QoS management for WebRTC : loose coupling strategies

Ewa Czeslawa Janczukowicz

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UNIVERSITE
BRETAGNE
LOIRE

THÈSE / IMT Atlantique

sous le sceau de l'Université Bretagne Loire

pour obtenir le grade de

DOCTEUR DE IMT Atlantique

Mention : Informatique

École Doctorale Matisse

Présentée par

Ewa Janczukowicz

Préparée au département Systèmes Réseaux, Cybersécurité et Droit du numérique
Laboratoire Irisa

Gestion de la qualité de service pour WebRTC: Stratégies de couplage lâche

---

QoS Management for WebRTC : Loose Coupling Strategies

Thèse soutenue le 13 mars 2017
devant le jury composé de :

Noëmie Simoni
Professeure émérite, Telecom ParisTech / Présidente

Jean-Charles Grégoire
Professeur, INRS-Centre EMT - Canada / Rapporteur

Pascal Lorenz
Professeur, Université de Haute-Alsace / Rapporteur

Abdelhamid Mellouk
Professeur, Université Paris-Est Créteil / Examinateur

Ahmed Bouabdallah
Maître de Conférences, IMT Atlantique / Examinateur

Arnaud Braud
Ingénieur de recherche, Orange Labs / Examinateur

Stéphane Tuffin
Ingénieur de recherche, Orange Labs / Examinateur

Jean-Marie Bonnin
Professeur, IMT Atlantique / Directeur de thèse

Remerciements

Tout d'abord j'aimerais remercier Jean-Marie Bonnin, Ahmed Bouabdallah, Arnaud Braud et Stéphane Tuffin pour leur travail d'encadrement tout au long de ces trois années. Je voudrais aussi remercier Gaël Fromentoux et Xavier Marjou pour leurs conseils et leur soutien.

Je remercie également Jean-Charles Grégoire et Pascal Lorenz pour avoir accepté d'être rapporteurs, et Noëmie Simoni et Abdelhamid Mellouk pour leur rôle d'examinateurs dans mon jury de thèse.

J'adresse également mes remerciements aux nombreuses personnes qui m'ont aidé pendant ces travaux : l'équipe ARC (Imed Allal, Nicolas Bihannic , Marc Bouillon, Romuald Corbel, Luc Le Beller, Olivier Le Grand, Nathalie Omnes, Joël Penhoat, Veronica Quintuna, Farah Slim, Emile Stephan), l'équipe SINA dirigé par Eric Debeau, mais aussi Jean-Michel Portugal, Simon Becot, Eric Paillet, Laura Pirio, Gwenaël Le Lay, Patrice Odermatt, Marc Claquin, Michel Quinquis , Claude Lamblin, Jean-Yves Le Saout, Antonin Marechal, Alassane Samba, William Diego, Monique Davai et le département RSM de Télécom Bretagne à Rennes.

Je voudrais remercier l'équipe IOS dirigé par Tony Capo-Chichi pour leur accueil, et Nabil Charkani et Alain Henry pour leur accueil dans les départements NCA et NCS.

Merci aussi à Pascal qui était le meilleur collègue de bureau et à Kitty qui me motivait pour faire du sport chaque midi.

Un grand merci à mes parents et à ma sœur pour leurs encouragements. Enfin, merci à toi Joachim, pour ton soutien infini.

Abstract

Real-time communication services have evolved in the recent years, notably more communication services are offered by the Web players, e.g. Google Hangouts or WhatsApp. We have identified two approaches that are currently used when providing real-time communication services: *Over-The-Top* (OTT), and *Network Service Provider* (NSP) solutions. OTT solutions are designed to work over the Internet and are independent from network assistance, while NSP solutions are tightly coupled with networks that they operate.

OTT solutions use best-effort Internet delivery, so they do not benefit from any specific network operator assistance. They rely on mechanisms built into the endpoints, e.g. congestion control or adaptive codecs, and adapt to underlying network fluctuations. Nevertheless, it is questionable whether OTT approach is enough to provide acceptable quality of communication regardless the network conditions. Therefore, can network assistance be used to improve the quality of OTT real-time communication services?

To address this question, we study the existing solutions. First, we analyse OTT solutions with a focus on WebRTC, along with Google Congestion Control (GCC). Secondly, we discuss how NSP solutions benefit from network assistance in order to ensure the end-to-end QoS for their communication services. Finally, we identify three loose coupling strategies that leverage network mechanisms for improving OTT communication services quality: NSP driven, OTT driven and User/Enterprise driven. For each of these strategies, we propose an example of implementation.

We have chosen to verify the pertinence of these coupling strategies in the context of traffic management. Firstly, we show that the current Internet engineering practices are not designed for real-time traffic, but are optimized for TCP traffic, i.e aim at assuring high link utilization. Secondly, we identify the network requirements for WebRTC, and perform a survey of existing queuing mechanisms. It leads to the identification of two approaches of traffic management solutions adapted to WebRTC traffic: 1) aiming at assuring lower queuing delays regardless the traffic or 2) isolating the sensitive traffic. For each of these approaches, we select a combination of queuing mechanisms.

The evaluation of these solutions is done for wireline access networks (uplink ADSL and fiber). We study the impact of identified traffic management solutions on WebRTC and its congestion control mechanism, notably in the presence of long-lived TCP flows. We use various metrics, including the application layer measurements strongly linked with quality. The obtained results show that current practices in queuing mechanisms configurations are not well adapted to the WebRTC traffic. Furthermore, the proposed solutions ensure more fairness between WebRTC and TCP flows and consequently enable avoiding WebRTC traffic starvation and improve the overall quality of the communication.

In the final analysis, the evaluated traffic management solutions are positioned in the context of identified coupling strategies. Based on this assessment, we provide recommendations of improving WebRTC quality with the assistance of the NSP. We also discuss the research perspectives in relation to the ongoing works.

Résumé

Les services de communication temps réel ont beaucoup évolué ces dernières années, avec notamment la multiplication des services proposés par les acteurs du Web, tels que Google Hangouts ou WhatsApp. Nous avons identifiés deux approches qui sont utilisées pour fournir des solutions de communication temps réel : les solutions *Over-The-Top* (OTT), et *Network Service Provider* (NSP). Les solutions OTT reposent sur Internet et sont indépendantes de l'assistance des opérateurs réseaux, alors que les solutions NSP dépendent des réseaux qu'ils exploitent.

Les solutions OTT utilisent l'Internet «best-effort», et ne bénéficient d'aucune assistance particulière de la part des opérateurs réseaux. Elles s'adaptent aux fluctuations du réseaux grâce à des mécanismes tels que le contrôle de congestion ou les codecs adaptatifs. Néanmoins, il est discutable que l'approche OTT soit suffisante pour fournir une qualité de service de communication acceptable quelles que soient les conditions réseaux. Dès lors, est-il possible d'utiliser l'assistance réseau pour améliorer la qualité de service des solutions OTT ?

Pour traiter cette question, nous étudions tout d'abord les solutions OTT, et particulièrement la technologie WebRTC ainsi que le Google Congestion Control (GCC). Ensuite, nous analysons comment les solutions NSP exploitent l'assistance réseau afin d'assurer les mécanismes de QoS de bout en bout. Enfin, nous identifions trois stratégies de couplage lâche qui permettent de tirer parti des mécanismes réseaux pour améliorer la qualité de service des solutions OTT : de type NSP, OTT et User/Enterprise.

Nous vérifions la pertinence de ces stratégies de couplage dans le contexte de la gestion du trafic. Tout d'abord, on montre que les pratiques d'ingénierie d'Internet actuelles ne sont pas conçues pour le trafic en temps réel, mais sont adaptées au trafic TCP. On identifie les besoins du trafic WebRTC et on réalise une étude des mécanismes de queuing. Ensuite, on identifie deux approches de gestion du trafic adaptées à WebRTC : 1) qui assure des délais d'attente courts quel que soit le trafic ou 2) qui isole le trafic sensible. Pour chaque approche, on choisit une combinaison des mécanismes de queuing adaptés.

On évalue les solutions proposées pour les réseaux d'accès filaire (uplink, ADSL et fibre). On étudie l'impact des solutions identifiées de gestion du trafic, sur WebRTC et son mécanisme de contrôle de congestion, notamment en présence du trafic TCP long-lived. Pour cela, on utilise des mesures variées, y compris les mesures au niveau application liées à la qualité. Les résultats obtenus montrent que les pratiques actuelles de gestion du trafic ne sont pas adaptées au trafic WebRTC. De plus, les solutions proposées assurent plus d'équité entre le trafic WebRTC et TCP. En conséquence, elles permettent d'éviter que le trafic WebRTC soit affamé et elles améliorent la qualité de communication.

Enfin, ces solutions de la gestion du trafic identifiées sont positionnées dans le contexte des stratégies de couplage proposées. A partir de là, on fournit des recommandations pour améliorer la qualité WebRTC avec l'assistance du NSP. On donne également des perspectives de recherche en lien avec les travaux en cours.

