fourth approach corresponds to the one proposed in this paper, including minimal values and layer drop priority.
The video used in the simulations is the City MPEG reference video sequence with 2 spatial, 2 quality and 4 temporal enhancement values. The highest layer corresponds to 4CIF spatial resolution at 30 FPS with just one quality enhancement. For both lower spatial resolutions (CIF and QCIF) a second quality enhancement was encoded.
For the evaluation the layer drop priority has been set to $D > Q > T$, the minimum values chosen were D=1, Q=1, T=2. These minimum values correspond to CIF size and a frame rate of 7.5 FPS.
Figure 5.5 shows the results using the first approach, i.e. best effort adaptation. Figure 5.6 represents the second approach with minimal values for the scalability dimensions and



(a) DQT Selection

(b) Bit-rate evolution

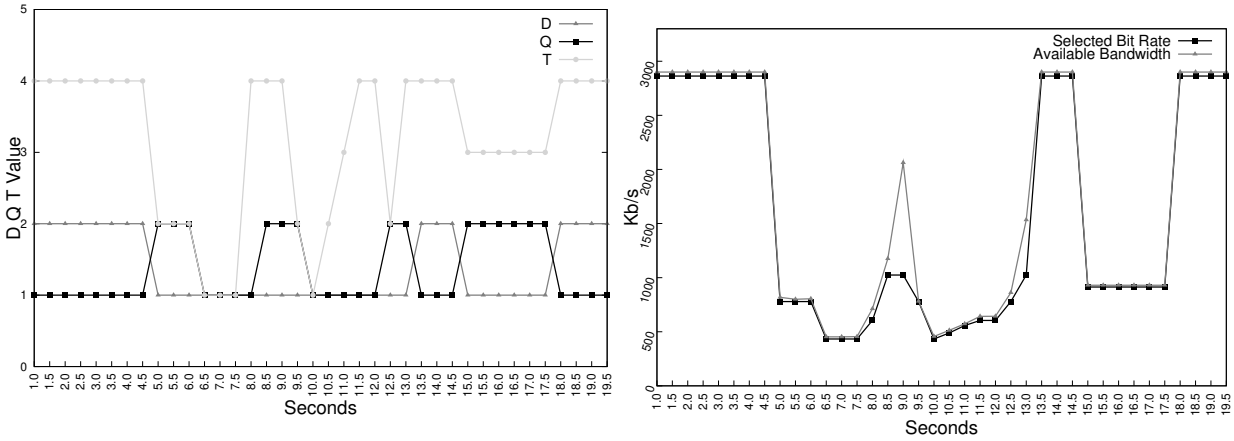
Figure 5.5: Best effort approach

Figure 5.7 corresponds to the third approach including the layer drop priority. Finally in Figure 4 one can see the results achieved by the proposed approach using minimal values and layer drop priority combined. Each figure consists of two diagrams. The left one shows the selection of DQT values by the approaches, while the right diagram depicts the available bandwidth which triggered the selection, as well as the corresponding selected bit rate.

In the interval between the seconds 4 and 5.5 one can observe that the approaches with layer drop priority $D > Q > T$ (Figure 5.7 and Figure 5.8) try to keep the temporal value as high as possible. When looking at seconds 10 to 11 in Figure 5.7a and Figure 5.8a, the difference between using drop priority or a combined approach becomes clearer. When in Figure 5.8a, the fidelity stays higher even at the cost of loosing temporal resolution due to the defined minimums.

When looking at second 14 in the Figures the effect of the minimums becomes clearer. The result of the first approach (Figure 5.5) is a 4CIF slide show (1.875 FPS), while in the second approach (Figure 5.6) the user receives a video in CIF resolution, at high quality and with 15 frames per second. The algorithms with layer drop priority (Figure 5.7 and Figure 5.8) keep 30 FPS although at the cost of reduced quality, which corresponds to the user request. Moreover, with the third and the fourth approach less bandwidth is required compared to the first and the second approach, while perfectly matching the user preferences using the fourth approach. This shows that both layer drop priority and minimum values are relevant for an increased QoE at reasonable bandwidth utilization.

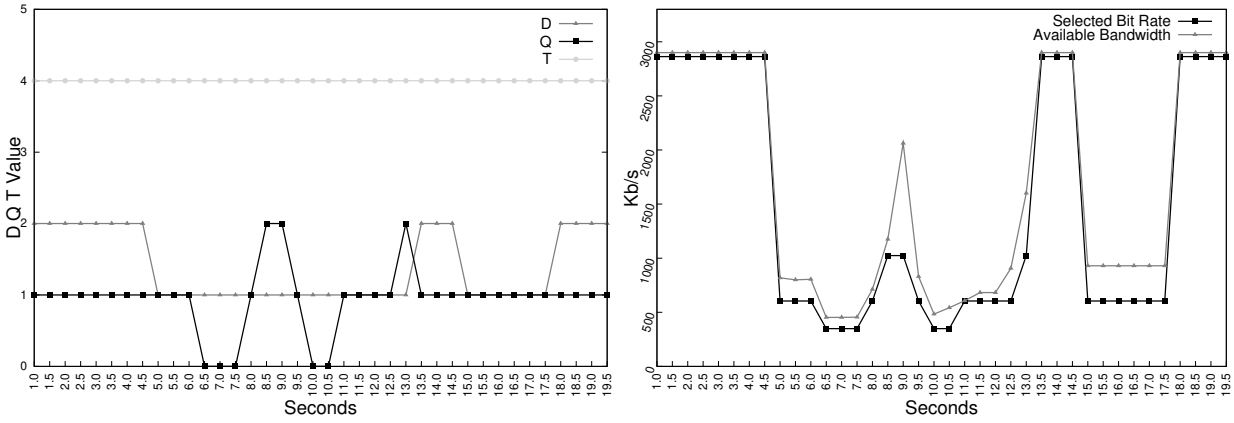
The adaptation using this approach enables a higher QoE compared to existing adaptation



(a) DQT Selection

(b) Bit-rate evolution

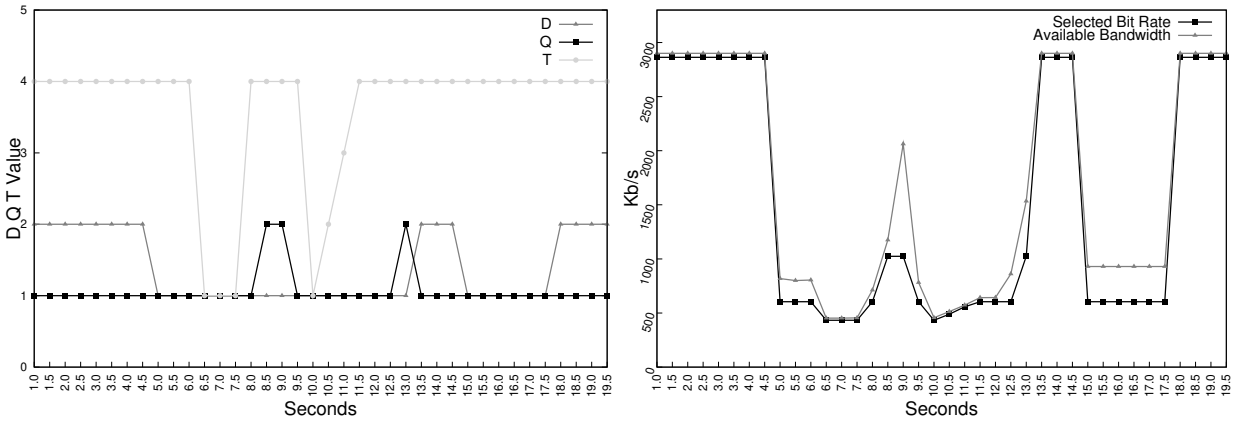
Figure 5.6: Minimal values approach



(a) DQT Selection

(b) Bit-rate evolution

Figure 5.7: Layer drop priority approach



(a) DQT Selection

(b) Bit-rate evolution

Figure 5.8: Combined approach

mechanisms at the same bit rate.

In the environment of H.264/SVC DASH video transmission, although it is true that naturally adaptation is made in the client side thanks to the RDA, it is also true that the ISP SDN controller on top of which this proposal is collocated has the information relative to the bandwidth consumed and available. It is therefore straightforward to make an adaptation of the H.264/SVC stream by informing the ICNaaS about the available bandwidth and trimming the requests that would exceed that value. But multiple layers of H.264/SVC could offer that working point in which the stream wouldn't exceed the bandwidth. Mixing in the user driven concept introduced in [121] would optimize that point providing not only with a better QoS but also a better QoE since it is the user perception and preferences the ones to steer the adaptation. Unfortunately this approach as was introduced in previous work there is a need for signaling which means that this approach would only be possible with client side coordination.

In addition, this approach is able to feed the caching policy algorithm to limit the prefetch of data chunks to those that fit into the available bandwidth thus reducing the uplink usage wasted in chunks that won't be able to go through the user link.

Since the proposal of obtaining the available bandwidth from the ISP SDN controller is limited to sitting on top of it plus limited by possible policy, political and law restrictions (such as anonymity), another system is proposed with which the available bandwidth can be calculated. The proposal is to employ the meter capability to collect statistics (OFPMF_STATS) [66] during the streaming. Those statistics give an idea on the mean bit-rate employed by the RDA so that when the latter tries to go over the average the requests can be blocked.

Another scenario in which the adaptation might be needed and user preferences could enhance the perceived QoE is in the case that the uplink to the provider is congested, in that case the ICNaaS could avoid launching requests from the caches to the provider.

5.6 Conclusions

This chapter introduced the proposal of ICNaaS and how it is possible to merge the ICN approach to that of the actual CDNs providing dynamism as an added value to the content provider while still maintaining the actual CDN providers but in a side role by employing SDN.

The '*as a Service*' part is really important because it aligns with the network slicing techniques that are being researched, in that sense, the ICNaaS system could be provided by the operator as a slice, or a third party could offer the service on top of the network operator slice or even further, each icn instance could be provided on top of a network slice. From the point of view of the proposal this changes completely the role of the SDN controller and the scope that is affected by the decisions made by the application.

Furthermore, offering ICN as part of Network Function Virtualisation (NFV) is foreseeable and this proposal serves as an starting point to provide this kind of services.

The presented approach leverages on SDN as many others do in the bibliography but the difference here is that the service is offered in a non-disruptive approach, by offering end to end HTTP without adapters as the basis for the communication, therefore making it transparent to most users nowadays. In addition, the proposal integrates actual caching systems avoiding 'nouveau' approaches thus relying on well known and reliable systems.

This chapter also has highlighted the importance of the SDN controller application layering. In addition to the separation of the controller APIs into northbound, southbound, REST among others, the SDN applications themselves need to take into account the diversity of the actions being taken by splitting in layers each functionality therefore making the code reusable and extensible.

The adoption of ICN with the centralized control system of SDN offers a new world of possibilities in terms of caching algorithms and how the traffic can be easily steered transparently to the decided end-point which in turn will provide with the requested content. The example of applicability to DASH and to H.264/SVC demonstrated the potential of this kind of systems.

Finally, scalable coding is regaining momentum thanks to the in-network caching and their inherent characteristics that allow distribution among the caching system related to the

relevance of the data itself.

One field on which the work presented in this chapter, and the rest of this Thesis, could be extended by the introduction of security in the equation. It is clear that the proposal of this chapter relies on the proxy to extract the URL from the HTTP connection, which in turn means, that the certificate of the server endpoint need to be deployed into the proxy as well, so that HTTP over TLS (HTTPS) transmission is possible and the endpoint is identified as the desired destination of the connection. This approach is already present in nowadays CDNs but the ICNaaS approach could simplify the deployment of the keys and what is more important the control over which entities store them.

The use of HTTP historically ties implicitly to TCP. Once enough bandwidth is assumed, TCP provides with reliable transmission not only in terms of packet transmission but also in terms of confidence onto the communication. Despite its skills, TCP complicates the task of actually inspecting what is going on for a certain transmission while its unreliable counterpart not tied to peer to peer binding makes easier steering the traffic. In TCP the traffic can naturally only be steered or redirected on the SYN message while UDP can be redirected afterwards thanks to the absence of session information (although SDP and RTSP could provide such session capabilities, they are bonded to the application level).

Although it is true that HTTP historically is tied to TCP, it is also true that other transport protocols have been used with HTTP, one of those protocols is SCTP and similarly to the proposal of Chapter 3 the mapping of H.264/SVC layers with SCTP streams could be utilized by the network to apply optimizations, deny access or apply bandwidth enforcement. The solution would nevertheless require a minimal synchronization between the SDN controller and the video player to inform which SCTP stream would be used by each scalability level. Other emerging protocols like *QUIC* and others related with HTTP/2 could be taken into consideration.

The term bandwidth enforcement in OpenFlow clearly calls for the meter feature as has been already commented above. Exploiting this feature to avoid waste of resources in DASH connections is also an interesting field of research. The idea would be to mitigate the effect of the bandwidth estimation algorithm on the client side by applying a meter to all the traffic related to that stream directed to the client. The challenge there is the limited meter amount that OpenFlow devices provide.

Chapter 6

Development and Evaluation of SDN ICNaaS

The design introduced in the previous chapter has been implemented and evaluated to asses the feasibility of the proposal. In this chapter, the details on the work performed to implement the proposed solution are introduced and evaluated not only the performance of the implementation but also the difficulties in integrating today's entities such as DASH clients, HTTP caches and HTTP servers into the ICNaaS proposal based on SDN.
Two set of experiments were performed. The former was carried on employing virtual infrastructure while the later has been performed by means of hardware switches. The experience obtained with the deployment of the first proof of concept served us as input for the design and deployment of the final experimentation. During the first experimentation phase, the ICNaaS was evaluated as a generic HTTP service where any static web based content could be delivered following the ICN paradigm by a provider that registered it in the system. On a second phase, the inclusion of the application level (DASH and H.264/SVC) inside the proposal is faced, covering one of the most important market niches in today's networks, video streaming.

6.1 Architecture Elements

6.1.1 ICNaaS

Three are the main parts of the ICNaaS application. The REST interface serving the content provider, the REST interface accepting requests from the proxy and the ICN functionality and flow management based on the caching policies.

#	Source	Destination	URL	Parameters
1	Provider	ICNaaS	onos/icn/icn	name, description, type
2	Provider	ICNaaS	onos/icn/proxy	name, description, mac, ip, port, prefetch_port, type, location (dpid,port) isProactive
3	Provider	ICNaaS	onos/icn/cache	instance, name, description, mac, ip, port, type, location (dpid,port)
4	Provider	ICNaaS	onos/icn/provider	instance, name, description, network, uripattern, ho

Table 6.1: ICNaaS's Provider northbound interface.

#	Source	Destination	URL	Parameters
1	Proxy	ICNaaS	/onos/icn/proxyrequest	uri, hostname, smac, source ip, destination ip, protocol, source port, destination port

Table 6.2: ICNaaS's internal northbound interface.

The REST interface with the provider allows the creation, modification and removal of ICN instances, as well as, registering proxies, caches and providers' servers. The classes providing the REST api defined in Table 6.1 are shown in Figure 6.1 and correspond to the 'ICN Management' interface shown in Figure 5.1. To implement the functionality the tools already offered by the ONOS Controller are used, in this case the *AbstractWebResource* which inherits from *BaseResource* that facilitates the implementation of REST interfaces for apps sitting on top of ONOS. The Path class is employed to create the URL that will be providing the REST call. The content of the calls is encoded as json and, despite not being represented in Figure 6.1, Codec classes are implemented for each Northbound class to 'de/jsonify' the internal representation of the ICNaaS elements into and from classes.
There is another REST interface that is intended for communication with the proxy and represented by the ProxyRequestNorthbound in 6.1 and defined in Table 6.2 that corresponds with the communication of the 'ICN Control' interface as shown in Figure 5.1. These notifications are the ones carrying the URL requested by the user and are responsible of triggering the ICN mechanism to redirect the communication to the desired content source (a cache or provider itself).
We are going to take profit of Figure 6.1 to enumerate the entities involved in the ICNaaS application:

- IcnService → The ICNaaS itself. The one in charge of keeping track of the ICN, instances and the proxies in the system. The IcnService is also in charge of creating

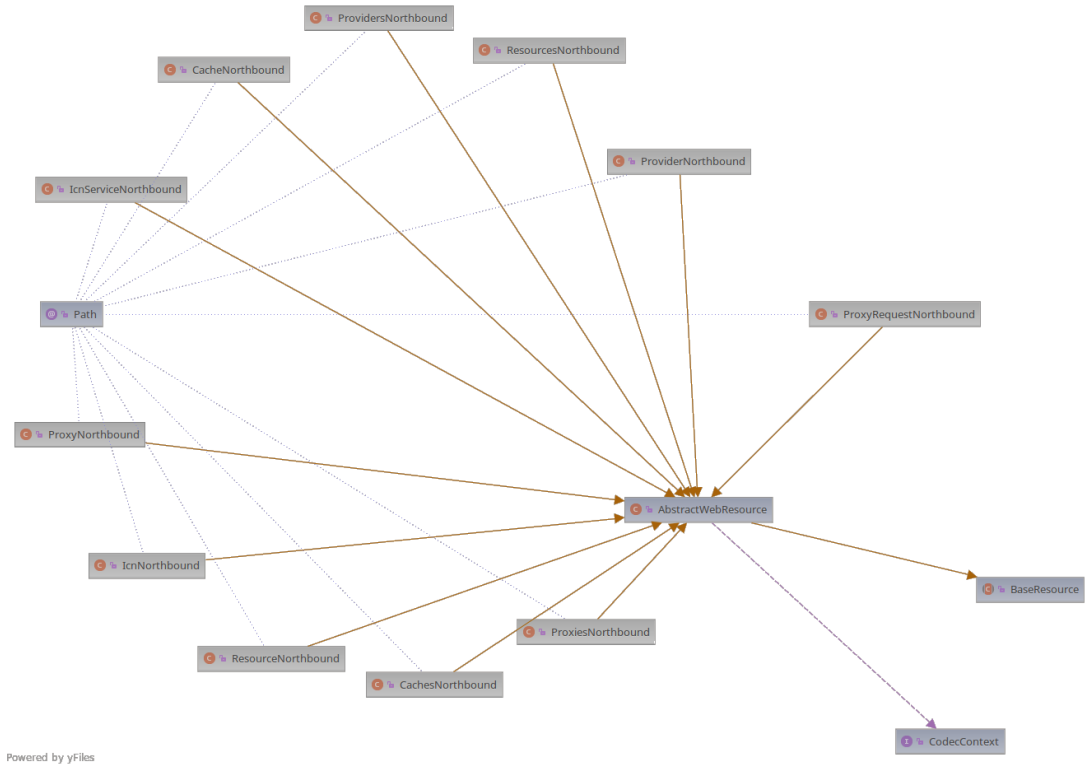


Figure 6.1: ICNaaS REST api.

the paths for the data to flow.