Streszczenie

Usługi komunikacyjne w czasie rzeczywistym ewoluowały w ostatnich latach. Coraz więcej usług oferują przedsiębiorstwa z branży internetowej. W związku z tym proponujemy podział tych usług na: 1) usługi typu *Over-The-Top* (OTT), usługi Internetowe niezależne od wsparcia operatora telekomunikacyjnego oraz 2) usługi typu *Network Service Provider* (NSP), ściśle zależne od sieci zarządzanych przez danego operatora telekomunikacyjnego.

Rozwiązania OTT korzystają z usług typu best-effort, dlatego też nie mają wsparcia ze strony operatora telekomunikacyjnego. Pomagają im mechanizmy znajdujące się na terminalach użytkowników, np. mechanizmy zapobiegające przeciążeniu albo kodeki. Niemniej jednak pojawiają się pytania, czy te rozwiązania są wystarczające do zaoferowania wystarczającej jakości, niezależnie od stanu danej sieci. Czy jest możliwe, żeby wykorzystać wsparcie sieci do ulepszenia jakości usług OTT?

W celu odpowiedzi na poruszone wyżej kwestie, analizujemy obecne rozwiązania. Najpierw, skupiamy się na technologii OTT na przykładzie WebRTC razem z mechanizmem zapobiegania przeciążeniu - Google Congestion Control (GCC). Następnie opisujemy w jaki sposób technologie NSP korzystają z mechanizmów sieciowych w celu ulepszenia jakości komunikacji „end to end". Ostatecznie identyfikujemy trzy strategie, które pozwalają na wykorzystanie mechanizmów sieciowych w celu polepszenia jakości komunikacji OTT, to jest strategie typu: OTT, NSP i User/Enterprise.

Następnie sprawdzamy słuszność zaproponowanych strategii w kontekście zarządzania ruchem. Pokazujemy, że obecne praktyki zarządzania ruchem w Internecie nie są zaprojektowane dla komunikacji w czasie rzeczywistym, ale skupiają się na zwiększeniu przepustowości, toteż są korzystne dla strumieni TCP. Definiujemy potrzeby strumieni WebRTC i analizujemy obecnie istniejące mechanizmy kontrolne i algorytmy kolejkowania pakietów. Na podstawie tych prac proponujemy dwa rozwiązania zarządzania ruchem: 1) zmniejszające czas przebywania pakietów w kolejce niezależnie od rodzaju ruchu lub 2) izolujące ruch wrażliwy na przeciążenia i opóźnienia. Dla każdego rozwiązania proponujemy odpowiednią kombinacje algorytmów kolejkowania pakietów.

Oceniamy zaproponowane rozwiązania dla technologii dostępu do Internetu typu ADSL i sieci optycznych (w kierunku upstream). Weryfikujemy wpływ różnych rozwiązań zarządzania ruchem na WebRTC, a w szczególności na mechanizm zapobiegania przeciążeniu w obecności strumieni TCP. W tym celu wykonujemy różne pomiary pozwalające na efektywną ocenę jakości. Otrzymane rezultaty pokazują, że obecne podejścia do zarządzania ruchem nie są dostosowane do technologii WebRTC. Ponadto, zaproponowane rozwiązania zapewniają bardziej sprawiedliwy podział zasobów sieciowych między TCP i WebRTC, w związku z tym polepszają jakość WebRTC.

Ostatnim etapem jest sprawdzenie zależności między proponowanymi rozwiązaniami zarządzania ruchem a zdefiniowanymi strategiami powiązania OTT i NSP. Z tej analizy wyciągamy wnioski i oferujemy rekomendacje w celu polepszenia jakości WebRTC przy wsparciu mechanizmów sieciowych. Ponadto, omawiamy perspektywy badawcze.

Personal Bibliography

Publications

(2016) E. Janczukowicz, A. Braud, S. Tuffin, A. Bouabdallah, JM. Bonnin, "Evaluation of network solutions for improving WebRTC quality", 24th Conference on Software, Telecommunications and Computer Networks (SoftCOM 2016), September 22-24, 2016, Split, Croatia

(2016) A. Marechal, E. Janczukowicz, "TURN Servers Impacts Over WebRTC QoE in 4G Network ", 19th International ICIN Conference - Innovations in Clouds, Internet and Networks (ICIN 2016), March 1-3, 2016, Paris, France

(2015) E. Janczukowicz, A. Braud, S. Tuffin, G. Fromentoux, A. Bouabdallah, JM. Bonnin, "Specialized network services for WebRTC: TURN-based architecture proposal", International Workshop on All-Web real-time Systems (AWeS 2015) in conjunction with ACM EuroSys 2015, April 21, 2015, Bordeaux, France

(2014) E. Janczukowicz, S. Tuffin, A. Braud, A. Bouabdallah, G. Fromentoux, JM. Bonnin, "Approaches for offering QoS and specialized traffic treatment for WebRTC", 20th EUNICE/IFIP EG 6.2, 6.6 International Workshop (EUNICE 2014), September 1-5, 2014, Rennes, France

(2015) E. Janczukowicz, A. Bouabdallah, A. Braud, S. Tuffin, JM. Bonnin, "Firefox OS Ecosystem - ambitions and limits of an open source operating system for mobile devices", Chapter 17 in Handbook of Research on Next Generation Mobile Communication Systems. Advances in Wireless Technologies and Telecommunication, IGI Global, pp.440 - 465, 2015

(2014) E. Janczukowicz, A. Bouabdallah, A. Braud, G. Fromentoux, JM. Bonnin, "Improving Mozilla's In-App Payment Platform", 10th International Conference on Open Source Systems (OSS 2014), May 6-9, 2014, San Jose, Costa Rica

Patents

(2015) E. Janczukowicz, X. Marjou, G. Fromentoux, "Warning Based on STUN messages"

(2014) E. Janczukowicz, X. Marjou, G. Fromentoux, "STUN for WebRTC flows redirection - STUPENDO" (WO/2016/097534)

Contributions to reThink project

(2015) R. Copeland, A. Bouabdallah, I. Jave, E. Paillet, S. Bécot, E. Janczukowicz, K. Corre, JM. Crom, P. Chainho, F. Beierle, S. Göndör, F. Luart, A. Al-Hezmi, A. Corici, M. Emmelmann, R. Lopes Pereira, R. Chaves, N. Santos, "reTHINK, Deliverable D2.1 Framework Architecture Definition", July 2015

(2015) P. Chainho, M. Pedrosa, V. Silva, M. Mesquita, L. Duarte, J. Pinheiro, M. Emmelmann, A. Corici, A. Cheambe, A. Portabales, S. Troncoso; Y. Riobó, A. Vallée, F. Luart, S. Druesedow, F. Oberle, I. Friese, E. Janczukowicz, S. Bécot, K. Corre, E. Paillet, R. Lopes Pereira, R. Chaves, N. Santos, R. Copeland, "Deliverable D3.1 Hyperty Runtime and Hyperty Messaging Node Specification", September 2015

Research report

(2013) E. Janczukowicz, "Firefox OS Overview", Télécom Bretagne - Research Report - RR-2013-04-RSM - 2013.

Contents

List of Figures

List of Tables

List of Abbreviations

ABU	Average Bandwidth Utilization
ACL	Access Control List
AF	Assured Forwarding
API	Application Programming Interface
AQM	Active Queue Management
ARCEP	Electronic Communications and Postal Regulatory Authority
ARED	Adaptive Random Early Detection
B2C	Business to Client
BE	Best-Effort
BEREC	Body of European Regulators for Electronic Communications
CDN	Content Delivery Networks
Codel	Controlled Delay
CPE	Customer Premises Equipment
CSP	Communication Service Provider
DCTCP	Data Center TCP
DRR	Deficit Round Robin
DSCP	Differentiated Services Code Point
DSL	Digital Subscriber Line
DTLS	Datagram Transport Layer Security
ECN	Explicit Congestion Notification
EF	Expedited Forwarding
EPC	Evolved Packet Core
EPS	Evolved Packet System
FEC	Forward Error Correction
FIFO	First In First Out
pFIFO	packet FIFO
bFIFO	byte FIFO
FQ Codel	Fair (or Flow) Queue Controlled Delay
FttH	Fiber to the Home
FTTx	Fiber To The...
GCC	Google Congestion Control
HTB	Hierarchical Token Bucket

KPI	Key Performance Indicator
L4S	Low Latency, Low Loss, Scalable Throughput
LAN	Local Area Network
LLT	Latency Loss Tradeoff
LTE	Long-Term Evolution
MTG	Mobile Throughput Guidance
NACK	Negative Acknowledgment
NADA	Unified Congestion Control Scheme for Real-Time Media
NIC	Network Interface Controller
NSP	Network Service Provider
OTT	Over-The-Top
P2P	Peer-To-Peer
PDN-GW or P-GW	Packet Data Network Gateway
PIE	Proportional Integral controller Enhanced
PoC	Proof of Concept
QCI	QoS Class Identifier
QoS	Quality of Service
QUIC	Quick UDP Internet Connection
RAN	Radio Access Network
RED	Random Early Detection
RMCAT	RTP Media Congestion Avoidance Techniques
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTT	Round-Trip Time
SBC	Session Border Controller
A-SBC	Access SBC
I-SBC	Interconnect SBC
SCReAM	Self-Clocked Rate Adaptation for Multimedia
SCTP	Stream Control Transmission Protocol
SDN	Software Defined Networking
SFQ	Stochastic Fairness Queueing
SLA	Service Level Agreement
SPUD	Substrate Protocol for User Datagrams
SRTCP	Secure Real-time Control Protocol
SRTP	Secure Real-time Transport Protocol
TC	Traffic Control
TRAM	TURN Revised and Modernized
UC	Unified Communications
VC	Virtual Circuit
VoIP	Voice over IP
WebRTC	Web Real-Time Communication

Resumé étendu

Contexte des travaux

Les services de communication temps-réel ont beaucoup évolué ces dernières années, avec notamment la multiplication des services proposés par les acteurs du web, tels que Google Hangouts ou WhatsApp.