- Icn → Represents each ICN instance held in the system. It stores the information to the Caches and Providers related to the ICN instance as well as the information of the Resources already managed by the ICN instance.
- Provider → Represents a Provider which includes management of url pattern matching, as well as, IP address ranges belonging to the instance. It also manages providers' servers provided as content source at the ICN boot up instance.
- Proxy → Represents a Proxy instance deployed in the system. Information related to identification such as IP, mac or location (dpid and port) is stored.
- Cache → Represents a Cache instance deployed in the System.
- Resource → Represents a Resource already retrieved through the ICN instance and present in the caching system.

The *IcnService* is the central class of the ICNaaS architecture, as can be seen in Figure 6.3. The class is implemented as a Karaf Component and Service so that it can be easily instantiated from other Karaf components if needed. The decision to employ Karaf derives from the controller election since Open Network Operating System (ONOS) is based on Karaf and therefore the applications are developed as components. It contains the mechanisms to create paths between IMiddleBox elements (Proxy and Cache) and stores the information about the ICN instances present in the system. In addition it contains the *IcnPacketProcessor* that is in charge of reactively inspect TCP packets directed to the HTTP port registered in the provider server call #4 from Table 6.1 coming into the SDN (produced by a OFPT_PACKET_IN message coming from the OpenFlow switch to the controller) and redirect them to the nearest proxy, this behavior can be overridden when registering the proxy with the 'Proactive' flag. In the case that 'Proactive' is selected, the paths between the nearest ports and the Proxies are created prior to data coming into the network, discharging from that burden to the controller and reducing the time for a request to be treated. Nevertheless, the 'Reactive' form of client-proxy path creation is set with high flow timeout to avoid flows from being discarded and recreated regularly, since it is based always on proximity and unless the proxy is removed or the client moves, the proxy to which a client is redirected is not foreseeable going to be changed.
The *IcnService* implements *IIcnPrivateService* that provides with the abstractions to be used by the REST API in Table 6.2 as well as the capability to issue path creation requests

for the prefetching engine (as an internal call and not a REST call since the controller here acts as the client, initiating the connection to the prefetcher). The *IIcnService* interface on the other hand offers the abstractions needed by the 6.1.
Lastly the *IcnService* reacts to Flow events issued by ONOS to identify expirations and remove the internal representations of the flows.
The *Resource* interface defines a common definition of what a Resource should be, thus enabling extensibility. *ResourceHTTP* stores the relation between the already requested URLs with the cache holding it and statistics such as how many times a resource has been requested. If the content was decided to be provided directly from the provider server, then the cache stored is the provider. Similarly, the *ResourceHTTPDASH* is implemented to store information specific to the DASH domain. It stores the URLs listed in the MPD for each representation. The *RepresentationDASH* is defined to store information related to the representation such as the frame size, the frame rate and the bandwidth among others. It also optionally (if the stream is H.264/SVC) the dependencies from other representations. One of the more important functionalities around these classes is the capability to map a URL with the ResourceDASH and retrieve which representation is it contained in, for H.264/SVC streams. This correspondence is employed by the prefetcher to be able to also prefetch the representations on which the actual URL depends.
The ICN instances are modelled after the *Icn* interface. With this approach, any ICN behavior can be easily implemented by inheritance. As an example, the *IcnClosestCache* implementation redirects the traffic to the Proxy's closest cache which, in turn, was the closest one to the client. That implementation is extended in the *IcnClosestCacheDASH*, which as its name implies, is designed to cope with DASH streams. This implementation checks the url of the resource being inserted into the *Icn* and if it is an MPD, it is downloaded and parsed in the *MPDParser* internal class and optimizations are applied, e.g.: prefetching the DASH chunks as introduced in Section 5.3.3.1 and implemented in *RepesentationPrefetcher* which in addition is able to prefetch H.264/SVC streams by analyzing the dependencies between layers (so that non-dependable layers are not requested) defined in the MPD.
Figure 6.2 shows the relation between Resources and Icn implementations and how the implementation evolved from the HTTP transport service corresponding to what was defined in section 5.3.2 to the DASH data aware service corresponding to 5.3.3.
As a consequence, the Prefetcher REST interface is defined as in Table 6.3, to enable the communication of the ICNaaS and the entity issuing the requests that will populate the cache prior to being requested. Note that in this case the ICNaaS acts as a client and not as the service. This functionality could have been implemented as part of the controller

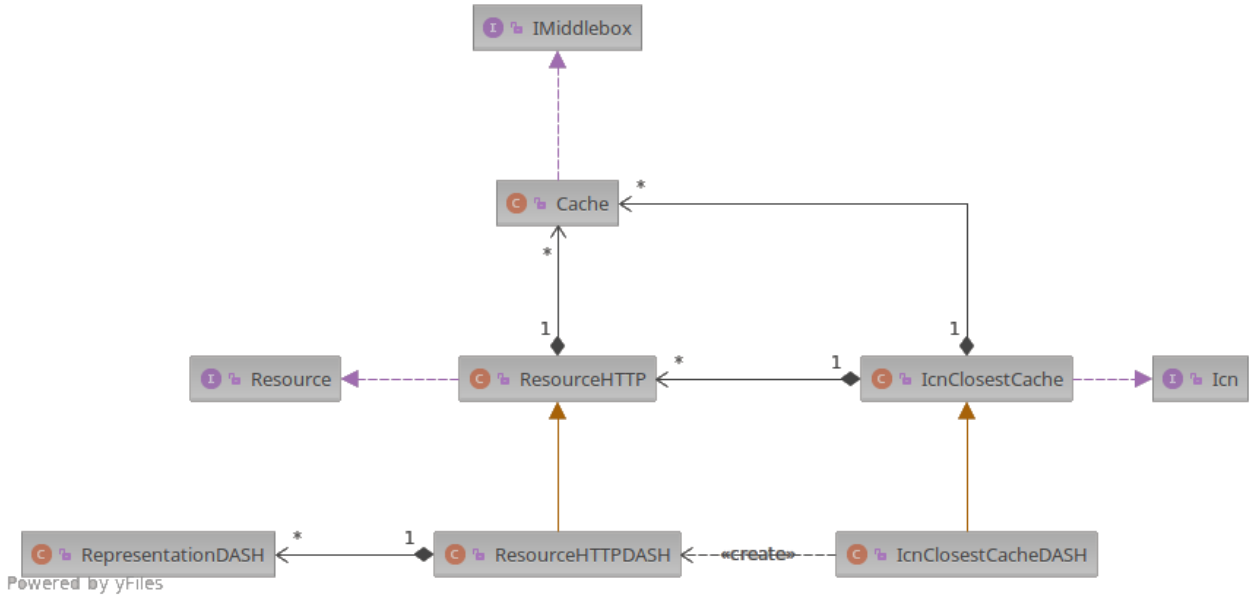


Figure 6.2: ICNaaS resource diagram.

#	Source	Destination	URL	Parameters	
1	ICNaaS	Prefetcher	/prefetch	uri, server, port	

Table 6.3: ICNaaS's prefetcher northbound interface.

by means of OFPT_PACKET_IN and OFPT_PACKET_OUT messages which would in turn imply at least 5 TCP messages (the 3-way handshake, the http request and the FIN). That approach would depend nevertheless on the caching entity behavior when receiving the FIN, if it still continues downloading on the server side, it would be fruitful, if not, it would be a waste. On the other hand, this process would take these 5 messages per chunk which would rise linearly in relation with the number of SVC layers desired. The prefetcher implements a full HTTP heap which means that issues requests identical to those issued by clients so that any caching system would be able to be used.

The ICNaaS application is developed as an ONOS app. The app relies on ONOS to calculate paths between the elements of the ICN architecture and to enforce the decisions into the network via the FlowObjective service. The FlowObjective service is an abstraction that avoids the application the need of being aware of what actually is the methodology to enforce the actions, meaning the Southbound provider being employed, such as if the protocol below is OpenFlow 1.0 or 1.3.

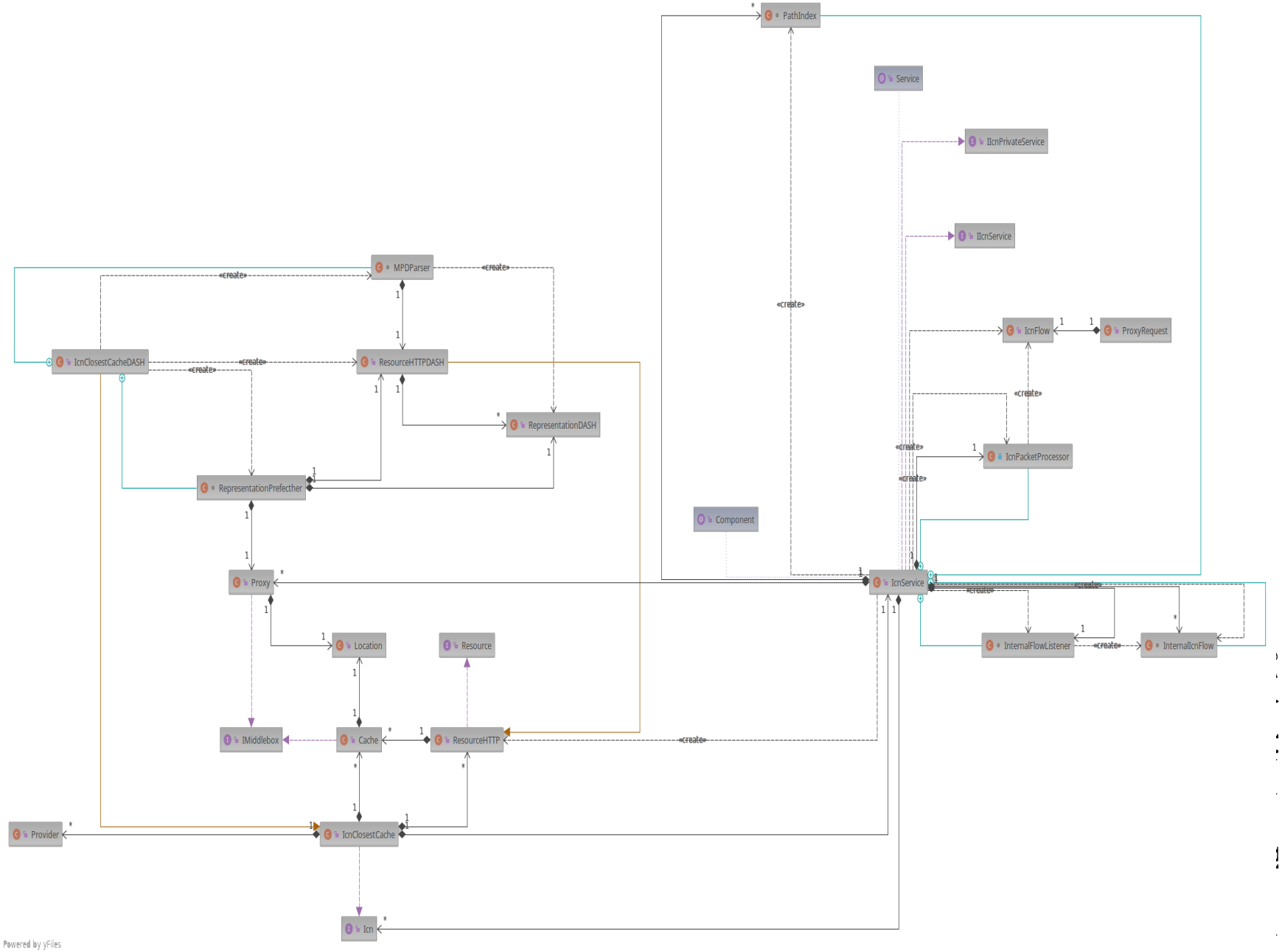


Figure 6.3: ICNaaS reduced core diagram.

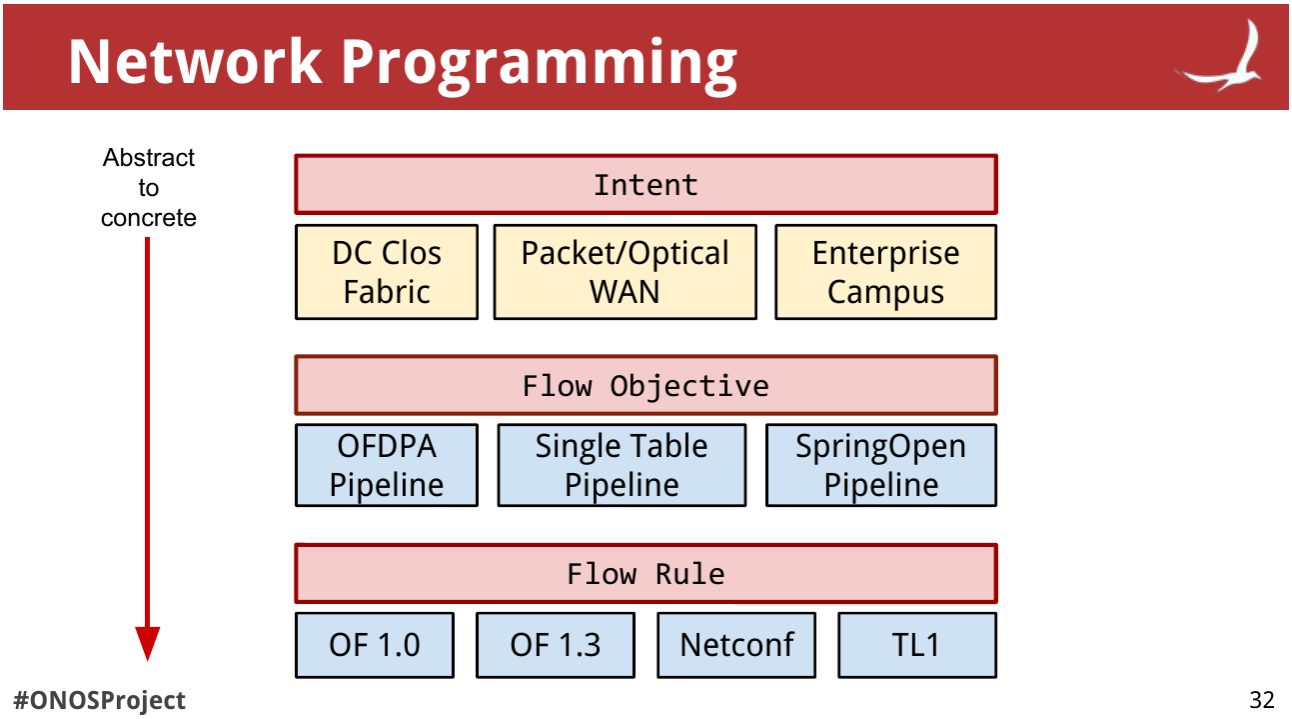


Figure 6.4: ONOS Abstraction.

6.1.2 Proxy

The proxy is in charge of the TCP Splicing as introduced in Section 5.3.2. Therefore client's requests are directed to the proxy. The proxy extracts the URL and notifies through the REST API using message #1 from Table 6.2. The controller creates the corresponding flows for adaptation.

In the first approach the proxy was implemented in python 2.6 and based on the *httplib* and *BaseHTTPServer* libraries.

For the second proof of concept a thorough implementation was done based on the *tornado* [123] python 3.6 framework. In addition, a *nginx* [124] proxy was deployed collocated with the python script to redirect connections locally to the python script. This way, incoming connections are managed by the *nginx* which is well known for its performance and scalability capabilities. Source code of the proxy can be found at `https://gitlab.atica.um.es/gn3plus/gn3proxy`.

6.1.3 Prefetcher

The prefetcher is the element in charge of transparently precaching content in the cache. It implements the REST API defined in Table 6.3. This element simply issues a HTTP GET request to the URI passed as an argument but using at the TCP/IP level the server and port as defined in the arguments so that the connection is steered by the channel created by the ICNaaS on top of the SDN. This approach makes the prefetching system backward compatible with any caching software or hardware.
For simplicity the prefetcher was deployed as a side entity of the proxy but it could be deployed anywhere into the network (even collocated with each cache). In that case the API in Table 6.2 would be extended with a new message to register the prefetcher independently removing the prefetch_port from message #2. The pyprefetcher is implemented in python also based on *tornado* and also taking profit (thanks to being collocated with the proxy) of the *nginx* Source code of pyprefetcher can be found at `https://gitlab.atica.um.es/jordi.ortiz.um.es/pyprefetcher`.
Some caching appliances or software might offer other means to explicitly indicate that a certain URI should be present in the cache. In that case, the prefetcher might make use of that method and in some cases the ICNaaS might itself contact the cache directly. Therefore, the cache should be also connected to 'ICN Control' ('ICN Control' is shown in Figure 5.1).

6.1.4 SVC Video Player

In order to automate the execution of the evaluation and in order to measure times per request, the 'umulibdashplayer' was played based on the libdash library. The client issues the chunk requests on time on forked processes and takes time and size measurement upon completion. The client receives as an argument the desired scalability level (Dependency Id in DASH nomenclature) and downloads the corresponding chunks as well as the dependencies without switching. Basically, the client side adaptation algorithm were trimmed from it. Source code can be found at `https://gitlab.atica.um.es/jordi.ortiz.um.es/umulibdashplayer`.

6.1.5 Video Sources

For the first evaluation implemented in floodlight on the feasibility of employing SDN to steer HTTP traffic on an ICN fashion, the need of eliminating the client side adaptation factor to compare objectively the executions. The fastest way to obtain the desired results

is to encapsulate just one representation into the DASH stream. To that end, the well known Big Buck Bunny H.264/AVC coded stream in 480p `http://plexp.inf.um.es/bitdash/bunny/` and 1080p `http://plexp.inf.um.es/bitdash/bunny1080p/` was encapsulated. In addition each chunk is limited to one second of video, the minimum encapsulation size allowed by the mp4box software.
For the evaluation of the final approach implemented on top of ONOS which included the use of SVC streams that employed the publicly available resources from Universität Klagenfurt Information Technology (`http://www-itec.uni-klu.ac.at/dash/?page_id=207`). In particular, the Big Buck Bunny video encoded with 50 scalability levels in 360p and 720p resolutions was used.

- `http://concert.itec.aau.at/SVCDataset/dataset/mpd-temp/BBB-I-360p.mpd`
- `http://concert.itec.aau.at/SVCDataset/dataset/mpd-temp/BBB-I-720p.mpd`
- `http://concert.itec.aau.at/SVCDataset/dataset/mpd-temp/BBB-I-1080p.mpd`

6.2 Floodlight evaluation

A first version of ICNaaS application was developed on top of Floodlight within the environment of the GN3Plus project. The application was intended to be integrated in the OpenNaaS framework which would be the one interacting with the ICNaaS REST interface.
In order to evaluate our proposal, an SDN based two cache system has been deployed (see Fig. 6.5) over University of Murcia campuses in the context of the Gaia Testbed [12] as part of the SmartFire [125] federation. The elements smartfire3 and gaia-cache act as caches. The gaia-proxy acts as proxy. Meanwhile smartfire2 is the streaming client and omf-gaia-1 acts as router to provide internet access to the SDN network. The controller is placed in ATICA building collocated with the video HTTP server (`http://plexp.inf.um.es`). The deployed caches are based in Squid 2.7 [126] and are configured as transparent [116] cache. All the switches represented in the scenario are based in OpenVSwitch 1.4.2 [127]. To perform the streaming the bitdash [128] javascript software was employed. Our SDN application has been built on top of the Floodlight controller [129] using OpenFlow protocol as a mean to modify the switches forwarding plane.

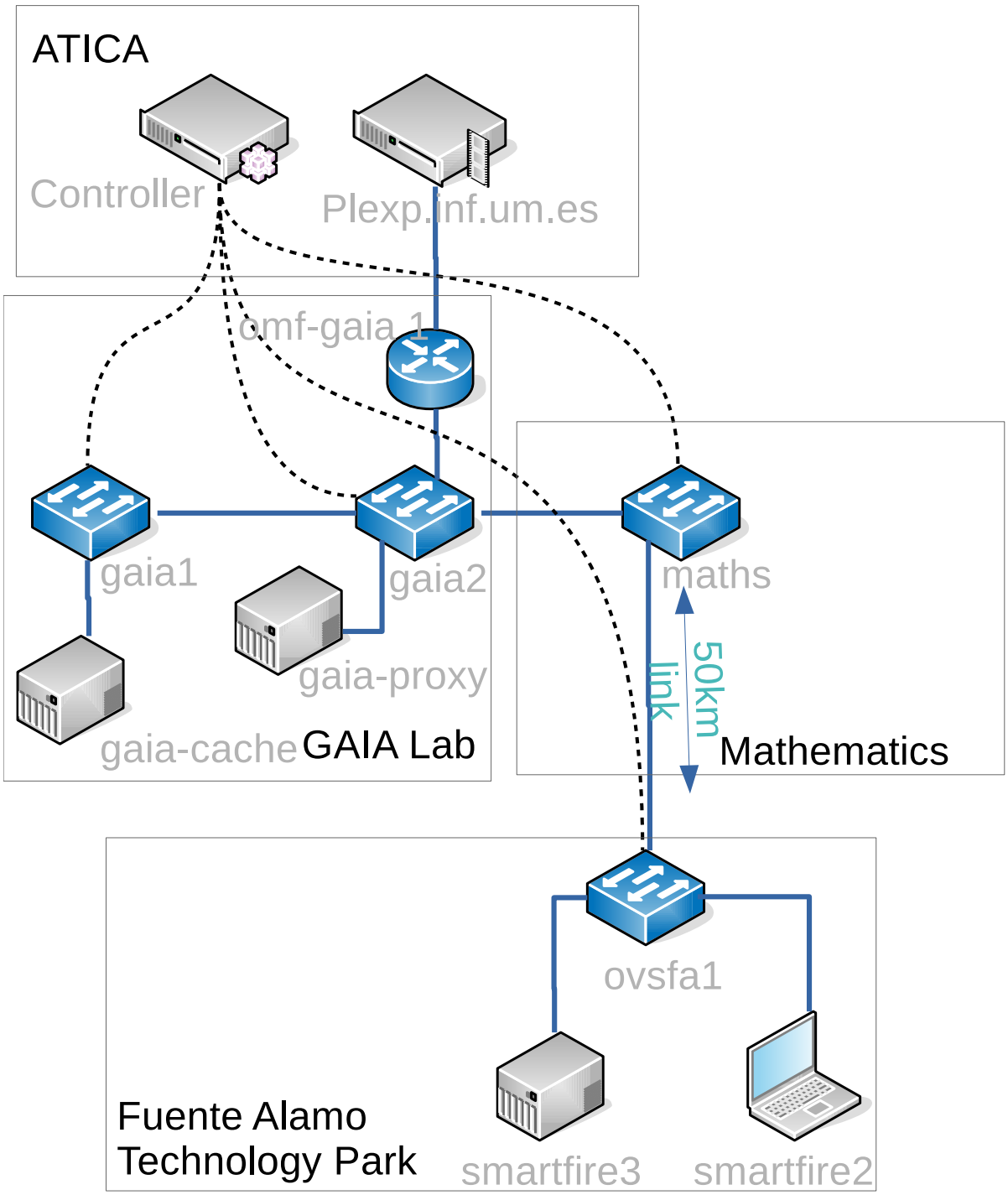


Figure 6.5: Floodlight Evaluation Scenario

The computers holding the OpenVSwitch are deployed in different university premises and are connected to a Network Operations Center (NOC) provided Virtual Local Area Network (VLAN) on top of which VTUN virtual tunnels are created providing direct connectivity between the OpenVSwitch instances.
The software developed and deployed to evaluate this publication is publicly accessible:

- *Floodlight cdn app*: `http://gitlab.atica.um.es/gn3plus/cdn-floodlight`
- *Proxy*: `http://gitlab.atica.um.es/gn3plus/gn3proxy` . The proxy implements iptables to redirect all the incoming connections regardless the ip address to the local proxy port.

First of all and as a basis for any conclusion the NOCACHE case has been tested, in this case the client retrieve video directly from the server. Then, a typical caching system, NOICN case, where client uses a predefined cache is tested. Finally, this paper's proposal or ICN case is tested. The scenario is always the same with the same caching software but in the NOCACHE and NOICN cases the SDN controller is programmed as a L2 learning switch.
Looking at Table 6.4 one can observe that the NOCACHE cases take longer while downloading less data than the cases in which caches are involved. The reason for such a difference is that the caching systems involved force pipelining allowing parallel chunk download. On the other hand, comparing the two caching system results, it is clear that there is no big negative impact in the streaming process when introducing the ICN approach and no negative optical influence was detected. Note that the ICN cases achieve higher download rates while duration is slightly higher than the NOICN cases due the initial delay produced by the proxy signalling through the REST API. Timing takes into account from TCP SYN message until the last TCP ACK while the DOWNRATE takes into account only incoming packages.
In order to have experimentation results non dependent neither from web browser nor from streaming library, another set of experiments was performed but downloading each chunk (including the mpd file) independently, not in a video streaming process. This approach also avoids the cloaking of any possible drawback of the solution caused by web browser caching or pipelining systems, meaning that the NOCACHE system is also using one TCP flow per chunk. Table 6.5 shows the statistics per chunk. As can be seen in AVERAGE column, representing the average time in seconds for a chunk to be retrieved in each scenario, the ICN cases mean an increase in time from 7% to 1026% but not exceeding in

any case more than 0,2 seconds per chunk, an acceptable value taking into account that the average size of a chunk is approximately 416K for 480p and 1.2M for 1080p which is a 1s video chunk. The best conclusion extracted is that for our ICN (0.41-4.608% enhancement in nearest cache case) the distance between cache and client is not relevant in comparison with the non-ICN case (80-246% enhancement in nearest cache case). The reason for this result comes from the northbound API signalling that becomes an important part of the consumed time per chunk. This results are for 1s video chunk which means that fine grain video scrolling is possible without downloading extra data and reducing power consumption. Bigger video chunks would decrease the relevance of the northbound API signalling while smaller video chunks would not have sense for human vision although they would influence caching fragmentation.
The TIMExBYTE column adds the chunk size as input element. Caching systems offer the poorest results in cache miss (empty cache) cases, while best results are achieved with the same systems in cache hit (filled up cache) cases. There is, as expected, also a slight performance decrease when introducing this paper solution in front of the NOICN caching but still producing enhancement over the NOCACHE case. This measurement is considered of relevance since chunk sizes differ due to variable birate video coding (VBR) and thus the effect in terms of TCP/HTTP overload differ as well as the burden of Northbound signalling.

6.3 ONOS evaluation

The first evaluation performed on the Gaia Testbed with OpenVSwitch demonstrated the feasibility of the solution, but, with the migration of the virtual OpenVswitch environment to a hardware based scenario with the acquisition of HPE Aruba 2920 switches, the limitations of the Floodlight controller and the arrival of new alternatives such as ONOS, triggered the decision of migrating the system to a new controller.
To get rid of the virtual tunnels, a new infrastructure was deployed on top of our own CWDM which was distributed among two buildings (ATICA and GaiaLab). Leveraging on VLAN tagging and configuring QinQ the links that interconnect the SDN devices were instantiated as shown in Figure 6.6a.
The software entities of the system (client, proxy, controller and prefetcher) virtually run on top of LibVirt managed commodity servers and employ VLAN tagging to reach the SDN devices. Since in this case we are not interested in introducing VLAN as part of the equation, the VLANs are removed by employing loops on the devices that convert access

	SCENARIO	SIZE (bytes)	DURATION (sec)	DOWNRATE (bps)
480p	NOCACHE	239.891.091	557,0853	3.440.749,24
	NOICN NEAREST CACHE EMPTY	272.818.063	557,4943	3.718.256,96
	NOICN NEAREST CACHE FULL	251.868.515	556,9431	3.589.009,28
	NOICN FURTHEST CACHE EMPTY	270.913.384	557,5884	3.712.513,85
	NOICN FURTHEST CACHE FULL	267.028.208	556,8303	3.495.248,51
	ICN NEAREST CACHE EMPTY	260.976.310	557,0979	3.747.654,28
	ICN NEAREST CACHE FULL	261.228.081	557,0549	3.751.559,80
	ICN FURTHEST CACHE EMPTY	261.424.570	557,1328	3.753.856,57
	ICN FURTHEST CACHE FULL	261.486.751	557,4061	3.752.908,06
1080p	NOCACHE	636.082.418	562,0779	9.049.156,84
	NOICN NEAREST CACHE EMPTY	788.450.118	558,8509	10.749.786,39
	NOICN NEAREST CACHE FULL	729.040.424	556,9834	10.236.077,81
	NOICN FURTHEST CACHE EMPTY	778.085.776	556,2790	10.691.572,16
	NOICN FURTHEST CACHE FULL	770.755.922	557,2209	10.641.239,69
	ICN NEAREST CACHE EMPTY	758.935.705	558,8510	10.770.887,16
	ICN NEAREST CACHE FULL	757.339.029	557,5036	10.867.575,12
	ICN FURTHEST CACHE EMPTY	759.486.883	558,0119	10.798.963,94
	ICN FURTHEST CACHE FULL	759.542.011	557,4960	10.899.336,58

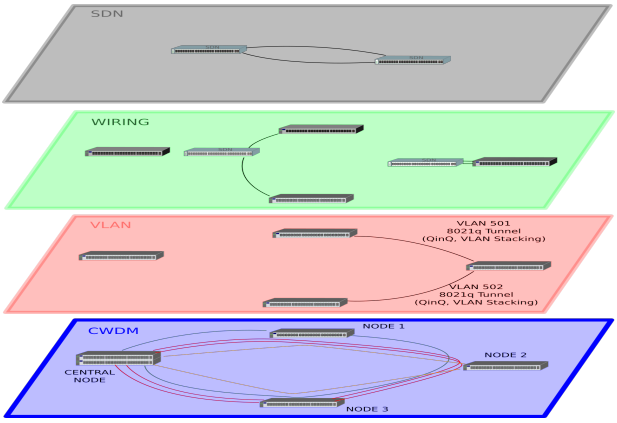
Table 6.4: Streaming Client Experimentation results

ports to nearby untagged ports, thus being presented to the switch as independent clients each one on an independent OpenFlow port. The scenario can be seen in Figure 6.6c and the ONOS' view of the scenario is shown in Figure 6.6b.
This evaluation not only aimed at achieving a deeper study on the effect of proxy location on the video streaming leveraging this proposal but also introducing the H.264/SVC factor in the equation. In addition, the original approach in which iptables was needed to make the packets directed to the server to be accepted by the proxy or the cache, was migrated to field rewrite inside the OpenFlow switches. To that end the results of the following milestones are presented in this section.

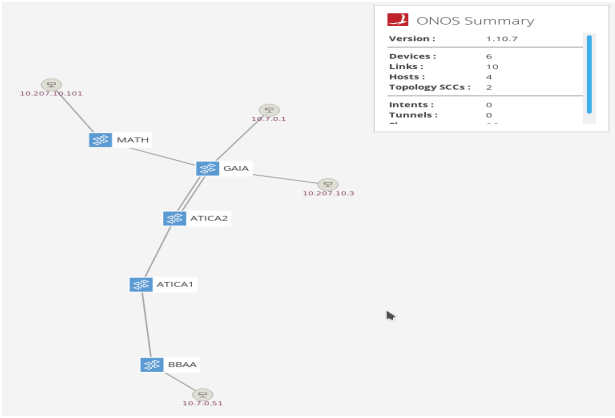
- Video resolution and scalability level. The testing was performed with video resolution in 360p, 720p and 1080p for dependency ids 1 , 2, 17 and 18 for the videos from Universtität Klagenfurt introduced in 6.1.5.
- Proxy location. In the same switch as the client and the furthest location related to the client.
- Cache empty and full

	SCENARIO	AVERAGE seconds	MEDIAN seconds	STDEVP seconds	MIN seconds	MAX seconds	TIME x BYTE seconds
480p	NOCACHE	0,1975	0,1738	0,1082	0,0154	1,6408	$4,7439 \cdot 10^{-7}$
	NOICN NEAREST CACHE EMPTY	0,2671	0,2344	0,1421	0,0101	1,5308	$6,4163 \cdot 10^{-7}$
	NOICN NEAREST CACHE FULL	0,0120	0,0075	0,0255	0,0017	0,3710	$0,2897 \cdot 10^{-7}$
	NOICN FURTHEST CACHE EMPTY	0,1000	0,0830	0,0633	0,0154	0,7995	$2,4009 \cdot 10^{-7}$
	NOICN FURTHEST CACHE FULL	0,0608	0,0546	0,0238	0,0072	0,1581	$1,4600 \cdot 10^{-7}$
	ICN NEAREST CACHE EMPTY	0,2482	0,2193	0,1007	0,0846	1,3182	$5,9612 \cdot 10^{-7}$
	ICN NEAREST CACHE FULL	0,1360	0,1207	0,0542	0,0433	0,3675	$3,2655 \cdot 10^{-7}$
	ICN FURTHEST CACHE EMPTY	0,2283	0,2021	0,1122	0,0583	1,6417	$5,4840 \cdot 10^{-7}$
	ICN FURTHEST CACHE FULL	0,1297	0,1164	0,0481	0,0405	0,3306	$3,1150 \cdot 10^{-7}$
1080p	NOCACHE	0,3679	0,3360	0,1617	0,0487	1,6119	$8,8369 \cdot 10^{-7}$
	NOICN NEAREST CACHE EMPTY	0,2472	0,2165	0,1049	0,0112	0,8622	$5,9375 \cdot 10^{-7}$
	NOICN NEAREST CACHE FULL	0,0409	0,0354	0,0234	0,0016	0,1461	$0,9821 \cdot 10^{-7}$
	NOICN FURTHEST CACHE EMPTY	0,2431	0,2078	0,1407	0,0321	1,4885	$5,8399 \cdot 10^{-7}$
	NOICN FURTHEST CACHE FULL	0,1416	0,1197	0,0698	0,0070	0,4391	$3,4001 \cdot 10^{-7}$
	ICN NEAREST CACHE EMPTY	0,5401	0,4647	0,2522	0,0684	2,5348	$12,9733 \cdot 10^{-7}$
	ICN NEAREST CACHE FULL	0,2934	0,2463	0,1427	0,0412	0,8355	$7,0464 \cdot 10^{-7}$
	ICN FURTHEST CACHE EMPTY	0,4375	0,3853	0,1977	0,1392	1,9769	$10,5097 \cdot 10^{-7}$
	ICN FURTHEST CACHE FULL	0,2946	0,2474	0,1451	0,0395	0,8520	$7,0753 \cdot 10^{-7}$