Nous pouvons désormais observer une fragmentation des services de communication selon deux segments :

- Les services télécom traditionnels, tels que les services voix sur fixe et mobile, SMS, MMS.
- Les services Web, tels que la voix sur IP (VoIP), les réseaux sociaux, les services mail ou chat.

Dans cette thèse, nous nous sommes concentrés sur les services de communication en temps réel. Ainsi, nous avons identifiés deux approches qui sont utilisées pour fournir ce type de solutions : les solutions Over-The-Top (OTT) et Network Service Provider (NSP). Les solutions OTT reposent sur l'Internet et sont indépendantes de l'assistance fournie par les opérateurs réseaux, alors que les solutions NSP dépendent des réseaux qu'ils exploitent.

Plus particulièrement, les solutions OTT utilisent l'Internet «best-effort», et ne bénéficient d'aucune assistance particulière de la part des opérateurs réseaux. Elles s'adaptent aux fluctuations des réseaux grâce à des mécanismes tels que le contrôle de congestion ou les codecs adaptatifs. Néanmoins, il est discutable que l'approche OTT soit suffisante pour fournir une qualité de service de communication acceptable quelles que soient les conditions réseaux. Dès lors, est-il possible d'utiliser l'assistance réseau pour améliorer la qualité de service des solutions OTT ?

WebRTC et Google Congestion Control

Pour traiter cette question, nous étudions tout d'abord les solutions OTT, et plus particulièrement la solution WebRTC (Web Real-Time Communication) ainsi que le Google Congestion Control (GCC).

Nous avons choisi de travailler sur WebRTC car c'est une solution émergente qui permet une communication directe entre les navigateurs Web de type audio, vidéo ou transfert de données. Elle est déployée dans des solutions commerciales, mais est également étudiée dans un cadre académique. De plus, WebRTC est particulièrement bien documenté, du fait de sa normalisation en cours par les instances IETF et W3C, contrairement à d'autres solutions OTT qui sont souvent propriétaires et par conséquent, manquent d'une documentation détaillée.

Dans WebRTC le plan de signalisation et séparé du plan media. Il existe deux principaux modes de communication possibles : le mode Peer-To-Peer (P2P) et le mode utilisant un relai media, comme présenté sur la figure 1.

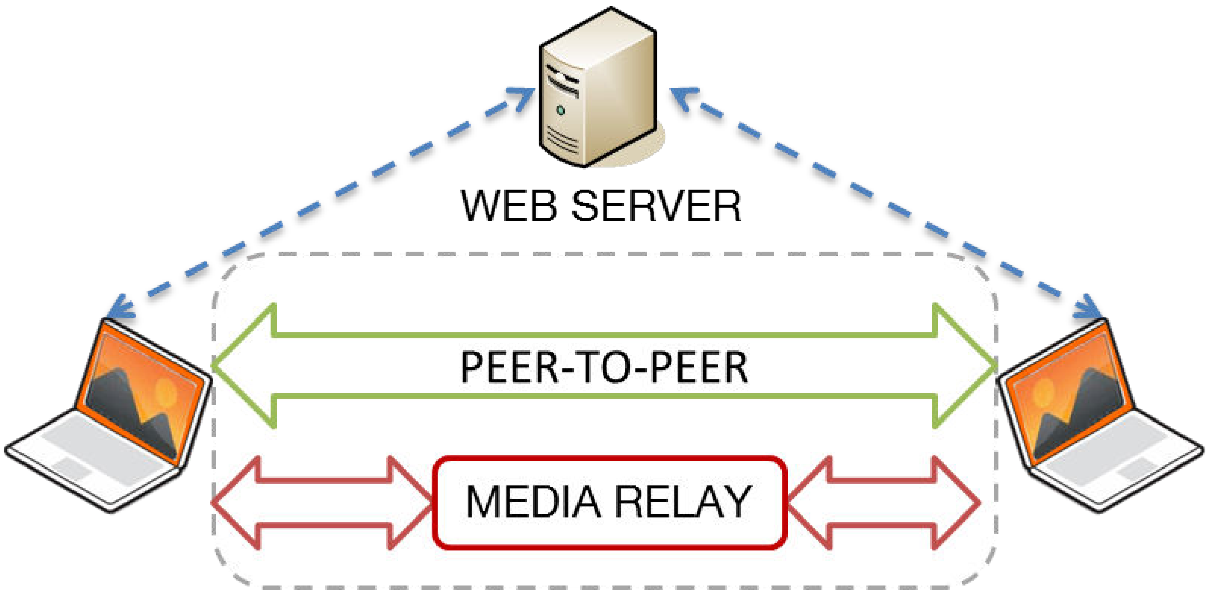


Figure 1 – WebRTC - modes de communication

WebRTC étant une solution de type OTT, elle n'utilise donc pas de mécanismes réseau pour gérer la qualité. En revanche, WebRTC peut se baser sur des mécanismes du niveau applicatif, tels que les mécanismes de contrôle de congestion, les codecs adaptatifs et les mécanismes de gestion de pertes de paquets. Dans cette thèse, nous nous focalisons principalement sur les mécanismes de congestion.

A l'heure actuelle, trois algorithmes de contrôle de congestion sont proposés au sein de RMCAT, un groupe du travail de l'IETF. Néanmoins, nous avons décidé de se focaliser sur Google Congestion Control (GCC) pour WebRTC, parce qu'il est le plus avancé et le plus déployé.

Dans le cadre de nos travaux, nous nous intéressons à l'impact de GCC sur la qualité perçue par les utilisateurs. Nous allons également nous intéresser à la manière dont cet algorithme pourrait bénéficier de l'assistance des réseaux.

Stratégies de couplage lâche

Nous avons étudié quel framework collaboratif pourrait être utilisé pour permettre une coopération entre les acteurs NSP et OTT. Ainsi nous avons étudié s'il était possible d'utiliser l'assistance réseau pour améliorer la qualité de service des solutions OTT. Nous avons proposé trois stratégies de couplage lâche qui permettent de tirer parti des mécanismes réseaux pour améliorer la qualité de service des solutions OTT :

- NSP-driven (stratégie orientée NSP),
- OTT-driven (stratégie orientée OTT),
- User/Enterprise-driven (stratégie orientée User/Enterprise).

Pour chaque de cette stratégies, nous précisons un exemple d'implémentation.

NSP-driven

Dans la stratégie NSP-driven nous voulons améliorer la qualité en général, quelle que soit l'application.

Ici, nous proposons des solutions neutres qui se focalisent sur le traitement optimal de différents types de trafic. Le but est d'obtenir un bon compromis entre faible latence et utilisation efficace du lien.

OTT-driven

Dans la stratégie OTT-driven nous voulons améliorer la qualité pour un service OTT particulier.

Ici, nous proposons une solution basée sur l'utilisation des relais media avec brokering. Cette solution est inspirée de l'approche NSP implémenté dans la solution de VoIP managé. Les relais média, spécifiques à WebRTC, sont utilisés pour isoler le trafic WebRTC du reste du trafic de type best-effort. Afin d'assurer le service global, un broker est proposé en tant que point de contact unique. Le principe de cette solution est illustré sur la figure **??**.