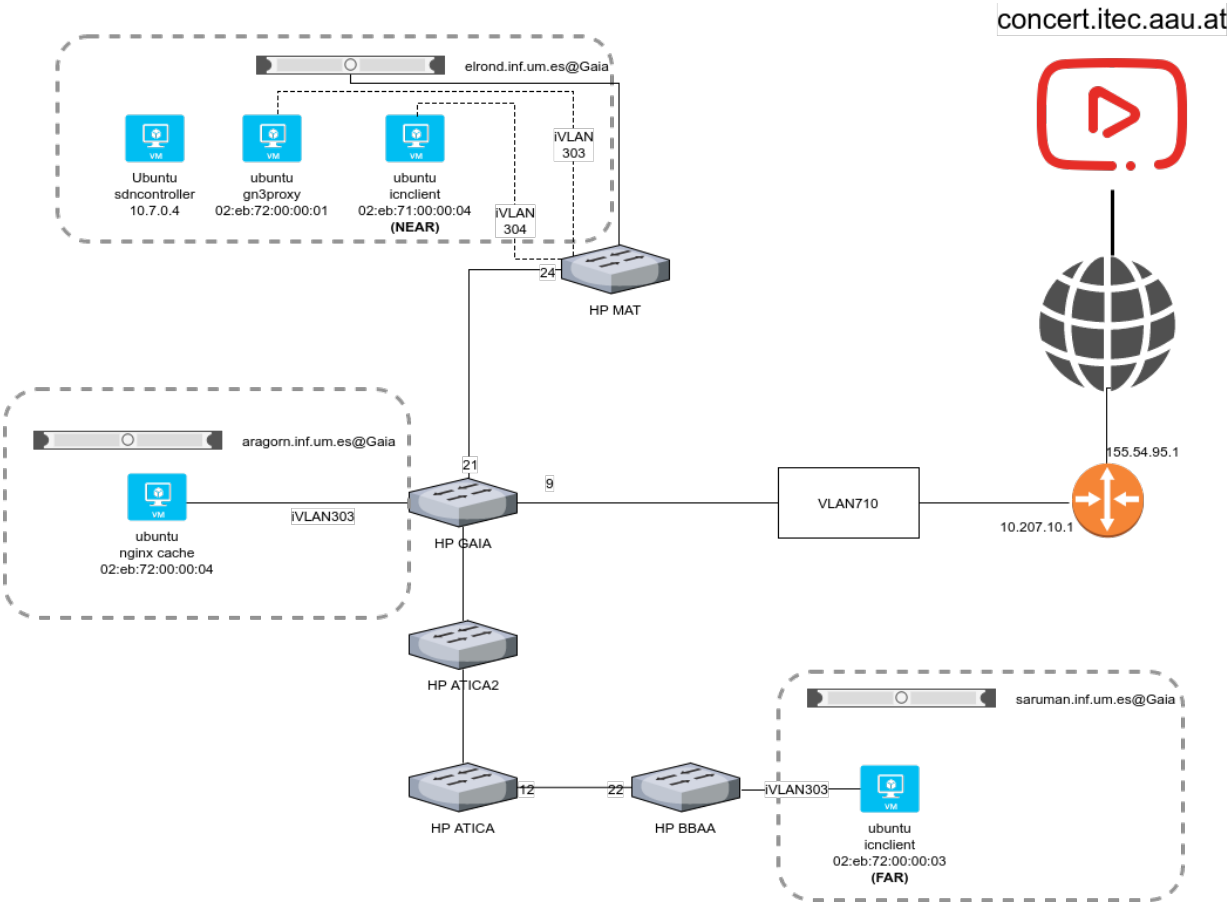
Table 6.5: Per Chunk Experimentation Results



(a) SDN over CWDM with QinQ



(b) ONOS Scenario view



(c) ONOS Scenario details

Figure 6.6: ONOS Evaluation Scenario

- Finally prefetching mechanism has been evaluated

The distance from the testbed to the video sources is shown in Listing 6.1 showing a tracepath to the server hosting the files.

```
 1:  gateway (155.54.95.1)                                          0.546ms
 2:  labredes-reservada.inf.um.es (155.54.210.254)                 21.205ms
 3:  firewall-lan.red.um.es (155.54.213.3)                          3.529ms
 4:  f_integra-lan.red.um.es (155.54.213.66)                        4.232ms
 5:  XE1-0-3-60.cica.rt1.and.red.rediris.es (130.206.194.53)  12.920ms
 6:  CICA.AE4.ciemat.rt1.mad.red.rediris.es (130.206.245.37)  24.745ms
 7:  CIEMAT.AE2.telmad.rt4.mad.red.rediris.es (130.206.245.2)  25.163ms
 8:  rediris.mx1.mar.fr.geant.net (62.40.124.192)                  43.147ms
 9:  ae3.mx1.mil2.it.geant.net (62.40.98.71)                       45.356ms
10:  ae5.mx1.vie.at.geant.net (62.40.98.38)                        60.353ms
11:  aconet-gw.mx1.vie.at.geant.net (62.40.124.2)                  56.673ms
12:  gigabitethernet9-1.klu1.aco.net (193.171.18.17)               67.056ms
13:  ubwcis.edvz.uni-klu.ac.at (193.171.18.18)                     66.576ms
```

Listing 6.1: Tracepath time from Gaia to Universität Klagenfurt where the videos are publicly hosted.

For this evaluation, the video player described in Section 6.1.4 was employed and the proxy's last version as described in 6.1.2 arranged as displayed in Figure 6.6c. Finally, to demonstrate the prefetching approach the pyprefetcher software described in 6.1.3.

In order to achieve repeatability and simplify the task of retrieving results from the testbed the nepi-ng [130] framework was employed, developed by INRIA in the onelab context with the experience gained during the OpenLab [9] project, which enables the automation of experiment execution and result gathering.

Script in Listing 6.2 launches the executions for the milestones detailed above. The reasons to have this script out of the NEPI's script are two, the need to control the initialization of the controller (Listing 6.3) that turned harder to be done in the python script and the use of timeout for each run so that if any of the components falls in an infinite loop, the execution of the rest of the experiments is not stopped. Take into account the duration of all the execution runs:

$2\ clients\ x\ 2\ frame\ sizes\ x\ 4\ svc\ layers\ x\ 2\ cache\ states\ x\ 20\ iterations\ =\ 640\ executions$

$640\ executions\ x\ 10\ \frac{min}{execution} = 6400\ min \approx 106 hours 40 minutes$

```
#!/bin/bash

onos_app_path=/home/nenjordi/icn_onos_app/
this_path=$PWD

for case in "CLIENT_NEAR" "CLIENT_FAR"
    do
        if [[ case="CLIENT_NEAR" ]]
        then
            clientip=10.207.0.52
        else
            clientip=10.207.0.51
        fi
        mkdir -p $case/CACHE_EMPTY/
        mkdir -p $case/CACHE_FULL/

        for size in "360p" "720p"
        do
            for dependency in 1 2 17 18
            do
                for iteration in $(seq -f '%05g' 1 20)
                do
                    date
                    echo "Launching $case($clientip) $size $dependency $iteration"
                    echo "Restarting Onos"
                    cd $onos_app_path
                    ./preparescenario.sh
                    cd $this_path
                    echo Running $case $cache $size $iteration\
                        iteration on dependency $dependency
                    cache="CACHE_EMPTY"
                    timeout 40m python ../src/emptycache.py \
    -u http://concert.itec.aau.at/SVCDataset/dataset/mpd-temp/BBB-I-"$size".mpd \
    -c $clientip -e 10.207.0.102 -p 10.207.0.101 \
    -d $dependency \
    -k "$case/$cache/$size"_"$iteration"_"$dependency"
                    timeout 40m python ../src/fullcache.py \
    -u http://concert.itec.aau.at/SVCDataset/dataset/mpd-temp/BBB-I-"$size".mpd \
    -c $clientip -e 10.207.0.102 -p 10.207.0.101 \
    -d $dependency \
    -k "$case/$cache/$size"_"$iteration"_"$dependency"
                    date
```

```
43                  done
44              done
45          done
46      done
47  done
```

Listing 6.2: Shell script to launch the scenarios

```
1  #!/bin/bash
2
3  ssh 10.7.0.4 -t sudo docker stop onos_phd
4  ssh 10.7.0.4 -t sudo docker rm onos_phd
5  ssh 10.7.0.4 -t sudo docker run -d -i -e KARAF_DEBUG=true \
6      -e ONOS_APPS=openflow --name onos_phd -p 6633:6633 -p 8181:8181 \
7      -p 8101:8101 -p 5005:5005 -p 8080:8080 onosproject/onos:1.10.2
8
9  nc -z 10.7.0.4 8101
10 while [ $? -ne 0 ]
11 do
12     echo "Wainting for 10.7.0.4 to become available"
13     sleep 60
14     nc -z 10.7.0.4 8101
15 done
16 # install app
17 curl -sS --user karaf:karaf --noproxy localhost -X POST -HContent-Type:application/octet
18 sleep 15
19 /home/nenjordi/icn_onos_app/icncreationitecuniklu.sh
```

Listing 6.3: Shell script to deploy onos with the ICNaaS application and register the scenario

Listing 6.4 shows the NEPI script that arranges and synchronizes the different elements of the experiment. Four schedulers are defined, installation, launch, stop and pull data. Installation is in charge of launching the different components involved in each run and cleaning any previous state if needed. Launch, launches the streaming client right after previous phase finalization. Stop, is obviously in charge of stopping services when necessary and finally Pull Data phase recovers the log files from the different computers to apply further analysis. The correspondent execution for full cache case, needs to change the commands in line 71 to stop squid, remove the squid log and start squid again which produces a clean log file for later analysis hence using disk cache to support the relaunch of the service and still having the data available for following requests.

```
#!/usr/bin/env python3

from asynciojobs import Scheduler, Job

from apssh import SshNode, SshJob, Pull, Run, RunScript

from optparse import OptionParser
import os

usage = ("usage: %prog -u <url> -d <dependencyid> -k <keyexp>  -c <clientip> -e <cacheip

parser = OptionParser(usage = usage)
parser.add_option("-u", "--url", dest="url",
        help="DASH URL", type="str")
parser.add_option("-d", "--dependencyid", dest="depid",
        help="Scalability layer to download", type="str")
parser.add_option("-k", "--keyexp", dest="keyexp",
        help="Key to be used as key for values such as output path", type="str")
parser.add_option("-c", "--client", dest="clientip",
        help="Client IP address", type="str")
parser.add_option("-e", "--cache", dest="cacheip",
        help="Cache IP address", type="str")
parser.add_option("-p", "--proxy", dest="proxyip",
        help="Proxy IP address", type="str")
(options, args) = parser.parse_args()

url = options.url
depid = options.depid
keyexp = options.keyexp
clientip = options.clientip
cacheip = options.cacheip
proxyip = options.proxyip

if not os.path.exists(keyexp):
        os.makedirs(keyexp)

verbose_ssh = False


##########
# Create a scheduler
installationscheduler = Scheduler()
```

```
launchscheduler = Scheduler()
stopscheduler = Scheduler()
pulldatascheduler = Scheduler()

##########
# Video streaming client
client = SshNode(hostname = clientip, username = 'nenjordi',
                  verbose = verbose_ssh)


##########
# Proxy
proxy = SshNode(hostname = proxyip, username = 'nenjordi',
                  verbose = verbose_ssh)

##########
# Cache
cache = SshNode(hostname = cacheip, username = 'root',
                  verbose = verbose_ssh)

##########
# Jobs
cachecleanjob = SshJob(
    # on what node do we want to run this:
    node = cache,
    # assign the scheduler
    scheduler = installationscheduler,
    # what to run
    commands = [
            Run('systemctl stop squid'),
            Run('rm /var/log/squid/* -f'),
            Run('rm /var/spool/squid/* -rf'),
            Run('squid -z'),
            Run('systemctl start squid'),
    ],
)

proxystartjob = SshJob(
    # on what node do we want to run this:
    node = proxy,
    # assign the scheduler
    scheduler = installationscheduler,
    # what to run
```

```
86      commands = [
87              Run('cd /home/nenjordi/gn3proxy'),
88              Run('/home/nenjordi/gn3proxy/stopproxy.sh'),
89              Run('/home/nenjordi/gn3proxy/launchproxy.sh')
90      ],
91  )
92
93  clientjob = SshJob(
94      # on what node do we want to run this:
95      node = client,
96      # assign the scheduler
97      scheduler = launchscheduler,
98      # what to run
99      command = [ '/home/nenjordi/umulibdashplayer/libdash/bin/umuplayer', url, depid,\
100         '|',  'tee', '/tmp/client.log' ],
101 )
102
103
104 proxystopjob = SshJob(
105     # on what node do we want to run this:
106     node = proxy,
107     # assign the scheduler
108     scheduler = stopscheduler,
109     # what to run
110     command = [ '/home/nenjordi/gn3proxy/stopproxy.sh' ],
111 )
112
113
114 # Pull data jobs
115 pullclientjob = SshJob (
116     node = client,
117     scheduler = pulldatascheduler,
118     commands = [
119                 Run("echo Recovering client data from $(hostname)"),
120                 Pull('/tmp/client.log', keyexp + '/client.log')
121                ],
122 )
123
124 pullproxyjob = SshJob (
125     node = proxy,
126     scheduler = pulldatascheduler,
127     commands = [
128                 Run("echo Recovering proxy data from $(hostname)"),
```

```
129                  Pull('/home/nenjordi/gn3proxy/icnproxy.log', keyexp + '/icnproxy.log')
130                 ],
131 )
132
133 pullcachejob = SshJob (
134     node = cache,
135     scheduler = pulldatascheduler,
136     commands = [
137                  Run("echo Recovering cache data from $(hostname)"),
138                  Pull('/var/log/squid/access.log', keyexp + '/squidaccess.log'),
139                 ],
140 )
141
142
143 # run the scheduler
144 ok = installationscheduler.orchestrate()
145 # give details if it failed
146 ok or installationscheduler.debrief()
147
148 ok = launchscheduler.orchestrate()
149 ok or installationscheduler.debrief()
150
151 ok = stopscheduler.orchestrate()
152 ok or stopscheduler.debrief()
153
154 ok = pulldatascheduler.orchestrate()
155 ok or pulldatascheduler.debrief()
156
157 # return something useful to your OS
158 exit(0 if ok else 1)
159
```

Listing 6.4: NEPI script to execute each experiment with empty cache and retrieve the output

6.3.1 Experimentation ICNaaS Results

To present an overview of the experimentation results, Tables 6.6 and 6.7 present a summary of the results obtained from streaming the three different frame size (360p, 720p and 1080p in column *Size*) for four different scalability layers (1, 2, 17 and 18 in column *Layer*) with the cache both empty and full (column *cache*) from the client located

(column *Position*) "far" connected to the BBAA switch as shown in Figure 6.6c and "near" being collocated with the proxy at MAT switch. For each combination, twenty streaming processes have been sequentially carried out of which this data is representing only the successful ones (meaning that the client didn't abort the transmission) being represented in the last column (#) of the tables. For each execution the mean download time per chunk employed by the client is calculated and for the means of all the combination's execution the mean (μ *time(s)*), standard deviation (σ), minimum (*Min*) and maximum (*Max*) values are shown.

In general the mean download time increases in relation with the number of layers and the frame size and is reduced for the same values when the cache is full. The distance to the proxy is clearly also affecting the mean download time. There is just an exception on the observed values present in results for nearer client with cache empty with a frame size of 360p and scalability layer 2 which is marked in Table 6.6. It is clear that some of the executions have produced spikes just looking at the standard deviation or at the max value for the row, in particular experiment number 1 has a mean of 3.9118 being the maximum and experiment number 14 has a value of 3.4244 as can be seen in Figure 6.7. There is no direct relation between the spurious events in each of the cases, while experiment number 1 has several values over 10 seconds in mid of the streaming process, experiment number 14 has the spurious events near the end of the process actually delaying the end of the streaming.

Another event to be examined is the high number of failures for some of the cases also highlighted in the tables in row #. All these cases are due a failure in the download of the mpd file, therefore aborting the full execution of the case. Since in most cases the connectivity fails for both the empty cache and the full cache cases which are run sequentially thus can be assumed that it is a general connectivity problem and not a problem in the TCPPacketProcessor or in any of the flows installed on the devices. In addition these cases always occur in the time period between 00:00am and 05:00am which is the time in which servers perform their backups for our lab and also for university central offices, so although not sure it is likely that these series of experiments were affected by those events. To what extent these experiments are influenced by the backups can be easily accounted by the fact that the entities involved in them are virtual machines that are backed up by the same backups.

A key factor for reactive SDN designs is the time spent in controller interaction. In this case, the call to the northbound by the proxy is measured in contrast with the time spent by the transaction from the proxy point of view without taking into account the communication with the client.

Position	Cache	Size	Layer	μ time(s)	σ	Min	Max	#
NEAR	EMPTY	360p	1	0.5566	0.0986	0.4575	0.8349	18
NEAR	EMPTY	360p	2	0.9822	0.9050	0.6192	3.9118	20
NEAR	EMPTY	360p	17	0.8640	0.0281	0.8331	0.9353	18
NEAR	EMPTY	360p	18	1.3140	0.0304	1.2439	1.3761	20
NEAR	EMPTY	720p	1	0.7200	0.0458	0.6501	0.8655	20
NEAR	EMPTY	720p	2	0.9263	0.0666	0.8478	1.0834	19
NEAR	EMPTY	720p	17	1.4491	0.0467	1.3778	1.5471	20
NEAR	EMPTY	720p	18	4.0224	0.1419	3.7391	4.2124	13
NEAR	EMPTY	1080p	1	2.0263	0.1212	1.8678	2.3150	20
NEAR	EMPTY	1080p	2	5.5988	1.2927	3.1918	8.0564	17
NEAR	EMPTY	1080p	17	23.9344	4.9793	16.2921	36.0098	14
NEAR	EMPTY	1080p	18	57.8806	1.9719	54.9597	61.8601	17
NEAR	FULL	360p	1	0.3194	0.0340	0.2498	0.3722	19
NEAR	FULL	360p	2	0.4915	0.0247	0.4499	0.5484	19
NEAR	FULL	360p	17	0.7339	0.0350	0.6642	0.8016	19
NEAR	FULL	360p	18	1.2505	0.0297	1.2051	1.3082	19
NEAR	FULL	720p	1	0.4557	0.0391	0.3823	0.5229	20
NEAR	FULL	720p	2	0.7383	0.0561	0.6569	0.9055	19
NEAR	FULL	720p	17	1.2357	0.0723	1.1338	1.3917	20
NEAR	FULL	720p	18	3.7616	0.1379	3.5371	4.0348	13
NEAR	FULL	1080p	1	0.8795	0.0591	0.8070	1.0634	20
NEAR	FULL	1080p	2	1.7749	0.1818	1.5298	2.0564	17
NEAR	FULL	1080p	17	3.0292	0.2262	2.7690	3.6320	14
NEAR	FULL	1080p	18	38.5018	5.7604	35.8250	56.7763	19