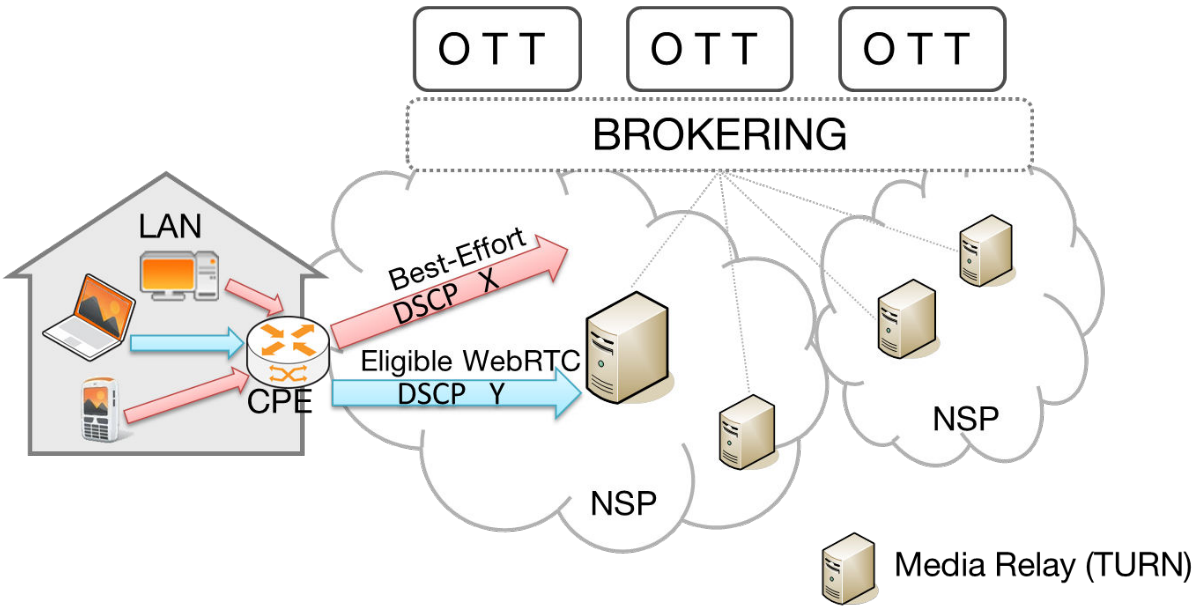


Figure 2 – Stratégie OTT driven

User/Enterprise driven

Dans la stratégie User/Enterprise-driven nous voulons améliorer la qualité pour un service OTT particulier ou pour un utilisateur donné.

Ici, nous proposons d'utiliser une solution d'«enterprise policies», qui donne à une entreprise la possibilité de gérer les services de communication utilisés par ses employés. Un administrateur réseau dans une entreprise peut configurer un marquage DSCP au niveau des applications, qui sera ultérieurement pris en compte au niveau de «Customer Premises Equipment» (CPE) afin d'assurer une différentiation du trafic. Le principe de cette solution est illustré sur la figure 3.

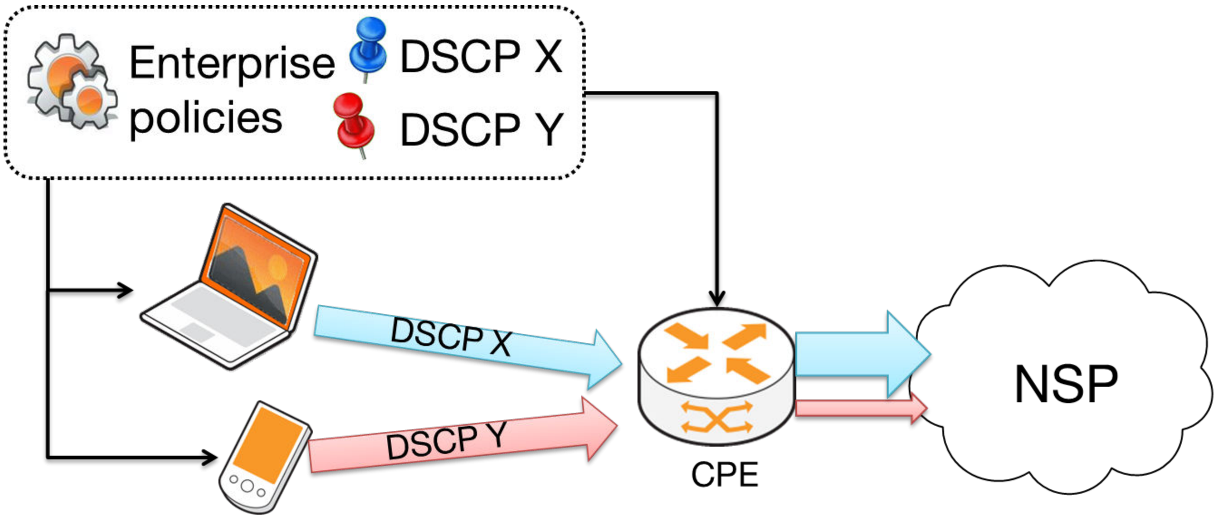


Figure 3 – Stratégie User/Enterprise driven

Gestion du trafic adaptée au WebRTC

Les stratégies de couplage lâche que nous avons proposées, sont basés sur la différentiation du trafic WebRTC du reste de trafic de type best-effort. Pour cette raison, nous avons décidé d'étudier la gestion du trafic adaptée au trafic en temps-réel, notamment WebRTC.

Tout d'abord, on montre que les pratiques d'ingénierie d'Internet actuelles ne sont pas conçues pour le trafic en temps réel, mais sont adaptées au trafic TCP. Cela veut dire que généralement dans les équipements réseaux les buffers larges sont configurés afin de minimiser la perte de paquets et de maximisez l'utilisation des liens. Cette approche fonctionne bien pour le trafic non-temps réel, comme TCP qui utilise un mécanisme de contrôle de congestion basé perte de paquet. Cependant, le trafic temps-réel est d'une nature différente. Pour ce trafic il est plus important de garantir un délai bas, que d'assurer aucune perte de paquet.

Ainsi, après avoir identifié les besoins du trafic WebRTC et nous avons réalisé une étude des mécanismes de queuing. Ensuite, nous avons identifié deux approches de gestion du trafic adaptées à WebRTC :

- Target delay based queuing mechanisms – qui assure des délais d'attente courts quel que soit le trafic.
- Rate and queue length based classful queuing mechanisms - qui isole le trafic sensible. Ici, on utiliser un mécanisme HTB pour isoler un trafic WebRTC du reste de trafic.

On évalue les solutions proposées pour les réseaux d'accès filaire (uplink, ADSL et fibre). Dans un premier temps nous évaluons les pratiques actuelles de gestion du trafic. Ensuite, nous étudions l'impact des solutions identifiées de gestion du trafic, sur WebRTC et son mécanisme de contrôle de congestion, notamment en présence du trafic TCP long-lived.

Nous évaluons trois types de configurations :

- Droptail – basé sur l'approche best-effort, donc un file de type FIFO.
- Target delay based queuing mechanisms, pour lesquels on utilise deux mécanismes de queuing : FQ CoDel et PIE.

- Rate and queue length based classful queuing mechanisms pour lequels on utilize HTB pour isoler un trafic WebRTC du reste de trafic.

Pour l'analyse, on utilise des mesures variées, y compris les mesures au niveau application liées à la qualité. Ainsi, on récupère des mesures :

- au niveau réseau : channel utilization,
- au niveau application : video flow bitrate, RTT, packet loss, video flow frame rate and frame width.

Un exemple des résultats obtenus est présenté sur la figure 4.

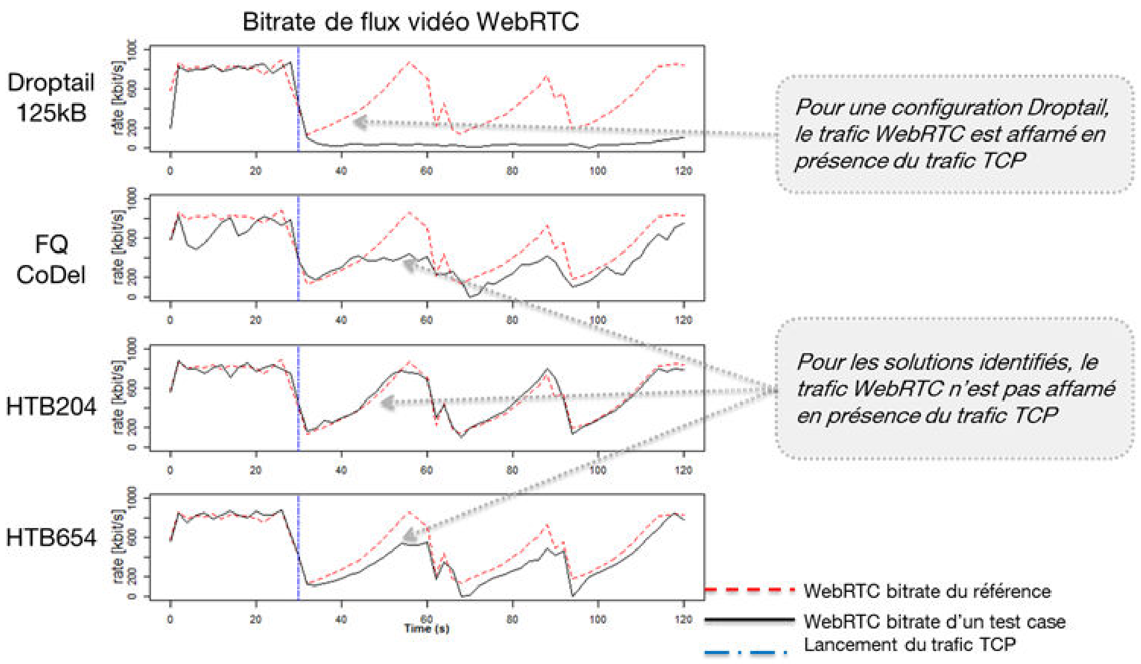


Figure 4 – Exemple des résultats

En résumé, les résultats obtenus montrent que les pratiques actuelles de gestion du trafic ne sont pas adaptées au trafic WebRTC. De plus, les solutions proposées assurent plus d'équité entre le trafic WebRTC et TCP. En conséquence, elles permettent d'éviter que le trafic WebRTC soit affamé et elles améliorent la qualité de communication.