Table 6.6: Summary client near results

FAR	EMPTY	360p	1	0.5963	0.0750	0.5223	0.8973	20
FAR	EMPTY	360p	2	0.7341	0.0546	0.6715	0.8942	20
FAR	EMPTY	360p	17	0.9997	0.0349	0.9447	1.0501	19
FAR	EMPTY	360p	18	1.5339	0.0309	1.4774	1.5873	20
FAR	EMPTY	720p	1	0.9612	0.0684	0.8645	1.1555	20
FAR	EMPTY	720p	2	1.1508	0.0919	1.0406	1.3656	20
FAR	EMPTY	720p	17	1.8513	0.0870	1.7252	2.0419	20
FAR	EMPTY	720p	18	3.4871	0.2373	3.2324	4.0027	20
FAR	EMPTY	1080p	1	2.9213	0.3621	2.4022	3.6007	18
FAR	EMPTY	1080p	2	5.9315	1.3266	4.8449	10.5962	20
FAR	EMPTY	1080p	17	23.9077	4.8543	15.2383	30.7734	18
FAR	EMPTY	1080p	18	58.4258	1.4205	55.3091	61.1419	20
FAR	FULL	360p	1	0.3945	0.0468	0.3203	0.5220	16
FAR	FULL	360p	2	0.5615	0.0322	0.5182	0.6190	15
FAR	FULL	360p	17	0.8958	0.0285	0.8291	0.9394	18
FAR	FULL	360p	18	1.4229	0.0447	1.3605	1.5347	17
FAR	FULL	720p	1	0.6900	0.0668	0.5699	0.8210	17
FAR	FULL	720p	2	0.9457	0.0889	0.8361	1.1786	20
FAR	FULL	720p	17	1.6825	0.0896	1.5422	1.8934	18
FAR	FULL	720p	18	3.2933	0.1911	3.0086	3.7977	16
FAR	FULL	1080p	1	1.5112	0.0711	1.3781	1.6486	13
FAR	FULL	1080p	2	2.9461	0.3364	2.5351	3.7240	18
FAR	FULL	1080p	17	5.0242	0.3315	4.2775	5.6981	16
FAR	FULL	1080p	18	41.0001	1.0068	39.6597	43.8372	18

Table 6.7: Summary client far results

Position	Cache	Size	Layer	μ full (s)	σ	Min	Max	μ ctrl (s)	σ	Min	Max	#
NEAR	EMPTY	360p	1	0.5125	0.0989	0.4140	0.7902	0.0073	0.0001	0.0070	0.0075	18
NEAR	EMPTY	360p	2	0.8737	0.7480	0.5680	3.2954	0.0070	0.0001	0.0068	0.0072	20
NEAR	EMPTY	360p	17	0.7932	0.0275	0.7633	0.8656	0.0066	0.0002	0.0064	0.0070	18
NEAR	EMPTY	360p	18	1.1848	0.0283	1.1287	1.2421	0.0065	0.0001	0.0063	0.0068	20
NEAR	EMPTY	720p	1	0.6271	0.0456	0.5566	0.7716	0.0074	0.0002	0.0069	0.0078	20
NEAR	EMPTY	720p	2	0.8071	0.0603	0.7312	0.9383	0.0071	0.0001	0.0069	0.0073	19
NEAR	EMPTY	720p	17	1.2099	0.0349	1.1519	1.2580	0.0069	0.0002	0.0066	0.0073	20
NEAR	EMPTY	720p	18	2.3490	0.0431	2.2520	2.3914	0.0068	0.0002	0.0065	0.0073	13
NEAR	EMPTY	1080p	1	1.5170	0.0687	1.4089	1.6564	0.0069	0.0001	0.0067	0.0071	20
NEAR	EMPTY	1080p	2	2.8888	0.3274	2.0161	3.3144	0.0066	0.0003	0.0063	0.0072	17
NEAR	EMPTY	1080p	17	5.9049	0.1678	5.4374	6.1779	0.0055	0.0001	0.0053	0.0058	14
NEAR	EMPTY	1080p	18	4.7933	0.0744	4.5979	4.8915	0.0051	0.0002	0.0050	0.0058	17
NEAR	FULL	360p	1	0.2752	0.0337	0.2034	0.3271	0.0052	0.0001	0.0050	0.0055	19
NEAR	FULL	360p	2	0.4422	0.0246	0.4014	0.4988	0.0048	0.0002	0.0045	0.0050	19
NEAR	FULL	360p	17	0.6662	0.0340	0.6024	0.7321	0.0048	0.0002	0.0045	0.0053	19
NEAR	FULL	360p	18	1.1204	0.0288	1.0820	1.1767	0.0046	0.0001	0.0044	0.0049	19
NEAR	FULL	720p	1	0.3659	0.0383	0.2924	0.4325	0.0053	0.0001	0.0050	0.0055	20
NEAR	FULL	720p	2	0.6132	0.0401	0.5448	0.7166	0.0048	0.0001	0.0046	0.0052	19
NEAR	FULL	720p	17	1.0188	0.0585	0.9326	1.1364	0.0049	0.0001	0.0047	0.0051	20
NEAR	FULL	720p	18	2.2027	0.0418	2.1451	2.2801	0.0048	0.0001	0.0045	0.0050	13
NEAR	FULL	1080p	1	0.6772	0.0546	0.6131	0.8521	0.0050	0.0001	0.0048	0.0054	20
NEAR	FULL	1080p	2	1.3727	0.1411	1.1703	1.5726	0.0046	0.0001	0.0043	0.0048	17
NEAR	FULL	1080p	17	2.1376	0.0978	2.0168	2.3800	0.0046	0.0001	0.0044	0.0047	14
NEAR	FULL	1080p	18	4.5923	0.0576	4.5520	4.8155	0.0042	0.0003	0.0041	0.0051	19

Table 6.8: Summary controller results

Position	Cache	Size	Layer	μ full (s)	σ	Min	Max	μ ctrl (s)	σ	Min	Max	#
FAR	EMPTY	360p	1	0.4545	0.0740	0.3854	0.7534	0.0073	0.0002	0.0071	0.0076	20
FAR	EMPTY	360p	2	0.5704	0.0484	0.5222	0.7324	0.0071	0.0001	0.0068	0.0073	20
FAR	EMPTY	360p	17	0.7854	0.0261	0.7335	0.8242	0.0069	0.0001	0.0066	0.0072	19
FAR	EMPTY	360p	18	1.1913	0.0202	1.1498	1.2297	0.0067	0.0002	0.0065	0.0072	20
FAR	EMPTY	720p	1	0.6331	0.0552	0.5635	0.7847	0.0073	0.0001	0.0071	0.0076	20
FAR	EMPTY	720p	2	0.8320	0.0933	0.7323	1.0584	0.0064	0.0001	0.0062	0.0066	20
FAR	EMPTY	720p	17	1.2940	0.0664	1.2034	1.4398	0.0063	0.0001	0.0062	0.0065	20
FAR	EMPTY	720p	18	2.1258	0.0667	2.0228	2.2962	0.0064	0.0002	0.0061	0.0071	20
FAR	EMPTY	1080p	1	1.6132	0.1664	1.3964	2.0449	0.0069	0.0001	0.0067	0.0071	18
FAR	EMPTY	1080p	2	2.9218	0.3896	2.4507	4.2046	0.0068	0.0001	0.0065	0.0070	20
FAR	EMPTY	1080p	17	5.9263	0.1846	5.5105	6.1192	0.0057	0.0002	0.0055	0.0063	18
FAR	EMPTY	1080p	18	4.8560	0.0535	4.7451	4.9392	0.0052	0.0001	0.0050	0.0053	20
FAR	FULL	360p	1	0.2526	0.0470	0.1788	0.3828	0.0052	0.0001	0.0051	0.0054	16
FAR	FULL	360p	2	0.4024	0.0280	0.3479	0.4638	0.0048	0.0001	0.0046	0.0051	15
FAR	FULL	360p	17	0.6847	0.0243	0.6232	0.7385	0.0049	0.0001	0.0047	0.0052	18
FAR	FULL	360p	18	1.0951	0.0351	1.0440	1.1870	0.0049	0.0002	0.0046	0.0052	17
FAR	FULL	720p	1	0.3792	0.0601	0.2803	0.5001	0.0054	0.0001	0.0051	0.0056	17
FAR	FULL	720p	2	0.6331	0.0921	0.5468	0.8658	0.0045	0.0001	0.0042	0.0046	20
FAR	FULL	720p	17	1.1341	0.0545	1.0600	1.2380	0.0045	0.0001	0.0044	0.0046	18
FAR	FULL	720p	18	2.0029	0.0510	1.9217	2.1403	0.0046	0.0001	0.0044	0.0051	16
FAR	FULL	1080p	1	0.8040	0.0353	0.7432	0.8818	0.0049	0.0001	0.0047	0.0052	13
FAR	FULL	1080p	2	1.6489	0.1318	1.4554	1.9139	0.0046	0.0001	0.0044	0.0048	18
FAR	FULL	1080p	17	2.6838	0.1637	2.4095	3.0885	0.0047	0.0003	0.0044	0.0056	16
FAR	FULL	1080p	18	4.6805	0.0380	4.6282	4.7765	0.0041	0.0001	0.0040	0.0043	18

Table 6.9: Summary controller far results

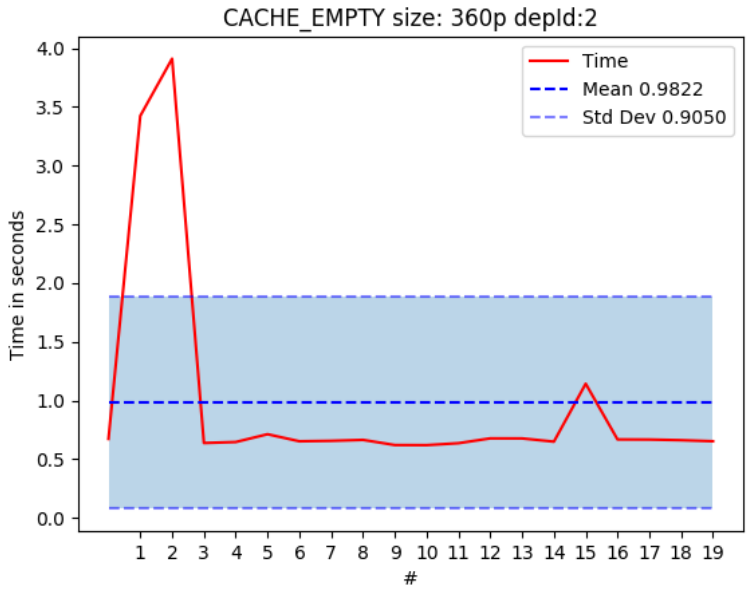


Figure 6.7: Client near, cache empty, 360p, layer 2 mean dowload time value for each succesful experiment

Tables 6.8 and 6.9 show the results from the perspective of the proxy, showing the mean time spent for the full download ($\mu full(s)$) from the point of view of the proxy, meaning that timer is started when the request from the client arrives to the proxy and is stopped when the cache has returned the data, and the mean controller time ($\mu ctrl(s)$) or how much time the REST interface with the controller is spent. Although the last value includes the calculations for which cache to be employed or if the element is already cached among others, it has to be taken into account that ONOS makes use of internal stores that take the flow changes immediately but apply them afterwards on the network with a small delay. Nevertheless, it can be easily seen that the time spent is not scaling as rapidly as for Tables 6.6 and 6.7 which means that most of the time is spent on the link between the proxy and the client, in addition, it is clear that the time spent by the REST interface is almost despicable.

6.3.2 Analyzing the layering problems

It can be observed that the performance of the system is degraded when the number of layers increases. There is no explanation in terms of algorithms for such degradation since each HTTP request is an independent event and there is no indicative that the controller, neither the cache were introducing extra delay. It seems that the reason for

such degradation is indeed related to the H.264/SVC layers involved in the experiment but due to the fact that the switches are running OpenFlow 1.3 which is run partly on software on the device and not by the video structure itself. To ensure that this claim is true, an analysis was made and an example of such a case is shown in detail.
The situation analyzed and shown in detail corresponds to the execution on a client collocated on the same switch with the proxy (near case), one jump away from the cache which in this case was full mitigating therefore the impact of the Internet connection, streaming the 360p version of the video and dependency id 18 (meaning that layers 18, 17, 2, 1 and 0 need to be downloaded for each chunk). In particular this corresponds to the sixth iteration out of 20 for this particular use case. Table 6.10 shows the time spent for each data chunk in the moment in which the degradation is perceived. Layers of segment 277 take in average more than 2 seconds to be downloaded. Taking into account that 2 seconds is a proper caching time for DASH video clients this would mean that this chunks would arrive late to the decoding and either would be discarded either would stop the decoding process performing frozenness to the video play. Table 6.11 shows the corresponding timing in the squid cache and as can be observed, the increase is not related to the cache miss/hit case (although all the chunks have been previously cached, squid has its own eviction policies which can cause cache miss events), as a particular example seg279-L2 needs 2 ms for the cache while the result is over 1.4 seconds.
The next element to be analyzed while looking for explanations is the sdn controller and the time taken to serve the proxy request (the flow creation contained in that time). The mean for the controller request is of 0.00466778 seconds (as shown in Figure 6.8), while the full time (including the time to recover the data from the cache) is of 1.10853. The conclusion is that the data plane is introducing the delay. And finally taking a look at Figure 6.9, that presents the CPU usage of the network nodes recovered with SNMP and dumped on a InfluxDB which in turn is accessed by Kibana for plotting the figure, can be observed that at 17:55:30 there is a peak of 100% cpu corresponding to the analyzed moment. The timestamps of the three elements shown in the study are partly synchronized by Network Time Protocol (NTP) and even though it is not precise to microsecond, it is accurate enough to correlate the information and describe the problem.
In addition to that particular case, it can be seen thorough the observation of the effect of a complete series of twenty executions on each client how the cpu of the switches is used and therefore its effects on Tables 6.6 and 6.7 last rows (highlighted in gray). Figure 6.10a and Figure 6.10b show the cpu usage for the client's switch collocated with the proxy (near scenario and tagged as MAT) and the client's switch placed on the other end of the scenario(far scenario tagged as BBAA) as well as the switch bordering with the router

TimeStamp	Download Time	Data chunk	size
Wed Dec 20 17:55:28 2017	0.315126 sec	360p/BBB-I-360p.seg278-L18.svc	38773bytes
Wed Dec 20 17:55:29 2017	1.7521 sec	360p/BBB-I-360p.seg277-L17.svc	55108bytes
Wed Dec 20 17:55:29 2017	1.82293 sec	360p/BBB-I-360p.seg277-L2.svc	69702bytes
Wed Dec 20 17:55:29 2017	2.01693 sec	360p/BBB-I-360p.seg277-L1.svc	101344bytes
Wed Dec 20 17:55:29 2017	2.20274 sec	360p/BBB-I-360p.seg277-L16.svc	133002bytes
Wed Dec 20 17:55:29 2017	2.37001 sec	360p/BBB-I-360p.seg277-L0.svc	149206bytes
Wed Dec 20 17:55:30 2017	1.73863 sec	360p/BBB-I-360p.seg278-L17.svc	45568bytes
Wed Dec 20 17:55:30 2017	1.77656 sec	360p/BBB-I-360p.seg278-L2.svc	48789bytes
Wed Dec 20 17:55:30 2017	1.94013 sec	360p/BBB-I-360p.seg278-L1.svc	73112bytes
Wed Dec 20 17:55:30 2017	2.07093 sec	360p/BBB-I-360p.seg278-L16.svc	106739bytes
Wed Dec 20 17:55:30 2017	2.20616 sec	360p/BBB-I-360p.seg278-L0.svc	153626bytes
Wed Dec 20 17:55:31 2017	1.21208 sec	360p/BBB-I-360p.seg279-L18.svc	19456bytes
Wed Dec 20 17:55:32 2017	0.0819143 sec	360p/BBB-I-360p.seg280-L18.svc	13750bytes
Wed Dec 20 17:55:32 2017	1.41701 sec	360p/BBB-I-360p.seg279-L2.svc	23061bytes
Wed Dec 20 17:55:32 2017	1.5106 sec	360p/BBB-I-360p.seg279-L17.svc	33973bytes
Wed Dec 20 17:55:32 2017	1.61031 sec	360p/BBB-I-360p.seg279-L1.svc	54744bytes
Wed Dec 20 17:55:33 2017	1.86638 sec	360p/BBB-I-360p.seg279-L16.svc	96010bytes
Wed Dec 20 17:55:33 2017	1.95353 sec	360p/BBB-I-360p.seg279-L0.svc	118385bytes
Wed Dec 20 17:55:33 2017	1.2821 sec	360p/BBB-I-360p.seg280-L2.svc	11132bytes
Wed Dec 20 17:55:33 2017	1.53678 sec	360p/BBB-I-360p.seg280-L17.svc	32539bytes
Wed Dec 20 17:55:33 2017	1.55224 sec	360p/BBB-I-360p.seg280-L1.svc	33484bytes
Wed Dec 20 17:55:34 2017	1.89187 sec	360p/BBB-I-360p.seg280-L0.svc	96057bytes

Table 6.10: Detail of client logs in which the performance is degraded.

TimeStamp	Time(ms)	cache result	element
20/Dec/2017:17:55:27	36	TCP_HIT/200	360p/BBB-I-360p.seg277-L18.svc
20/Dec/2017:17:55:28	205	TCP_MISS/200	360p/BBB-I-360p.seg278-L18.svc
20/Dec/2017:17:55:28	233	TCP_MISS/200	360p/BBB-I-360p.seg277-L17.svc
20/Dec/2017:17:55:28	274	TCP_HIT/200	360p/BBB-I-360p.seg277-L2.svc
20/Dec/2017:17:55:28	498	TCP_HIT/200	360p/BBB-I-360p.seg277-L1.svc
20/Dec/2017:17:55:29	656	TCP_HIT/200	360p/BBB-I-360p.seg277-L16.svc
20/Dec/2017:17:55:29	746	TCP_HIT/200	360p/BBB-I-360p.seg277-L0.svc
20/Dec/2017:17:55:29	134	TCP_HIT/200	360p/BBB-I-360p.seg278-L17.svc
20/Dec/2017:17:55:29	177	TCP_HIT/200	360p/BBB-I-360p.seg278-L2.svc
20/Dec/2017:17:55:29	376	TCP_HIT/200	360p/BBB-I-360p.seg278-L1.svc
20/Dec/2017:17:55:30	601	TCP_HIT/200	360p/BBB-I-360p.seg278-L16.svc
20/Dec/2017:17:55:30	804	TCP_HIT/200	360p/BBB-I-360p.seg278-L0.svc
20/Dec/2017:17:55:31	1	TCP_HIT/200	360p/BBB-I-360p.seg279-L18.svc
20/Dec/2017:17:55:32	1	TCP_HIT/200	360p/BBB-I-360p.seg280-L18.svc
20/Dec/2017:17:55:32	2	TCP_HIT/200	360p/BBB-I-360p.seg279-L2.svc
20/Dec/2017:17:55:32	106	TCP_HIT/200	360p/BBB-I-360p.seg279-L17.svc
20/Dec/2017:17:55:32	169	TCP_HIT/200	360p/BBB-I-360p.seg279-L1.svc
20/Dec/2017:17:55:32	460	TCP_HIT/200	360p/BBB-I-360p.seg279-L16.svc
20/Dec/2017:17:55:32	591	TCP_HIT/200	360p/BBB-I-360p.seg279-L0.svc
20/Dec/2017:17:55:33	1	TCP_HIT/200	360p/BBB-I-360p.seg280-L2.svc
20/Dec/2017:17:55:33	113	TCP_HIT/200	360p/BBB-I-360p.seg280-L17.svc
20/Dec/2017:17:55:33	113	TCP_HIT/200	360p/BBB-I-360p.seg280-L1.svc
20/Dec/2017:17:55:33	463	TCP_HIT/200	360p/BBB-I-360p.seg280-L0.svc

Table 6.11: Detail of cache logs in which the performance is degraded.

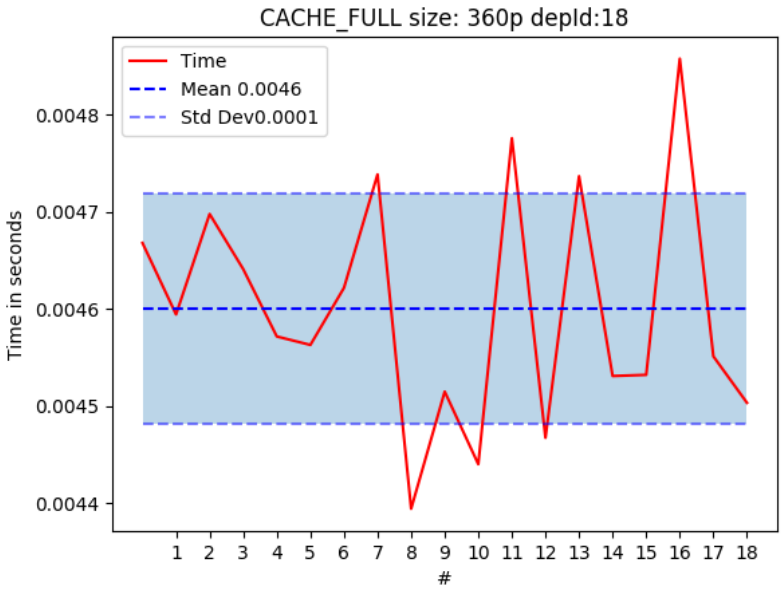


Figure 6.8: Client near, cache full, frame size 360p, Layer 18 mean time for proxy-controller interactions.

giving access to the Internet (labeled as GAIA).
It can be seen how the switch to which the proxy is connected (MAT) is almost constantly at 100% cpu. The overhead in this switch is in two factors, one is that it receives the traffic from the client and also the traffic directed to the cache. This last one is important because each http connection is managed with two OpenFlow flows. The effect can also be seen in GAIA that is holding the cache and acting as a pass-through switch for the proxy-client connection (case far shown in Figure 6.10b) therefore having the same amount of packets passing by. There is a slight reduction in GAIA cpu in the case in which the client is collocated with the proxy, traffic between them is not passing through GAIA (corresponding to case near and shown in Figure 6.10a) and despite having a high cpu usage, it can be seen easily the reduction. The same effect can be seen in BBAA which is only having traffic in the far case (b). The cpu usage for BBAA in that case is very low which is explained that the fact that the switch is only rewriting ip addresses and macs for the response packets in the TCP connection between client and proxy. These series of tests show clearly how just one client overwhelms the switches when the packets need to be sent to the operating system kernel for advanced computations not available in the hardware pipeline.

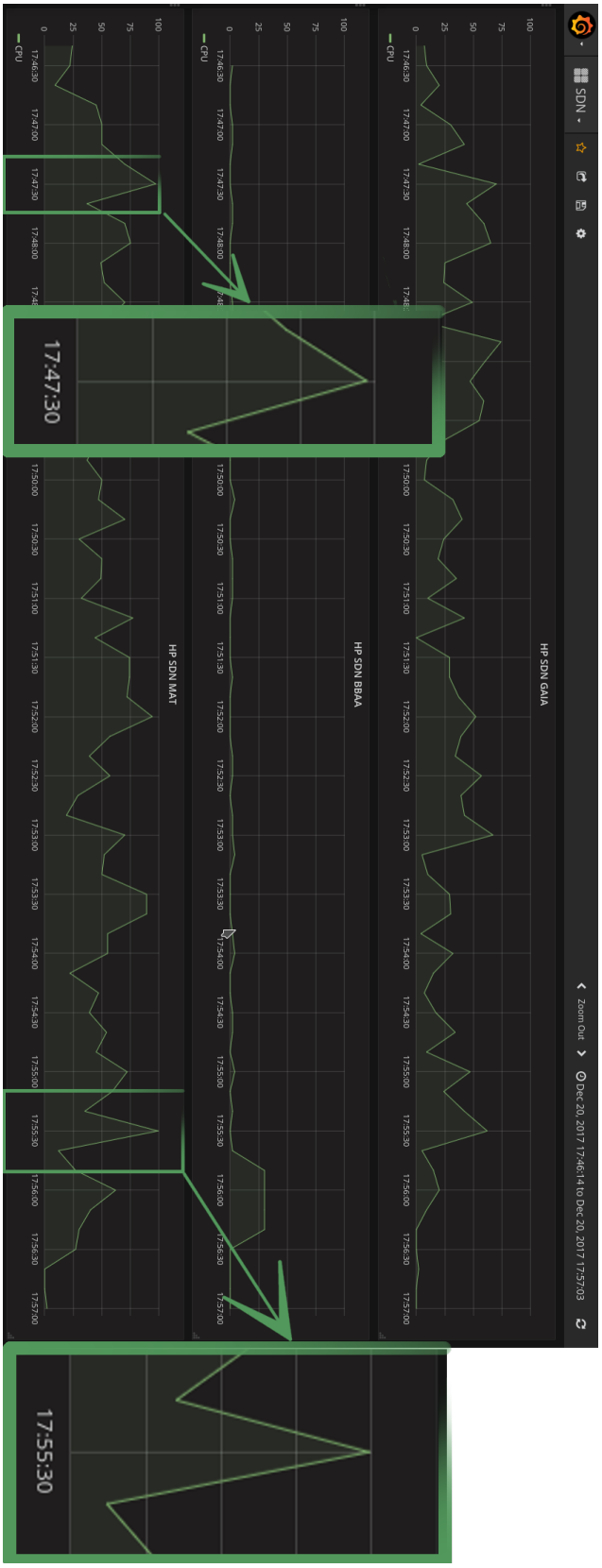
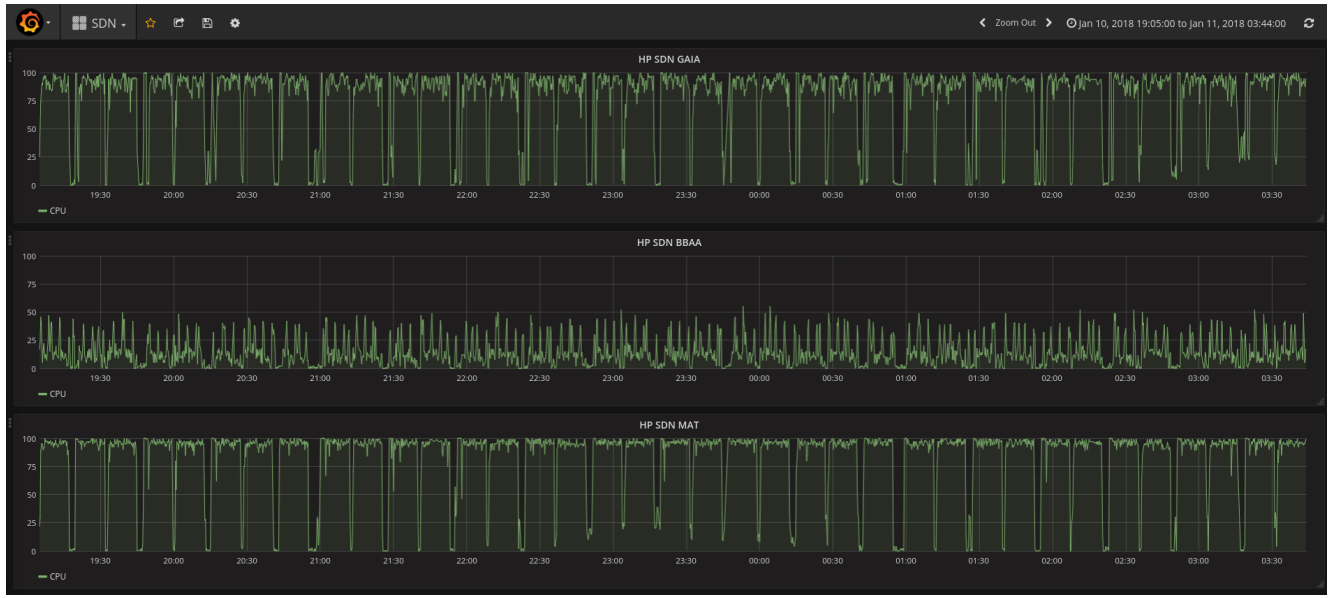


Figure 6.9: CPU usage of the switches in the testbed

(a) CPU usage of the switches in the testbed, all executions for client near, cache full, 1080p and Layer 18



(b) CPU usage of the switches in the testbed, all executions for client far, cache full, 1080p and Layer 18

6.3.3 Evaluating the Prefetching Mechanism

The prefetching mechanism described in 6.1.3 has been evaluated by a series of executions of the scenario in which the client farther away from the proxy streams 360p video at H.264/SVC layer 17. 182 Streaming processes were carried out with a clean squid cache for each process. In terms of cache hit ratio the prefetching mechanisms achieves in average 60.37% while the same case for the 20 executions carried out as the evaluation of the ICNaaS achieved a $98,66\%$ with a squid from which a second round for the same video is retrieved, meaning that the cache has been already filled. In average 2013 HTTP requests are performed to the cache of which 722 are hits that for each layer has a ratio of $L0 \rightarrow 57,87\%$, $L1 \rightarrow 76,26\%$, $L16 \rightarrow 51,50\%$ and $L17 \rightarrow 55,84\%$. In average the first hit is achieved 1,8 seconds after the MPD file is retrieved while the precaching process is finished 4 minutes and 43 seconds afterwards.
Analyzing why the results are so distant from the 100% cache hit ratio, there is a slight difference between the number of requests received by the prefetche, the prefetcher receives in mean 817 requests from the controller while only 722 materialize into cache hits afterwards. In any case, there should be 1196 requests corresponding to 299 files per layer so there are some requests not being performed from the controller. Looking at the logs in general there is a delay on the requests for the two higher layers. In general, the two lower layers (0 and 1) are fully requested prior to any chunk of the upper layers. Although debugging of the controller's prefetching algorithms and prefetching threading should be done to achieve better results, the experiments achieved the goal proposed for this Thesis. Demonstrate that the SDN controller can take profit of the data layer (the H.264/SVC chunk description in the MPD) to trigger protocol specific (prefetcher DASH flows to the cache) actions in real time.

6.4 Conclusions

This chapter has introduced and evaluated the implementations of the proposal described in Chapter 4. The results shown in this chapter are the evolution of research in SDN and ICN as well as CDN within the context of the gn3plus [10] project.
The proof of concepts have shown the feasibility of proxying an HTTP connection by means of SDN traffic steering and how the inspection of the application layer can be fed to the SDN controller to steer the proxy connection to the content source based on the url and some other metadata, such as the MPD for DASH connections or the scalability layer for H.264/SVC on top of DASH.

Additionally, the whole ICNaaS concept is demonstrated by implementing the REST interfaces that would be used by the Content Provider as a mean to instantiate and configure the ICN instances.
The prefetching concept is demonstrated for any DASH stream and in particular for H.264/SVC by taking profit of the inter-layer dependency at the moment of selecting which chunks should be downloaded. The backward compatibility of the concept is demonstrated by employing a standard squid cache, a well known and widely deployed piece of software.
As a result of the evolution of the architecture presented in previous chapter two software implementations were performed and evaluated. One on top of the floodlight [129] controller and another on top of ONOS [14], a growing trend in terms of open source SDN controllers. As a side effect of the elaboration of this Thesis, contributions to the core code of ONOS have been made while investigating the Meter concept introduced with OpenFlow 1.3.
One of the more promising capabilities of OpenFlow is the standardization of a common communication channel and set of capabilities that homogenize the device 'flora'. Nevertheless and as has been explained in this Chapter, not taking into account the hardware capabilities of a device is a fatal error, although capabilities can be retrieved from the switch and the software tables can be avoided unless really necessary, that approach is only feasible for a small amount of packets.
The approach introduced in this Thesis for inspecting the URL is based on the introduction of a stateful proxy that remains during the whole conversation as a man-in-the-middle but its purpose is finished just after the URL is obtained and the controller is notified with the value. Two ways to remove the proxy are envisioned:

- One is introducing a padding that synchronizes the ack values for the proxy server side and the controller server side so that when the url is retrieved some flows rewriting IP addresses can directly connect the client and the server side overlooking the proxy. The big problem with this approach is that TCP is so well designed that the sequence number ranges are up to 4 Gigabytes which is too much information to be used as padding. Multiple connections could be started from the proxy, as many as the statistics demonstrates enough to obtain the desired results and a study would be needed, and use the connection which produces an ack value on the server nearer to the one employed with the client by the proxy, nevertheless it is not a predictable method since is based on statistics and would probably be limited by Denial of Service (DoS) countermeasures.

- Another method would be a more hardware related one in which the openflow

protocol would introduce a new action dedicated to shift the sequence number values in the TCP connections, it is a 'simple' addition to a 32 bit field per packet and would probably be interesting for other applications. Despite the simplicity, it is also true that it probably would be an optional action that the vendor might see as expendable, like the ip address rewriting in the switches employed in the Gaia laboratory. In this direction the P4 [131] protocol, intended to modify programmable data-plane of the network switches, could be employed since it already defines in section 8.5 the addition operation on unsigned integers.

Despite the fact that remaining loyal to HTTP and TCP introduces a hughe cost and complexity, it is also true that both protocols are the standard de-facto and must be integrated in any proposal for the future internet in a seamless and transparent manner. Adopting UDP adaptors allow for a more richful work on the backplane with the OpenFlow switches but introduces problems related to breaking the connection chain on upper levels. The work presented in this chapter could be easily extended by incorporating the NFV paradigm. A proposal that reuses the ICNaaS and the SDN leveraged HTTP ICN paradigm has been presented already (waiting for review) to be used within a Mobile Edge Computing positioning system to provide faster network interactions independent from the network address employed and tied to content generation position.

Chapter 7

Conclusions and future work

This chapter provides a summary of the thesis, its main contributions and conclusions, and discusses the envisaged future work.

7.1 Summary and main contributions

The increase in network bandwidth capabilities gave rise also to the emergence of video streaming services. In the case of the Internet, this rise is estimated to consume 80% of all consumer traffic by 2019 [40] therefore the importance to analyze and optimize how video is transported over the network.
This Thesis has suffered a technological evolution hand by hand with that of video streaming and computer networks. First related works were linked to the Scalnet Project [100] and were surrounded by datagram based video streaming, synchronized and signaled by the well known RTP and Real-time Streaming Protocol (RTSP) protocols. At the same time the work was influenced by the appearance of H.264/SVC as a mean to reduce bandwidth consumption by exploiting picture similarities at different operation points. The PhD work analyzed an available and standardized transmission mechanism (SCTP) to be employed as video transmission mechanism and in particular H.264/SVC video. The research work investigated the different possibilities offered by SCTP and its extensions, applying them to the solely purpose of reducing bandwidth consumption while asserting the best QoE possible for the same QoS.
Soon, and in the context of the OpenLab [9] and SmartFire [125] projects, the video streaming paradigm evolved with the standardization of DASH and the appearance of other HTTP-based video streaming mechanisms, therefore further research was accomplished having in mind the new *de facto* standard. At the same time but in the networking field,

the research community had started to face the ossification of the IP based networking and the FI emerged. Hence, this Thesis has evaluated two different paradigms related to the FI, HIMALIS and CCN which employ a clean-slate approach to the network ossification. In addition, the emerging IoT ecosystem captured the attention of the community as well as mine and decided to take profit of the aforementioned systems evaluation to inspect how video encoded for and by the 'things' would affect the networks on which was foreseeable to be transmitted the data. The exposed work and the articles in which it is supported, provide with a first evaluation of these systems albeit leaving the simulators to use a world wide testbed such as PLE. In addition, evaluation repeatability, usually a lack in our research field, was obtained by using NEPI framework.

Finally, the SDN paradigm and its maximum exponent OpenFlow emerged, germinating from the seeds spread with its standardization. Apart from the technical qualities of OpenFlow, one of its more interesting characteristics is the attention it attracted from the networking industry in general. National Research and Education Networks (NRENs) and in particular their European counterpart, G'eant, were also attracted by the new paradigm, therefore G'eant projects last three years have been pursuing the adoption of this technology for the provision of enhanced services. The experience gained during the project and the feedback obtained by NREN's colleagues gave birth to the ICNaaS proposal and its two implementations. ICNaaS is an 'as a Service' approach to the provision of ICN alike services with a high backward compatibility. The layered design of the solution makes it easily extensible and migratable to other application domains. The two evaluations performed have demonstrated the feasibility of the system despite the problems found on each implementation. In order to be able to evaluate such systems the first two and only University of Murcia's SDN networks deployments have been performed as part of this Thesis and will remain there for further research.

Part of the research performed has produced technical outputs that have already been placed on the opensource community in the form of contributions to two of the most influencing pieces of software in their scope. The Open source audio and video processing tools (LibAV) [13] as a mainstream library for video and audio coding and transmission, which received SCTP as well as dirac contributions (the last one out of the scope of this thesis) and ONOS [14] one of the two more popular opensource SDN controllers which received several patch-sets related to OpenFlow's metering capability as a result of the interest provoked by the pursuit of QoS for the H.264/SVC over ICNaaS evaluation.

7.2 Future work

In order to close this Thesis, a few future research lines, as well as side enhancements to the presented work, are introduced. The items are presented in an order relative to this document chapters being at the same time a summary with few extensions of the future work introduced in each chapter.

7.2.1 SCTP

7.2.1.1 Research lines

The extensions introduced in this work already introduce a new plethora of parameters and the inclusion of new extensions such as the commented SCTP-PF extension.