A partir de cette évaluation, on fournit des recommandations pour améliorer la qualité WebRTC avec l'assistance du NSP. On suggère également des améliorations pour le mécanisme GCC. Enfin, les solutions de la gestion du trafic identifiées sont positionnées dans le contexte des stratégies de couplage proposées. La stratégie NSP-driven peut bénéficier de l'approche target delay based. Tandis que les stratégies OTT-driven et User/Enterprise driven peuvent bénéficier de l'approche rate and que length based classful.

Conclusion

Le but de cette thèse a été d'étudier la possibilité de tirer parti des mécanismes réseau afin d'améliorer la qualité des solutions OTT, de type WebRTC. Les principaux apports de cette thèse sont :

- L'état de l'art sur les écosystèmes NSP et OTT, notamment concernant les mécanismes de gestion de qualité des services de communication.
- La proposition des stratégies de couplage lâche afin de permettre une collaboration entre les acteurs de type NSP et OTT.
- La proposition d'une gestion du trafic adaptée au WebRTC qui soit, assure des délais d'attente courts quel que soit le trafic, soit isole le trafic sensible.
  - Proposition d'environnement et de méthodologie de tests afin de vérifier la pertinence des mécanismes identifiés.
  - Evaluation des solutions de gestion du trafic adapté au WebRTC.
  - Positionnement des résultats obtenus par rapport aux études existantes.
  - Suggestion des améliorations pour GCC.
- Positionnement des solutions de gestion du trafic par rapport aux stratégies de couplage lâche initialement identifiés.

Enfin, on donne des perspectives de recherche en lien avec les travaux en cours, tel que les travaux sur un partage optimal des ressources réseaux ou des aspects de cross-layer.

Chapter 1

Introduction

Contents

1.1 Changes to communication services: Webco vs Telco

In the recent years we could observe an important growth of the Internet. In fact, a Cisco White Paper (Cisco 2016) reports that from 2015 to 2020 the global IP traffic will increase threefold and from 2005 to 2020 it will have increased nearly a hundredfold.

The ubiquity and decreasing cost of the Internet have led to some major changes in the real-time communication ecosystem. The new types of technologies, applications and players have emerged and the ratio of forces between different actors has changed. Additionally, the way users interact with provided communication services has evolved. The IBM report (Nelson & van den Dam 2015) highlights that nowadays the communication services are fragmented into traditional telecom services (e.g. fixed and mobile voice, SMS, MMS) and Web services (e.g. *Voice over IP* (VoIP), social networking, e-mail, chat).

In fact, the Web services are the new types of communication services that differ from the legacy telecommunication solutions in both technological and economic aspects. These services provide their functionalities by using the available best-effort Internet, without any specific network operator assistance. Hence more and more communication services are offered not by Network Service Providers but by Web Communication Services Providers, e.g. Skype, Facebook, Google. Furthermore, the omnipresent Web real-time communication solutions are much more than only traditional voice services. They are more versatile and can be a part of different interactive services, such as social networks or gaming.

The IBM report (**Nelson & van den Dam 2015**) also indicates that there is an overall growth of usage of communication services, but this growth concerns mostly Web communication services, whereas growth in traditional services is less significant. To emphasize this fact, an example of France is given where from 2005 to 2010, the growth of volume of mobile and fixed calls was estimated to only 9%, whereas over the same period of time, the growth of volume of Web communication services was estimated to 211%.

Given these points, there can be two types of *Communication Service Providers* (CSP) (**Minerva 2013**), (**Bertin et al. 2011**):

- Telco CSP, also called *Network Service Provider* (NSP) operating a network and providing communication services.
- Webco CSP, also called *Over-The-Top* (OTT) Web players providing communication services and relying only on the Internet, not operating any network.

Webco and Telco services are based on different technological approaches and distribution models (**Bécot et al. 2015**). In this chapter technological and economic overview of these two ecosystems is given.

1.1.1 Why is Webco different than Telco?

According to Minerva (**Minerva 2013**) two different views on communication services can be highlighted :

- For Telcos, services are an extension of connectivity and traditional communication means, so they deeply rely on networks.
- For Webcos, services can be provided by means of Web capabilities, OTTs treat network as a pipe providing best-effort connectivity and for them the value is on the edges of the network.

Telcos have used a traditional telephony approach when designing their communication services. In this approach, each NSP provides a service to its own subscribers, and later interconnects with other NSPs to ensure the end-to-end functionality. Therefore Telco solutions apply only to a certain number of users, i.e. given NSP subscribers, and consequently they are limited to a certain territory.

When providing communication services, Telco strongly rely on networks and consequently ensure a quality equivalent to the traditional telephony. Furthermore, troubleshooting and supervision is necessary as NSPs must comply with regulatory and contractual aspects. For instance, according to French *Electronic Communications and Postal Regulatory Authority* (ARCEP), in France since 2010 any NSP that has more than 100 000 broadband subscribers has to publish every three months the results of measurements of network quality concerning access to services like telephony, Internet and TV (**Arcep nown**).

Additionally, interoperability between NSPs plays an important role in the Telco ecosystem. For example, in France, there are regulatory aspects concerning interconnection so that subscribers of different NSPs can continue to communicate with each other (**Arcep 2006**). This interoperability requires a standardisation effort, which makes any deployment time consuming and innovations take time to be implemented.

Webcos do not have to deal with network infrastructure deployment, but they benefit from Internet global connectivity and focus on service functionality. Webco solutions are proprietary, so they are tied to given CSPs and do not interoperate with applications offering a similar service, e.g. User A using WhatsApp cannot communicate directly with User B using Google Hangouts. Furthermore, OTT services often use existing programming tools and Internet-based services, that make launching global services easier, less expensive and encourage innovation (**Andreessen 2011**).

A typical Web service is deployed on a server (or multiple servers) and accessible across numerous access networks. This approach became possible thanks to higher network bandwidth that decreased network latency (**O'Connell 2013**). Indeed, OTT services are meant to work over any network and can be used by any users with Internet access, so they benefit from a global reach.

Webcos rely on underlying networks for best-effort connectivity. Application and network layers are independent, so OTT cannot benefit from any specific network support or network layer information. They also rely on information provided by other Web services or on client-based data like e.g. browser regional preferences (**O'Connell 2013**). Consequently, Webco only use application layer mechanisms to adapt to underlying network fluctuations, with for instance, adaptive codecs or congestion mechanisms.

1.1.2 How these differences influence the business value chain?

Telcos were used to directly monetize their real-time communication solutions. Voice and SMS services used to be their principal source of revenue, but in recent years the telecom industry has become more data and connectivity centric (**Nelson & van den Dam 2015**). The Financial Times (**Thomas 2013**) estimates that in 2013 telecom service revenue decreased by 3% in the major European countries and that there had been a decline of the overall service revenues by 21bn euro between 2008 and 2012 across the whole Europe.

The Capgemini report (**Schön et al. 2011**) highlights that there is a decline in traditional voice service revenues, but NSPs have to continue to invest in their infrastructures given that there is an increasing demand of data service. We find this conclusion also in the IBM report (**Nelson & van den Dam 2015**) that mentions that historically traffic demand and revenue followed the same trend, but in the past ten years their paths have diverged. Huawei, in their new Telco ecosystem analysis (**Huawei nown**), indicates that even if there is an increasing demand for data, the growth of Telco income is low, especially when compared to the Web players, notably OTT content distribution providers.

This changes are a driving force for innovation and consequently for changes to business models. Telcos end up investing into different new concepts, e.g. digital services like videos and music content distribution services (**Huawei nown**) or sectors such as healthcare, energy, automotive (**Schön et al. 2011**).

On the other hand, Webcos take advantage from the flat rate data plans, in which users take care of the network connectivity cost. Moreover, Webcos do not monetize their services in a traditional way. Instead, they often use asymmetric business models, so they often do not directly monetize their services, but can monetize ads or analytics (**Vakulenko 2013**).

Equally important is the way the users interact with provided services. Minerva, in his thesis (**Minerva 2013**), indicates that nowadays users not only consume services but also contribute to their development and testing, so a user-centered approach gains an importance.