7.2.1.2 Technical gaps

The study presented in this Thesis related to SCTP as a mean to transport H.264/SVC video could be extended with a thorough study of SCTP's parameters for each video streaming use case, meaning live video and Bandwidth On Demand (BoD).
The promising results presented in this Thesis should be backed up by a real evaluation. This work that should be trivial for a standardized IETF protocol should not be underestimated. SCTP implementations for various systems are minimal, the best support is provided by FreeBSD with a patch including the aforementioned extensions to be applied to the BSD kernel `https://github.com/sctplab/sctp-refimpl/tree/master/KERN`, while linux support is restricted to the already standardized implementation and not frequently updated, Windows systems on the other hand do not have SCTP native support at all. A good option to carry out such an evaluation would be using the multi-platform non-kernel libraries from *sctplab* `https://github.com/sctplab/usrsctp`, despite not being code not part of the operating system kernel which means that is executed at lower priority, it would allow a thorough evaluation and to be installed in distributed testbeds such as PLE in which the kernel is limited to that being executed at the host.

7.2.2 FI

The FI approach is still valid and has become more important each day. What was considered future, such as IoT or SDN has become the present as the industry has started to accept them and take profit of their capabilities or their business value.

7.2.2.1 Research lines

Some others had not attracted industry's attention but have not yet been superseded such as HIMALIS, alternatives have though been available for long like Mobile Oriented Future Internet (MOFI).
The validity of ICN principles has increased and attracts more attention each time, therefore upgrading the study of this Thesis on FI architectures for content centered delivery would be desirable.
Focusing on the IoT and its evolution, now that IoT devices are easily available, an study of their video capabilities, formats and network streams should be made, updating the premises in which the study presented in this Thesis was founded.

7.2.2.2 Technical gaps

New ICN software has been delivered by the community (such as Named Data Networking (NDN) hosted by the 'Named data networking' project [132]) and distributed deployments have (such as CiCN [133]) have been instantiated simplifying the process of creating a proof of concept with which validate the studies.

7.2.3 ICNaaS

ICNaaS architecture presented in Chapter 5 is a generic HTTP content centered delivery system, therefore easily adaptable to other application domains not centered on DASH video streaming.

7.2.3.1 Research lines

As a first approach, it has been proposed as a mechanism to optimize communications in a location system, where the user mobile phone employs REST calls to a server to off-load position calculations to a commodity server. With the adoption of the ICNaaS architecture, those calls are directed to the nearest commodity server in a 'fog' like approach allowing later requests for the data to be optimally directed to the commodity server that made the calculation, that work is under review process.
The video application of ICNaaS still has some fields in which can be extended, and in particular taking profit of scalable or multi-view coding, such as the caching algorithms and how the content can be balanced over the caches depending on the business strategy, that strategy can be in addition observed from two point of view, the content provider who

probably wants to reduce the latency and enhance the perceived QoE of the user and the ISP that would probably focus on reducing the backbone stress.
One of the items not faced by the work carried out and of capital relevance in nowadays communications is **security**.
Later technologies like NFV and Management and Orchestration (MANO) offer, among other capabilities, the virtualisation of network elements such as the named proxy, but even without adopting those technologies, the proposal in this Thesis already considered the possibility of having various proxies per ICN instance, up to one proxy per client, more could be registered but would be unused. In general when speaking about video transmission optimisation techniques, security is delegated to a second phase and although there is work already done in the research community it is an open field for research.
The ICNaaS architecture could be enhanced by adopting transport layer protocols more suitable for transmission, speaking about multimedia transmissions, the presented SCTP extensions could be adopted and/or other emerging protocols like *QUIC* could be studied as TCP alternatives.
Finally, the adoption of orchestration mechanisms and the integration of the ICNaaS as an NFV service is envisioned as crucial for the continuity of the proposal. Should the SDN controller within the NFV be exposed to the ICNaaS? Should the ICNaaS be sitting on top of such a controller? Should a virtual network be exposed to a virtual SDN controller running on top of the NFV architecture? There are still some open questions and work to be done.

7.2.3.2 Technical gaps

The impact of adopting HTTPS in the ICNaaS architecture should be similar to that of adopting that technology on CDNs. The proxy in charge of inspecting the URL and acting as relying party between the HTTP client and the data source needs the cryptographic material from the content provider in order to be identified as part of the later. That is a common practice in CDNs, the advantage is that there is no need to deploy that same material on the nodes actually providing with the content, mainly the caches. The disadvantage is that the proxy becomes a bottleneck when not properly scaled.
The experience obtained while evaluating the ICNaaS proposal demonstrated that, although the SDN controllers try (in general) to abstract the network programmer from the devices running the network, it is necessary to make the applications aware of the capabilities of the devices. Therefore a review on the northbound interfaces exposed by controllers is needed, if not capabilities per device, certainly there is a need for capabilities

per path and the possibility to forbid the software based capabilities. From the practical experience, unless the devices is very powerful, while performing software modifications to the packet in the device it is better to adopt a virtual switch on a commodity server where CPU power is not a concern.
The proxy removal by synchronizing the client side and the server side TCP streams could probably be accomplished with programmable switches by means of the P4 [131] protocol. P4 indeed opens a new world of possibilities but also new challenges.

Glossary

AHS Adaptive HTTP Streaming. 21

API Application Programming Interface. 122, 140, 146, 150, 151

ARP Address Resolution Protocol. 123

AS Autonomous System. 32

BGP Border Gateway Protocol. 32, 33

BoD Bandwidth On Demand. 183

bps Bits per Second. xv

CCN Content-Centric Networking. viii, ix, xvii, xviii, xxiv, xxvii, xxviii, 2, 3, 5, 43, 46, 73, 74, 91, 94–99, 104, 107, 110–115, 119, 121, 123, 130, 182

CDN Content Delivery Network. ix, xviii, xxiii, 2, 6, 23–25, 43–45, 47, 117, 119–121, 123, 140, 141, 178, 185

CMT-SCTP Concurrent MultiPath Transfer for Stream Control Transmission Protocol. vii, xvii, xxiii, xxiv, 4, 25, 28, 29, 51, 53, 54, 56, 57, 71

CPU Central Processing Unit. xxviii, 50, 172, 176, 177, 186

DASH Dynamic Adaptive Streaming over HTTP. x, xix, xxiv, xxvii, xxix, 5, 6, 21–23, 44, 46, 77–80, 82, 85, 86, 88, 91, 93, 97, 104, 107, 110, 112, 115, 118, 119, 127, 130, 132, 139–141, 143, 147, 151, 152, 172, 178, 179, 181, 184

DNS Domain Name System. 39, 40, 42, 120

DONA Data-Oriented Network Architecture. 42, 43

DoS Denial of Service. 25, 179

dpid OpenFlow Datapath ID. 127–129, 146

eBGP External Border Gateway Protocol. 32

EID Endpoint Identifier. 39

Ethane Ethane: A Protection Architecture for Enterprise Networks. 33, 34, 37

FI Future Internet. vi, viii, ix, xvi–xix, xxiii–xxv, 1–3, 5, 8, 39, 42, 46, 47, 73, 74, 76–117, 120, 182–184

ForCES Forwarding and Control Element Separation. 32

FPS Frames per second. vii

GSMP General Management Switch Protocol. 30

H.264/AVC Advanced Video Coding. vii, xvi, xxiv, xxxi, 15–18, 22, 46, 50, 132, 133, 152

H.264/SVC Scalable Video Coding. vii, viii, x, xvi, xvii, xix, xxiii, xxiv, xxvii, xxxi, 3, 4, 6, 17–19, 47, 49–52, 71, 127, 130, 132–135, 139–141, 143, 147, 156, 172, 178, 179, 181–183

HD High Definition. 71

HEVC High Efficiency Video Coding. 16

HIMALIS Heterogeneity Inclusion and Mobility Adaptation through Locator ID Separation. viii, ix, xvii, xviii, xxiii, xxiv, xxvii–xxix, 3, 5, 40, 41, 73, 74, 79, 80, 82–89, 91–93, 97, 110–115, 182, 184

HIP Host Identity Protocol. 39, 40

HLS HTTP Live Streaming. 21

HTTP HyperText Transfer Protocol. viii–x, xviii, xix, 2, 3, 5, 6, 14, 15, 20, 21, 24, 46, 47, 71, 73, 79, 82, 84, 91, 94, 95, 99, 112, 117–120, 122, 123, 125, 130–132, 140, 141, 143, 146–148, 150–152, 155, 171, 178, 180, 181, 184, 185

HTTPS HTTP over TLS. 25, 141, 185

iBGP Internal Border Gateway Protocol. 32

ICMP Internet Control Message Protocol. 79

ICN Information Centric Networking. viii–x, xvii–xix, 2, 5, 6, 38, 41, 42, 44–47, 73, 74, 91, 107, 110, 112, 115, 117, 119–123, 125, 127, 128, 130, 131, 133, 140, 143, 144, 146–148, 151, 178–180, 182, 184, 185

ICNaaS Information Centric Network as a Service. ix, x, xviii, xix, xxiii–xxv, xxviii, xxix, xxxi, 6, 8, 43–45, 47, 117, 120–123, 125, 127, 128, 130–134, 139–141, 143–149, 151, 152, 161, 178–180, 182, 184, 185

IEC International Electrotechnical Commission. 118

IETF Internet Engineering Task Force. vi, xvi, 31–33, 118, 183

IoT Internet of Things. vi, viii, xvi, xviii, xxiii, xxiv, 3, 5, 25, 40, 47, 73–77, 79, 81–83, 85–87, 89, 91, 93–95, 97, 99, 101, 103–107, 109–111, 113, 115, 182–184

IP Internet Protocol. viii, xv–xvii, 3, 5, 32, 49, 73, 115, 119, 120, 123, 125, 127, 128, 146, 151, 179, 182

IRTF Internet Research Task Force. 40

ISDN Integrated Services Digital Network. 2, 16

ISO Organization for Standardization. 118

ISP Internet Service Provider. 24, 43–45, 118, 121–123, 129, 131, 134, 139, 185

ITU-T ITU's Telecommunication Standardization Sector. 14

kbps kilobit per second. 99, 104, 106

LibAV Open source audio and video processing tools. 7, 182

LISP Location/ID Separation Protocol. 39, 40

mac Media Access Control. 123, 127, 128, 146

MANE Media Aware Network Element. 134

MANO Management and Orchestration. 185

MEC Mobile Edge Computing. 121, 125

MOFI Mobile Oriented Future Internet. 40, 184

MPD Media Presentation Description. xxxi, 21, 22, 46, 112, 119, 130, 132, 133, 147, 178

MPEG Moving Picture Experts Group. 16, 21

MTU Maximum Transmission Unit. xxvii–xxix, 74, 75, 79, 80, 86–89, 92, 93, 97–102, 104–109, 115

NALU Network Abstraction Layer Unit. 50

NAT Network Address Translation. 118

NDN Named Data Networking. 184

NEPI Network Experimentation Programming Interface. xxxi, 5, 79, 82, 112, 116, 159, 161, 165, 182

NetConf Network Configuration Protocol. 33

NetInf Network of Information. 42

NFV Network Function Virtualisation. x, xix, 140, 180, 185

NICT National Institute of Information and Communication Technology of Japan. 40

NOC Network Operations Center. 44, 154

NodeID Node Identity Internetworking Architecture. 39, 40

NOS Network Operating System. xxvii, 33, 34

NREN National Research and Education Network. 182

NTP Network Time Protocol. 172

ONOS Open Network Operating System. 7, 146, 171, 182

OpenCache OpenCache. 44, 45

OpenFlow OpenFlow Protocol. ix, xviii, xix, xxvii, 29, 33–38, 45, 122, 123, 125, 127, 128, 141, 146, 148, 152, 156, 172, 175, 179, 180, 182

OPENSIG Open Signalling Working Group. 30, 31

OSI Open Systems Interconnection. vi, xvi, 34

PLE PlanetLab Europe. 5, 7, 74, 79, 80, 82, 84, 86, 91, 95, 97, 104, 107, 110, 182, 183

PSIRP Publish-Subscribe Internet Routing Paradigm. 42, 43

PSNR Peak Signal Noise Ratio. xxvii, 56, 57, 89, 90, 104–107

PURSUIT Publish Subscribe Internet Technology. 42, 43

QoE Quality of Experience. 15, 57, 89, 104–107, 140, 181, 185

QoS Quality of Service. 7, 24, 44, 134, 181, 182

RANGI Routing Architecture for the Next Generation Internet. 39

RCP Routing Control Platform. 32, 33

RDA Rate Determination Algorithm. 132, 139

REST REpresentation State Transfer. xxviii, 122, 140, 143–147, 150–152, 171, 179, 184

RLOC Routing Locator. 39

RPC Remote Procedure Call. 33

RTMP Real-time Messaging Protocol. 7, 15, 73

RTP Real-time Transport Protocol. vi, vii, xvi, xvii, xxiv, 7, 14, 15, 49, 51–53, 55, 56, 73, 181

RTSP Real-time Streaming Protocol. vi, xvi, 7, 14, 15, 23, 24, 181

RTT Round Trip Time. 37, 82, 97, 104, 110

SAP Session Announcement Protocol. 15

SCTP Stream Control Transmission Protocol. vi, vii, xvi, xvii, xxiii–xxv, 3, 4, 7, 8, 25–29, 47, 49–58, 60, 62, 64, 66, 68, 70–72, 141, 181–183, 185

SCTP-PF Stream Control Transmission Protocol Potentially Failed. 4, 71, 183

SDN Software Defined Networking. vi, ix, x, xvi, xviii, xix, xxiii, 3, 4, 6–8, 29–31, 33, 37, 38, 44–47, 117, 120–123, 125, 127, 128, 130–132, 139–141, 143, 146, 151, 152, 166, 178–180, 182, 183, 185

SDP Session Description Protocol. vi, xvi, 14, 15

SHVC Scalable High-Efficiency Video Coding. 47, 71

SLA Service Level Agreement. 44

TCP Transmission Control Protocol. vi, xv, xxiv, 5, 20, 49–53, 56, 71, 73, 82, 94, 117–119, 123, 125, 127, 128, 130, 141, 146, 148, 151, 175, 179, 180, 185, 186

UDP User Datagram Protocol. vi, xv, 5, 37, 49–51, 73, 82, 91, 117, 118, 180

URI Uniform Resource Identifier. 110, 125, 151

URL Uniform Resource Locator. viii, ix, xvii, xix, 6, 21, 44, 122, 123, 127, 128, 130, 141, 144, 147, 150, 179, 185

VCEG Video Coding Experts Group. 16

VCR Video Cassette Recorder. 14, 15

VLAN Virtual Local Area Network. 37, 154

VoD Video on Demand. 23, 44–46, 49

WebRTC Web Real-Time Communications. 46, 47

WLAN Wireless Local Area Network. 46

XML eXtensible Markup Language. 33, 118

Bibliography

[1] Jordi Ortiz, Eduardo Graciá, and Antonio F Skarmeta. SCTP as scalable video coding transport. *EURASIP Journal on Advances in Signal Processing*, 2013(1):115, 2013.

[2] Eduardo Martinez, Jordi Ortiz, L Rafael, and Antonio F Skarmeta. Video adaptation based on the SVC file format. *CONTENT 2013 : The Fifth International Conference on Creative Content Technologies*, (c):30–37, 2013.

[3] Jordi Ortiz, Pedro Martinez-Julia, and Antonio Skarmeta. 6. information-centric network for future internet video delivery. In *User-centric and Information-centric Networking and Services: Access Networks and Emerging Trends*. Taylor & Francis, 2018.

[4] Y. Nishida, P. Natarajan, A. Caro, P. Amer, and K. Nielsen. SCTP-PF: A Quick Failover Algorithm for the Stream Control Transmission Protocol. Technical report, 2016.

[5] J. Iyengar, P. Amer, and R. Stweart. Concurrent multipath transfer using sctp multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking*, 14(5), 2006.

[6] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad. *RFC 3758: Stream Control Transmission Protocol Partial Reliability Extension*. Internet Engineering Task Force, 2004.

[7] M. Ransburg, E. Martínez Graciá, T. Sutinen, J. Ortiz, M. Sablatschan, and H. Hellwagner. Scalable video coding impact on networks. 2010.

[8] Pedro Martinez-Julia, Elena Torroglosa Garcia, Jordi Ortiz Murillo, and Antonio F. Skarmeta. Evaluating video streaming in network architectures for the internet of

things. *Proceedings - 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2013*, pages 411–415, 2013.

[9] OpenLab project. `http://www.ict-openlab.eu/`.

[10] GN3 Plus project. `https://geant3plus.archive.geant.net/Pages/default.aspx`.

[11] ns-2 Network Simulator. `http://nsnam.isi.edu/nsnam`.

[12] Gaia Testbed. `http://projects.sigma-orionis.com/smartfire/european-testbeds/gaia-testbed-umu/`.

[13] LibAV - Open source audio and video processing tools. `http://libav.org`.

[14] ONOS project. `http://www.onosproject.org/`.

[15] Kostas Choumas, Thanasis Korakis, Jordi Ortiz, Antonio Skarmeta, Pedro Martinez-Julia, Taewan You, Loic Baron, Serge Fdida, Woojin Seok, Minsun Lee, et al. Federated experimentation infrastructure interconnecting sites from both europe and south korea (smartfire). In *Building the Future Internet through FIRE*, pages 717–743. 2016.

[16] Adam Austerberry. *The technology of Video & Audio Streaming*. Elsevier, second edition, 2005.

[17] Ramesh Jain. Let's Weave the Visual Web. *IEEE Multimedia*, 22(3):66–72, 2015.

[18] C. Perkins. *RTP. Audio and Video for the Internet*. Addison-Wesley, 2003.

[19] Benny Bing. *Next-Generation Video Coding and Streaming*. Wiley, 2015.

[20] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Technical report, Internet Engineering Task Force, July 2003. Standard, RFC 3550.

[21] Mark Handley, Colin Perkins, and Edmund Whelan. Session announcement protocol. 2000.

[22] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. Technical report, Internet Engineering Task Force, July 2006. Proposed Standard, RFC 4566.

[23] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). Technical report, Internet Engineering Task Force, April 1998. Proposed Standard, RFC 2326.

[24] H Parmar and M Thornburgh. Adobe's real time messaging protocol. *Copyright Adobe Systems Incorporated*, pages 1–52, 2012.

[25] ISO/IEC 11172-2:1993 - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video . Technical report, 1993.