The changes to the business value chain and especially a separation of network operation and services are the reason of a growing disintermediation of network operators. It may lead to the situation where NSPs would not deal anymore with every customer directly and would lose a large part of a direct business, i.e. business to customers (B2C). As a result, it may cause a redefinition of network operator's role in the value chain.

Over recent years, Telco have been working on making their networks more attractive in order to stay relevant in the business value chain. They started valorizing their network assets by offering network enablers and *Application Programming Interfaces* (APIs), e.g. SMS or click-to-call APIs (**Ramahandry et al. 2012**), (**Boyd 2015**).

As mentioned before, Telco services are limited to a given territory, so to impact the biggest number of users, there is a need of cooperation between different NSPs and in many cases some standardization effort. Cooperative efforts include among others API Exchange for GSMA by Apigee (**APIExchange nown**), (**Ramahandry et al. 2012**), Open Mobile Alliance API (**OMA nown**), (**Ramahandry et al. 2012**), and the most recent Open APIs by TM Forum (**TMForum nown**), (**Newman 2016**). Additionally, certain Telcos offer more proprietary solutions, e.g. Orange Developer APIs (**Orange nowna**), focusing on APIs for Identity, Payment, Communication and Internet of Things.

The road to success of Telco API work is not straightforward. The proposed solutions have difficulties to become widely adopted and are mostly criticized for their complexity and lack of adaptability (**Quayle 2012**), (**Chappell 2016**). Additionally there is a competition of APIs offered by Web giants, e.g. Facebook (**Facebook nown**) and Google (**Google nown**) that benefit from a global reach and are more developer friendly (**Levent-Levi 2013**).

Altogether, Telcos find it more difficult to monetize their services in a traditional way. They are also facing a competition of mostly free OTT services. Minerva, in his thesis (**Minerva 2013**), discusses that Telco have become less relevant and that networks are currently mostly seen as a pipe, i.e. just providing connectivity. Nevertheless, there is an increasing demand of data and consequently of network investment, but as we have mentioned above, Telco service revenues are not in line with this increasing demand of network resources.

1.2 Is it possible to leverage network assets to improve OTT communication services?

As we have shown, Web communication services differ from the legacy telecommunication solutions in technological and economic aspects. As a result, we could identify two approaches of providing the communication services: 1) NSP and 2) OTT.

Given the current evolution of communication services and the changes to the business value chain, it is interesting to study if it is possible and beneficial to couple these two approaches and address the following question: how can we leverage the network assets in order to propose improvements to OTT communication services?

We have identified three major NSP assets that could be helpful in addressing this question and enabling OTT and NSP coupling as seen on figure 1.1.

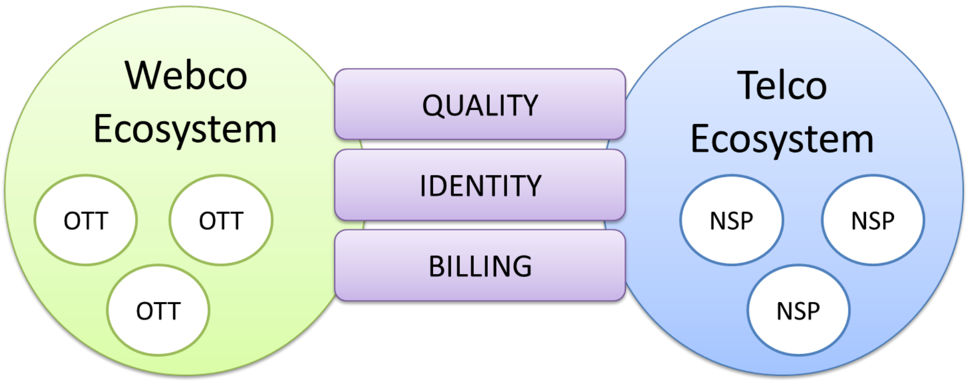


Figure 1.1 – Telco and Webco coupling

Quality

NSP solutions enable the cooperation between network operation and services. As a result, they ensure the end-to-end quality close to traditional telephony's. OTT solutions function without any specific network support and use mechanisms built into applications in the endpoints in order to adapt to underlying network fluctuations. Furthermore, current Internet engineering practices are not designed for real-time traffic, but are optimized for TCP traffic.

A research option could be to study how current network configurations interact with OTT adaptive mechanisms, and if it is possible to optimise these configurations in order to improve the quality of OTT real-time traffic that is independent from network resources.

Identity

Identity plays an important role in the Telco ecosystem, e.g. all NSPs use SIM cards to securely store information for subscriber authentication on the network. On the other hand, each OTT solution manages its own base of users, so that identity is linked with a given application.

It would be interesting to study whether Telco could provide more open and universal identity solution to OTTs. Furthermore, could a Telco position itself as a trusted identity provider?

Billing

NSPs use carrier billing for payment in addition to traditional payment methods, like bank transfer or credit card. The advantage of the carrier billing is that it can be used even if a given user does not have a bank account, e.g. with Orange Money[1]. OTT solutions are based on traditional payment methods but they also use different payment APIs, e.g. provided by PayPal[2] or certain NSPs, e.g. Orange[3].

[1]http://www.orange.com/en/Press-and-medias/Thematic-features/2015/SFM/In-Africa-Orange-Money-is-making-your-life-easier

[2]http://www.programmableweb.com/news/top-payments-apis-paypal-square-stripe-and-others/analysis/2015/03/11

[3]https://developer.orange.com/apis/direct-carrier-billing-europe/

Thus, carrier billing is an important Telco asset that could be beneficial for OTT solutions.

1.3 Problem statement

This thesis started as a general study on Telco APIs for OTT communication services and its aim was to evaluate the possibility of leveraging network assets in order to propose improvements to OTT real-time applications.

In this context, we have identified three possible network assets: quality, identity and billing. Nevertheless, we have decided to focus on one aspect, quality, which will be presented in this dissertation.

OTT solutions use best-effort Internet delivery, so they rely on mechanisms built into the endpoints, e.g. congestion control and adaptive codecs, to adapt to underlying network fluctuations. Nevertheless, it is questionable whether the OTT approach is enough to provide acceptable quality of communication regardless of network conditions.

Therefore, can network assistance be used in order to propose quality improvements to OTT real-time communication services?

We addressed this question by analysing one OTT solution, notably WebRTC project that is an emerging solution currently under standardization. WebRTC traffic benefits from the global Internet connectivity and is treated like any other Web traffic. To adapt to underlying network fluctuations, it uses mechanisms that probe networks in order to estimate the network parameters, e.g. Google Congestion Control (GCC) that calculates the sending rate based on the packet arrival time and packet loss. Nevertheless, we will show that the WebRTC traffic is vulnerable to starvation when competing for network capacity with flows using aggressive loss-based congestion control, notably long-lived TCP traffic.

In summary, the main focus of this thesis was to provide QoS management solutions for WebRTC by leveraging the proposed loose coupling strategies between OTT and NSP approaches.

Therefore, we have analysed the benefits of merging OTT and NSP approaches. We identified loose coupling strategies that use network mechanisms for improving OTT communication services quality. Furthermore, we have verified the pertinence of these coupling strategies in the context of traffic management, since, as we had shown, the current Internet engineering practices were not designed for real-time traffic, but were optimized for TCP traffic. It led to the identification of traffic management solutions adapted to WebRTC traffic, following two directions: 1) aiming at assuring lower queuing delays regardless the traffic or 2) isolating the sensitive traffic.

The implementation and evaluation of proposed solutions was done for wireline access networks (ADSL and fiber for uplink). We have studied the impact of various traffic management solutions on WebRTC, notably in the presence of long-lived TCP flows.

This work was done in the scope of Orange Labs research projects and in the scope of reThink European project.

The reThink[4] project aims at moving from telecom centric to Web centric P2P service architecture assuring dynamic and trusted relationship between distributed applications, i.e. *Hyperlinked Entities* (Hyperties).

In this project, I have managed a task on *Network QoS Policy Enforcement*, focusing on providing specialized network services for enabling QoS for real-time communication services.

In reThink, we proposed a global architecture for mobile and wireline networks. Nevertheless, in this thesis we focus on our contribution concerning the wireline access networks.

1.4 Document structure

In **Chapter 2** we provide a state of the art and motivations of using the WebRTC.

We present a thorough description of the WebRTC components and create an overview of ongoing standardisation efforts. We also give a survey of mechanisms used in WebRTC endpoints that adapt to underlying network fluctuations, e.g. Google Congestion Control mechanism and adaptive codecs.

In **Chapter 3** we provide a state of the art and motivations for implementing QoS for communication services.

We start by an overview of limits of best-effort delivery with a focus on OTT and real-time traffic. Then, we discuss how NSP solutions benefit from network assistance in order to ensure the end-to-end QoS for their communication services. Finally, we present the best practices in providing the managed VoIP mechanisms.

In **Chapter 4** we give a proposal of loose coupling strategies between OTT and NSP approaches aiming at improving OTT communication services.

We discuss that it is not possible to directly apply the existing network QoS mechanisms to OTT applications. We also show that some concepts can be reused, e.g. media steering through media relays and packet marking.

Therefore, we analyse the benefits of merging OTT and NSP approaches. Finally, we identify three coupling strategies that leverage network mechanisms for improving OTT communication services quality: NSP driven, OTT driven and User/Enterprise driven. For each of these strategies, we propose an example of implementation.

In **Chapter 5** we propose traffic management solutions adapted to the WebRTC solutions.

We give an overview of current Internet engineering practices and explain why the widespread best-effort delivery is not adapted to real-time traffic. Later, we perform a survey of network requirements for WebRTC and of existing queuing mechanisms.

This survey leads to a proposal of traffic management solutions adapted to the WebRTC traffic that either ensure lower queuing delays regardless the traffic or isolate the sensitive traffic.

In **Chapter 6** we detail the implementation of the Proof of Concept and the test environment.

[4]http://rethink-project.eu/

We focus on a detailed description of emulation of uplink wireline access networks with a focus on ADSL and fiber specific configurations.

In **Chapter 7** we evaluate the different traffic management solutions.

We start with an overview of evaluation methodology. Then, we provide the main results obtained for ADSL and fiber networks. We analyse the identified traffic management solutions and their impact on WebRTC, notably its congestion control mechanism. For this purpose, we not only use the network metrics, but we also leverage the application layer measurements that are enriched by subjective opinion on the perceived quality.

Based on these results, we discuss a positioning of performed evaluation in relation to the existing related studies. We also suggest several improvements to the GCC algorithm.

In the final analysis, we discuss recommendations for coupling strategies with an overview of net neutrality issues.

In **Chapter 8** we point out the main contributions of this thesis and give the research perspectives in relation to the ongoing works, notably for mobile networks and in the domain of optimal sharing of network resources.

Chapter 2

State of the art of WebRTC

Contents

2.1 Why WebRTC?

Global web services have been widely developed thanks to the availability of common platforms on devices, notably browsers. Before, each service was associated with a client application and needed to be adapted to underlying operating system or device hardware. The emergence of browsers has simplified the development of web applications and the browser has become an open platform (**O'Connell 2013**).

The browser's advantages were leveraged by *Web Real-Time Communication*[1] (WebRTC) project, i.e. a collection of standards, protocols and APIs (**Grigorik 2013**) allowing direct browser-to-browser communication. Additionally, there is ongoing work on developing WebRTC solutions for mobile platforms (**Hart 2015**) and the Internet of Things (**Sime 2016**).

[1]http://webrtc.org/

WebRTC is an emerging solution allowing audio and video communications, screen sharing and data transfer. It simplifies developers' tasks, since provided native browser tools are based on HTML5 and JavaScript. Furthermore, there is no major cost when integrating WebRTC into the existing web infrastructure, users do not need to install any plugins and interoperability between different browsers is expected to be ensured. As a result, WebRTC makes significant changes in real-time communications by lowering barriers to entry, since before there was a need of developing everything from scratch (including taking care of media engine and codec aspects) or using proprietary software development kits (**Levent-Levi 2014**).

We study WebRTC, because it is an over-the-top solution that is not proprietary and furthermore it is currently under standardization. WebRTC is discussed by three major standardization bodies:

- Internet Engineering Task Force (IETF) focusing on communication model and protocols and media functions[2].
- World Wide Web Consortium (W3C) working on APIs definition (**Bergkvist et al. 2016**).
- 3rd Generation Partnership Project (3GPP) discussing WebRTC and IMS interoperability[3].

Therefore, a description of WebRTC components is provided and easily accessible so its detailed assessment could be performed.

It is also important to point out that companies participating in standardization include: Google, Ericsson, Mozilla and Cisco.

2.2 Proprietary control plane and endpoint based media plane

In WebRTC the media plane is separated from the signaling one as it is shown on the Figure 2.1.

The signaling plane is not meant to be standardized. The communication is established thanks to exchanging signaling messages by using a web server. However, each CSP chooses implementation details on how this information should be exchanged. Additionally, each CSP has its own user base. Thus the proprietary "bubbles" are created (**Bertin et al. 2013**), and there is no interoperability between different CSPs.

Nevertheless, there are numerous standards concerning the media plane, i.e. protocols and data formats, but also application and browser APIs. When possible, WebRTC tries to reuse concepts known from VoIP (**Grigorik 2013**). The overview is given in the following sections.

2.2.1 Media plane details

There is ongoing standardization work in IETF for the media plane (**Alvestrand 2016c**).

Media traffic is time sensitive, so it requires low latency, whereas a certain packet loss can be tolerated. This packet loss tolerance is possible thanks to mechanisms explained

[2]http://tools.ietf.org/wg/rtcweb/
[3]http://www.3gpp.org/DynaReport/FeatureOrStudyItemFile-580062.htm

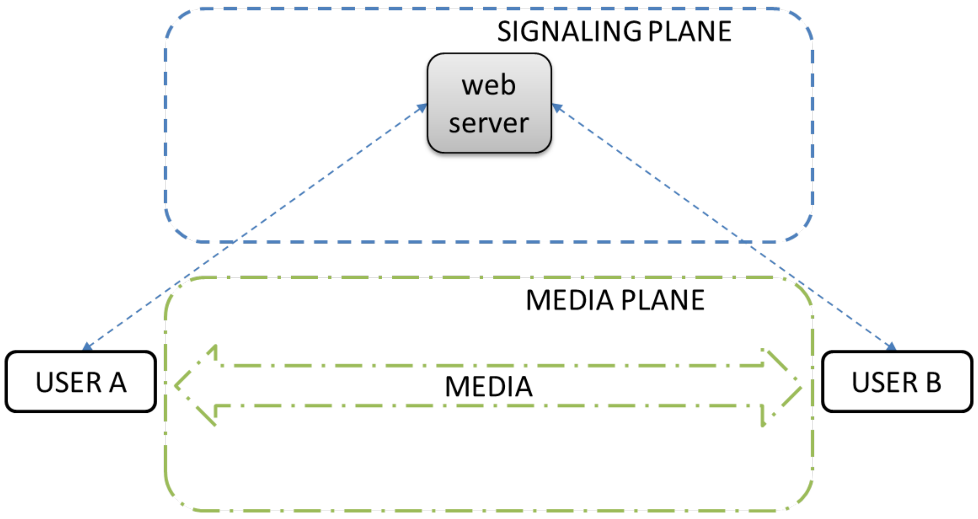


Figure 2.1 – Simplified WebRTC control and media planes

in the section 2.3.3 and the choice of codecs presented in the section 2.3.2. Therefore, the low latency plays a more important role than reliability. As a result, UDP protocol is preferred, because it provides lower delay. TCP can be also used, however it is considered to cause too much delay because of packet retransmissions (**Grigorik 2013**).

Ideally, in WebRTC communications should be done in *Peer-To-Peer* (P2P) mode, by using a direct media path between the WebRTC enabled devices. This approach is preferred as it reduces latency but also cost, since no intermediary is necessary. Nonetheless, it is not always possible, because there are NAT boxes or restricted firewalls blocking P2P traffic (**Janczukowicz et al. 2014**). Additionally, certain NSPs disable P2P in mobile networks for quality reasons (**Gavois 2014**), (**Orange nownb**), but also for charging related issues, because all traffic has to go through charging equipment that is situated behind the edge router.

Hence, there is a need to provide the connection when the direct media path is not possible. For this reason Interactive Connectivity Establishment (ICE) is used (**Rosenberg 2010**), since it allows choosing between different connection modes.

ICE - Interactive Connectivity Establishment

The ICE protocol (**Rosenberg 2010**), enables discovering possible media addresses that can be used for communication. It gathers possible IP addresses and ports in order to test their connectivity.

Typically there are two endpoints trying to establish a communication. They are able to exchange signaling messages indirectly, by using a Web server. Each endpoint discovers all his possible addresses, i.e. ICE candidates that can be used for communication, that include:

- Host address – address allowing a direct media connection.
- Reflexive address – a public address allocated to a device behind a NAT learnt from a STUN server placed in a public network (**Rosenberg et al. 2008**).
- Relayed address – an address provided by a TURN server that acts as a media relay and is placed in the media path (**Mahy et al. 2010**).

These possible ICE candidates are shown on the Figure 2.2.