[26] ISO/IEC 13818-2:2013 - Generic coding of moving pictures and associated audio information – Part 2: Video . Technical report, 2013.

[27] ISO/IEC 14496-2:2004 - Coding of audio-visual objects – Part 2: Visual . Technical report, 2004.

[28] Theora Specification, 2004.

[29] ISO/IEC 23008-2:2013 High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding . Technical report, 2013.

[30] Adrian Grange and Harald Alvestrand. A vp9 bitstream overview. 2013.

[31] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1107, September 2007.

[32] Yan Ye and Pierre Andrivon. The scalable extensions of HEVC for ultra-high-definition video delivery, 2014.

[33] Ying Chen, Ye-Kui Wang, Kemal Ugur, Miska M Hannuksela, Jani Lainema, and Moncef Gabbouj. The Emerging MVC Standard for 3D Video Services. *EURASIP Journal on Advances in Signal Processing*, 2009(1):786015, 2009.

[34] T. Wiegand, G. Sullivan, H. Schwarz, and M. Wien. *ISO/IEC 14496-10:2005/Amd3: Scalable Video Coding*. International Standardization Organitation, 2007.

[35] H. Schwars, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9), 2007.

[36] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7), 2003.

[37] Iain E. G. Richardson. *Video Codec Design.* John Wiley & Sons, 2002.

[38] Roy Fielding, James Gettys, Jeff Mogul, Henrik Frystyk, Larry Masinter, P. Leach, and Tim Berners-Lee. RFC2616 - Hypertext transfer protocol–HTTP/1.1. *Internet Engineering Task Force*, pages 1–114, 1999.

[39] J Postel. RFC 793 - Transmission Control Protocol. *Rfc 793*, 25(September):1–85, 1981.

[40] Cisco. Cisco Global Cloud Index : Forecast and Methodology , 2014–2019. *White Paper*, pages 1 – 41, 2014.

[41] Wang Bing, J I M Kurose, Prashant Shenoy, and D O N Towsley. Multimedia Streaming via TCP: An Analytic Performance Study. *ACM Transactions on Multimedia Computing, Communications & Applications*, 4(2):16:1 – 16:22, 2008.

[42] Patrick Seeling, Frank H. P. Fitzek, Gergö Ertli, Akshay Pulipaka, and Martin Reisslein. Video network traffic and quality comparison of VP8 and H.264 SVC. *Proceedings of the 3rd workshop on Mobile video delivery - MoViD '10*, page 33, 2010.

[43] Arkadiusz Biernacki and Kurt Tutschku. Performance of HTTP video streaming under different network conditions. *Multimedia Tools and Applications*, 72(2):1143–1166, 2014.

[44] Hina Rani and Er.khushboo Bansal. A Review on HTTP Streaming Strategies in Media Streaming. *International Journal Of Engineering And Computer Science*, 4(8):14033–14035, 2015.

[45] Roger Pantos. Http live streaming - draft-pantos-http-live-streaming-19. Internet-draft, 2015. draft-pantos-http-live-streaming-19.

[46] Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats. 2013:1–61, 2013.

[47] Christian Timmerer and Christopher Mueller. HTTP Streaming of MPEG Media. *Proceedings of the Multimedia Communication 26 (MM 26)*, pages 1–4, 2010.

[48] MG Michalos, SP Kessanidis, and SL Nalmpantis. Dynamic adaptive streaming over http. *Journal of Engineering Science and Technology Review*, 5(2):30–34, 2012.

[49] D. Singer, editor. *ISO/IEC 14496-12:2005 Part 12: ISO Base Media File Format.* International Organization for Standardization, 2005.

[50] Al-mukaddim Khan Pathan and Rajkumar Buyya. A Taxonomy and Survey of Content Delivery Networks. *Grid Computing and Distributed Systems GRIDS Laboratory University of Melbourne Parkville Australia*, 148:1–44, 2006.

[51] R. Stewart. Rfc 4960, stream control transmission protocol. Technical report, 2007.

[52] R. Stewart and Qiaobing Xie. *Stream control Transmission Protocol. A Reference Guide.* Addison-Wesley, 2002.

[53] L. Budzisz, J. Garcia, A. Brunstrom, and R. Ferrús. A taxonomy and survey of sctp research. *ACM Computing Surveys*, 44(4), 2012.

[54] Bruno AA Nunes, Manoel Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *Communications Surveys & Tutorials, IEEE*, 16(3):1617–1634, 2014.

[55] Andrew T. Campbell, Irene Katzela, Kazuho Miki, and John Vicente. Open signaling for ATM, internet and mobile networks (OPENSIG'98). *ACM SIGCOMM Computer Communication Review*, 29(1):97, 1999.

[56] R. Yang, L. and Dantu, R. and Anderson, T. and Gopal. Forwarding and Control Element Separation (ForCES) Protocol Specification. *RFC Editor*, (RFC3746):1689–1699, 2010.

[57] A Doria, J Hadi Salim, R Haas, H Khosravi, W Wang, J Gopal, and J Halpern. Forwarding and Control Element Separation (ForCES) Protocol Specification. *RFC Editor*, (RFC5810):1689–1699, 2010.

[58] Nick Feamster, Hari Balakrishnan, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe. The case for separating routing from routers. *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture - FDNA '04*, page 5, 2004.

[59] R Enns. NETCONF Configuration Protocol. Technical Report Xml, 2006.

[60] A Bierman, M Bjorklund, J Schoenwaelder, and A Bierman. Network Configuration Protocol (NETCONF). Technical report, 2011.

[61] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69, 2008.

[62] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The Road to SDN: An Intellectual History of Programmable Networks. *ACM Sigcomm Computer Communication*, 44(2):87–98, 2014.

[63] B Astuto. A Survey of Software–Defined Networking: Past, Present, and Future of Programmable Networks,. *in IEEE Communications surveys & tutorials*, 16(3), 2014.

[64] Ben Pfaff, Brandon Heller, Dan Talayco, David Erickson, Glen Gibb, Guido Appenzeller, Jean Tourrilhes, Justin Pettit, KK Yap, Martin Casado, Masayoshi Kobayashi, Nick McKeown, Peter Balland, Reid Price, Rob Sherwood, and Yiannis Yiakoumis. OpenFlow Switch Specification 1.0.0. *Current*, 0:1–36, 2009.

[65] Anders Nygren, Ben Pfaff, Bob Lantz, Brandon Heller, Casey Barker, Curt Beckmann, Dan Cohn, Dan Malek, Dan Talayco, David Erickson, David McDysan, David Ward, Edward Crabbe, Fabian Schneider, Glen Gibb, Guido Appenzeller, Jean Tourrilhes, Johann Tonsing, Justin Pettit, KK Yap, Leon Poutievski, Linda Dunbar, Lorenzo Vicisano, Martin Casado, Masahiko Takahashi, Masayoshi Kobayashi, Michael Orr, Navindra Yadav, Nick McKeown, Nico DHeureuse, Peter Balland, Rajesh Madabushi, Rajiv Ramanathan, Reid Price, Rob Sherwood, Saurav Das, Shashidhar Gandham, Spike Curtis, Sriram Natarajan, Tal Mizrahi, Tatsuya Yabe, Wanfu Ding, Yiannis Yiakoumis, Yoram Moses, and Zoltán Lajos Kis. OpenFlow Switch Specification 1.5.1. *Current*, 0:1–36, 2015.

[66] Ben Pfaff, Bob Lantz, Brandon Heller, Casey Barker, Curt Beckmann, Dan Cohn, Dan Talayco, David Erickson, David McDysan, David Ward, Edward Crabbe, Glen Gibb, Guido Appenzeller, Jean Tourrilhes, Johann Tonsing, Justin Pettit, KK Yap, Leon Poutievski, Lorenzo Vicisano, Martin Casado, Masahiko Takahashi, Masayoshi Kobayashi, Navindra Yadav, Nick McKeown, Nico DHeureuse, Peter, Balland, Rajiv Ramanathan, Reid Price, Rob Sherwood, Saurav Das, Shashidhar

Gandham, Tatsuya Yabe, Yiannis Yiakoumis, and Zoltán Lajos Kis. OpenFlow Switch Specification 1.3.0. *Current*, 0:1–36, 2012.

[67] K Benson and A Lerner. Brite-Box: Branded Switching + White-Box Switching . http://blogs.gartner.com/andrew-lerner/2014/11/19/britefuture/.

[68] Meng Zhang, Hongbin Luo, and Hongke Zhang. A survey of caching mechanisms in information-centric networking. *IEEE Communications Surveys and Tutorials*, 17(3):1473–1499, 2015.

[69] Athanasios V. Vasilakos, Zhe Li, Gwendal Simon, and Wei You. Information centric network: Research challenges and opportunities. *Journal of Network and Computer Applications*, 52:1–10, 2015.

[70] Gabriel M. Brito, Pedro Braconnot Velloso, and Igor M. Moraes. *Information-Centric Networks: A New Paradigm for the Internet*. 2013.

[71] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09)*, pages 1–12, New York, NY, USA, 2009. ACM.

[72] Dirk Trossen et al. Pursuing a Pub/Sub Internet (PURSUIT), 2011. `http://www.fp7-pursuit.eu`.

[73] Bengt Ahlgren, Matteo D'Ambrosio, Christian Dannewitz, et al. Netinf evaluation. Technical Report FP7-ICT-2007-1-216041-4WARD/D-6.3, 2010. Deliverable D-6.3, 4WARD EU FP7 Project, `http://www.4ward-project.eu`.

[74] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture, 2006. `http://www.ietf.org/rfc/rfc4423.txt`.

[75] David Meyer. The locator identifier separation protocol (lisp). *The Internet Protocol Journal*, 11(1):23–36, 2008.

[76] D. Farinacci, V. Fuller, D. Meyer, and D Lewis. Locator/id separation protocol (LISP). Internet-draft, IETF, 2011.

[77] Jukka Ylitalo and Pekka Nikander. BLIND: A complete identity protection framework for end-points. *Lecture Notes in Computer Science*, 3957:163–176, 2006.

[78] X. Xu. Routing Architecture for the Next Generation Internet (RANGI). Internet-draft, IETF, 2009.

[79] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme. A node identity internetworking architecture. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–6, Washington, DC, USA, 2006. IEEE.

[80] Heeyoung Jung and Seok Joo Koh. MOFI: Future internet architecture with address-free hosts for mobile environments. *Telecommunications Review*, 21(2):343–358, 2011.

[81] Randall Atkinson, Saleem Bhatti, and Stephen Hailes. ILNP: mobility, multihoming, localised addressing and security through naming. *Telecommunication Systems*, 42(3):273–291, 2009.

[82] Information-centric networking research group, 2012. `https://irtf.org/icnrg`.

[83] Ved P. Kafle and Masugi Inoue. HIMALIS: Heterogeneity inclusion and mobility adaptation through locator id separation in new generation network. *IEICE Transactions on Communications*, E93-B(3):478–489, 2010.

[84] Marcus Brunner, Henrik Abramowicz, Norbert Niebert, and Luis M. Correia. 4WARD: A european perspective towards the future internet. *IEICE Transactions on Communications*, E93-B(3):442–445, 2010.

[85] Thomas Edwall et al. Scalable and Adaptive Internet Solutions (SAIL), 2011. `http://www.sail-project.eu`.

[86] Vladimir Dimitrov and Ventzislav Koptchev. PSIRP project – publish-subscribe internet routing paradigm: New ideas for future internet. In *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*, pages 167–171, New York, NY, USA, 2010. ACM.

[87] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Computer Communication Review*, 37(4):181–192, 2007.

[88] Pedro Martinez-Julia and Antonio F. Gomez-Skarmeta. Using identities to achieve enhanced privacy in future content delivery networks. *Computers and Electrical Engineering*, 38(2):346–355, 2012.

[89] Panagiotis Georgopoulos, Matthew Broadbent, Bernhard Plattner, and Nicholas Race. Cache as a service: Leveraging SDN to efficiently and transparently support video-on-demand on the last mile. *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, 2014.

[90] Reinhard Grandl, Kai Su, and Cedric Westphal. On the interaction of adaptive video streaming with content-centric networking. *2013 20th International Packet Video Workshop, PV 2013*, 2013.

[91] Jill M. Boyce, Yan Ye, Jianle Chen, and Adarsh K. Ramasubramonian. Overview of SHVC: Scalable extensions of the high efficiency video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):20–34, 2016.

[92] Alex Eleftheriadis. Recent Developments in Scalable Coding in VP9. Chicago, 2016.

[93] Michael Grafl, Christian Timmerer, Hermann Hellwagner, George Xilouris, Georgios Gardikis, Daniele Renzi, Stefano Battista, Eugen Borcoci, and Daniel Negru. Scalable media coding enabling content-aware networking. *IEEE Multimedia*, 20(2):30–41, 2013.

[94] Junghwan Lee, Kyubo Lim, and Chuck Yoo. Cache replacement strategies for scalable video streaming in CCN. *2013 19th Asia-Pacific Conference on Communications, APCC 2013*, pages 184–189, 2013.

[95] Saeed Ullah, Choong Seon Hong, and South Korea. Opportunistic Quality Adaptation for Scalable Video Streaming in Information Centric Networks. pages 1241–1243, 2015.

[96] Horacio Sanson, Alvaro Neira, Luis Loyola, and Mitsuji Matsumoto. PR-SCTP for Real Time H.264/AVC Video Streaming. 2010.

[97] M Molteni and M Villari. Using SCTP with Partial Reliability for MPEG-4 Multimedia Streaming. pages 1–8, 2002.

[98] Information Sciences Institute (University of Southern California). The network simulator ns-2.

[99] Protocol Engineering Laboratory (University of Delaware). ns-2 sctp module.

[100] Celtic Scalnet project. https://www.celticplus.eu/project-scalnet/.

[101] P. Amon, T. Rathgen, and D. Signer. File format for scalable video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9), 2007.

[102] PD. Amer, AL. Caro Jr., J. Iyengar, P. Natarajan, and N. Ekiz. ns-2 sctp readme file. "http://nsnam.cvs.sourceforge.net/viewvc/nsnam/ns-2/sctp/sctp.README", 2009.

[103] Anastasius Gavras, Andrzej Bak, Gergely Biczók, Piotr Gajowniczek, András Gulyás, Halid Hrasnica, Pedro Martinez-Julia, Felicián Németh, Chrysa Papagianni, Symeon Papavassiliou, Marcin Pilarski, and Antonio Skarmeta. Heterogeneous testbeds, tools and experiments - Measurement requirements perspective. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013.

[104] Pedro Martinez-Julia, Antonio J. Jara, and Antonio F. Skarmeta. GAIA extended research infrastructure: Sensing, connecting, and processing the real world. In *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, volume 44 LNICST, pages 3–4, 2012.

[105] Heterogeneous Experimental Network. http://mediatools.cs.ucl.ac.uk/nets/hen.

[106] PlanetLab Europe. http://www.planet-lab.eu.

[107] Alina Quereilhac, Mathieu Lacage, Claudio Freire, Thierry Turletti, and Walid Dabbous. NEPI: An integration framework for Network Experimentation. *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, pages 1–5, 2011.

[108] Geoff Mulligan. The 6LoWPAN architecture. In *Proceedings of the 4th workshop on Embedded networked sensors - EmNets '07*, page 78, 2007.

[109] Ed Callaway, Paul Gorday, Lance Hester, Jose A. Gutierrez, Marco Naeve, Bob Heile, and Venkat Bahl. Home networking with IEEE 802.15.4: A developing standard for low-rate wireless personal area networks. *IEEE Communications Magazine*, 40(8):70–77, 2002.

[110] Joint Scalable Video Model, 2007.

[111] Xiph.org video test media. `http://media.xiph.org/video/derf/`.

[112] Libdash: ISO/IEC MPEG-DASH reference software. `https://bitmovin.com/libdash/`.

[113] Videolan x264. `https://www.videolan.org/developers/x264.html`.

[114] Content-Centric Networking Project. `http://www.ccnx.org`.

[115] M.R. Rohrer. Seeing is believing: the importance of visualization in manufacturing simulation. *Proceedings of the 2000 Winter Simulation Conference*, pages 1211–1216, 2000.

[116] Chandra Kopparapu. *Load Balancing Servers, Firewalls, and Caches*. Robert Ipsen, 2002.

[117] Andrej Binder, Tomas Boros, and Ivan Kotuliak. A SDN Based Method of TCP Connection Handover. pages 13–19, 2015.

[118] Danny H Lee, Constantine Dovrolis, and Ali C Begen. Caching in HTTP Adaptive Streaming: Friend or Foe? *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, pages 31:31—-31:36, 2013.

[119] Yago Sanchez, Thomas Schierl, Cornelius Hellge, Thomas Wiegand, Dohy Hong, Danny De Vleeschauwer, Werner Van Leekwijck, and Yannick Lelouedec. Improved caching for HTTP-based Video on Demand using Scalable Video Coding. *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 595–599, 2011.

[120] Raf Huysegems, Bart De Vleeschauwer, Tingyao Wu, and Werner Van Leekwijck. SVC-Based HTTP Adaptive Streaming. *Bell Labs Technical Journal*, 16(4):25–42, 2012.

[121] Jordi Ortiz, Michael Ransburg, Eduardo Martinez, Michael Sablatschan, Antonio Skarmeta, and Hermann Hellwagner. Towards User-driven Adaptation of H.264/SVC Streams. *Proceedings of EuroITV2010*, pages 289–292, 2010.

[122] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. Technical report, Internet Engineering Task Force, January 2003. Standard, RFC 3448.

[123] Tornado Web Server. http://www.tornadoweb.org/en/stable/.

[124] Nginx - high performance load balancer, web server & reverse proxy. https://www.nginx.com.

[125] SmartFire project. http://eukorea-fire.eu/.

[126] Squid: Optimising Web Delivery. http://www.squid-cache.org.

[127] Openvswitch - Production Quality, Multilayer Open Virtual Switch. http://openvswitch.org.

[128] Bitmovin: MPEG-DASH Players High Streaming Quality. https://www.bitmovin.com/bitdash-mpeg-dash-player/.

[129] Floodlight project. http://www.projectfloodlight.org/floodlight/.

[130] NEPI new generation. https://nepi-ng.inria.fr/.

[131] Language Consortium. P4 16 Language Specification - version 1.0.0. pages 1–104, 2017.

[132] Named-data project codebase. https://named-data.net/codebase/platform/.

[133] Community Information Centric Networking. https://wiki.fd.io/view/Cicn#Introduction.