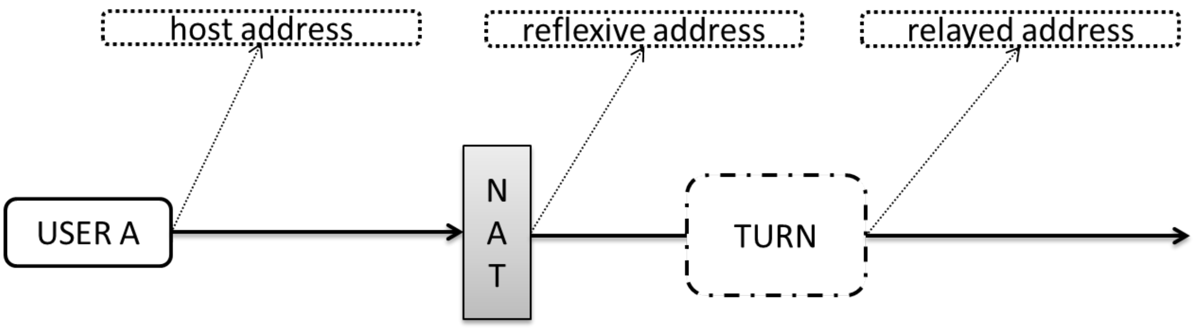


Figure 2.2 – Possible ICE Candidates

STUN - Session Traversal Utilities for NAT

STUN (**Rosenberg et al. 2008**) is a protocol, used as a tool by other protocols in case of NAT traversal. It allows discovering what public IP address and port is allocated by a NAT. It also enables keeping NAT bindings alive.

STUN messages consist of a fixed header containing a method, a class and a transaction ID. The header is followed by attributes indicating additional information for a given message.

When an endpoint (STUN client) wants to learn its public address, it sends a STUN message called Binding Request to a given STUN server. If there is a NAT in the path (or several NATs) it will modify the source transport address of the request. Therefore, the STUN server receives the Binding Request with a source transport address that corresponds to endpoints public address, i.e. its reflexive address.

The STUN server puts this reflexive address in the attribute XOR-MAPPED-ADDRESS within the body of the STUN message called Binding Response and sends it back to the endpoint. Thus, the endpoint can learn its address.

STUN call flow is presented on the figure 2.3.

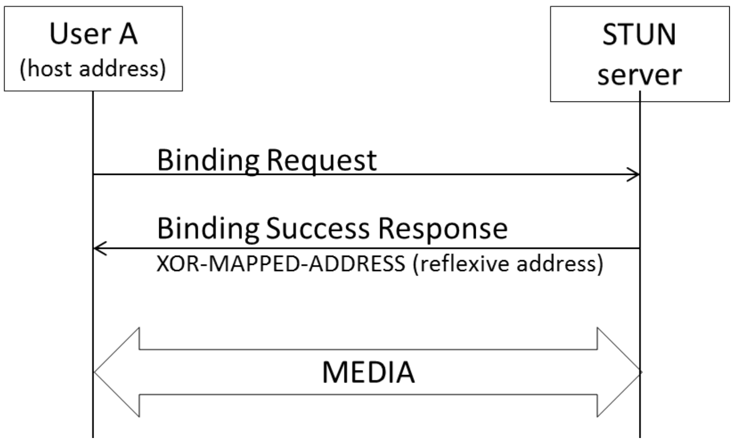


Figure 2.3 – STUN Call Flow

TURN - Traversal Using Relays around NAT

As mentioned before, in certain situations it is impossible for hosts to communicate directly. The TURN protocol (**Mahy et al. 2010**) allows using a media relay placed in the media path and ensures the communication. It is important to highlight that TURN is a STUN extension, thus all TURN messages, except ChannelData, are formatted like STUN messages.

The simplified TURN call flow is presented on the figure 2.4.

If User A, i.e. the TURN client, cannot establish a P2P connection, but wants to exchange media with User B, i.e. the remote peer, by passing through a TURN server, first it needs to create an allocation on the server.

The TURN client sends an Allocate Request message to the TURN server that responds with an Allocate Response message, that contains the allocated relayed transport address in the attribute XOR-RELAYED-ADDRESS. In existing implementations, e.g. coturn[4], if a TURN server has several possible addresses, it selects one by using a simple round robin for load balancing purposes. In this step, the TURN server may require the client to authenticate depending on its configuration. When the relayed address is allocated, the TURN client sends Refresh message periodically in order to keep the allocation alive.

Later the remote peer's address is indicated in the attribute XOR-PEER-ADDRESS. The TURN client knows the remote's peer address since they have exchanged their ICE Candidates before.

Then, to send data, the TURN server and client use ChannelData packet format (different than STUN format), containing a header with a channel number. To establish a channel, the TURN client sends a ChannelBind request. Later, the channel is bounded and there is a channel number associated with the peer. The client's and peer's data are sent with ChannelData messages between the client and the TURN server.

The peer does not interact with the TURN server in any particular way, so between the TURN server and the peer a UDP can be used.

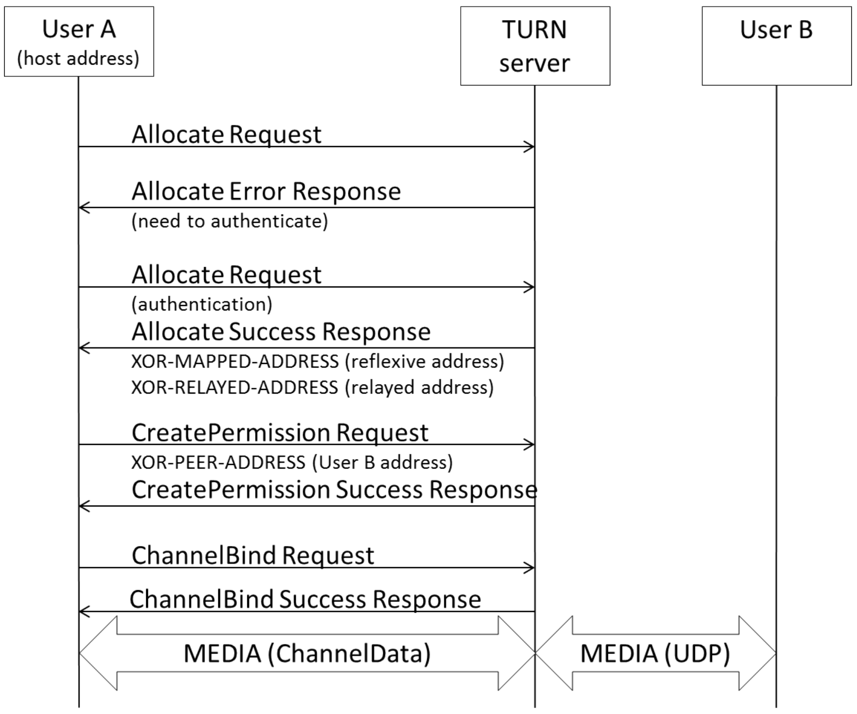


Figure 2.4 – TURN Call Flow

It is important to note that TURN is a "heavy" resource, since it needs to have good performance and high-bandwidth Internet connection. As a result, it may cause high

[4]https://github.com/coturn/coturn

costs to its provider.

Connectivity Checks

After each of the endpoints collects all ICE candidates (a combination of IP address and port), the connections between possible ICE candidate couples need to be verified.

First, each endpoint sorts collected candidates from the highest to lowest priority and sends them to the remote endpoint using an SDP offer. When the remote endpoint receives the offer, it answers with its own list of candidates. Each endpoint ends up having a complete list of candidates from both sides and one of the endpoints is selected as a controlling agent.

Then, the ICE candidates are paired up by each endpoint so that different combinations can be tested. Each endpoint runs a series of Connectivity Checks, starting from higher priority candidates. A connectivity check is a STUN Binding Request/Response sent from a local endpoint to a remote one. Connectivity checks are sent on exactly the same IP addresses and ports that will be used for the media.

Finally, the controlling agent selects a candidate pair to be used among all pairs that had valid connectivity checks. The standard recommends privileging host candidates and using relayed addresses as a last resort (**Rosenberg 2010**).

The call setup takes time when the traditional ICE mechanism is used, because the connectivity checks cannot start before all candidates are collected. *Trickle ICE*, (**Alvestrand 2016a**), is an extension to ICE that addresses this issue. With Trickle ICE, ICE can begin connectivity checks while still gathering the candidates. As a result, it considerably shortens the time of connection establishment.

There are three major connection modes possible: (1) direct using host addresses, (2) direct using reflexive addresses learnt from STUN, (3) through TURN server using relayed address provided by TURN. They are presented on the Figure 2.5.

SRTP and SRTCP

WebRTC uses *Real-time Transport Protocol*, notably the *Secure RTP* (SRTP) (**Perkins et al. 2016**), i.e. protocols already used in different real-time solutions (**Grigorik 2013**).

RTP (**Schulzrinne et al. 2003**) provides functionalities to ensure end-to-end network transport of real-time data, notably audio and video. RTP does not provide resource reservation or quality of service, but it provides a control protocol, i.e. *RTP Control Protocol* (RTCP) providing feedback about the quality. RTP enables payload type identification, sequence numbering, timestamps and monitoring of delivery.

SRTP (**Baugher et al. 2004**) is a secure profile of RTP. It is an extension that enables encryption and message authentication of RTP and RTCP. Just like RTP it does not provide any guarantees or reliability. However, SRTCP is used here and among other pieces of information, it provides the number of lost packets and last received sequence number.

When using RTP in WebRTC context, there are certain requirements to fulfill (**Perkins et al. 2016**). In WebRTC, encryption is mandatory, which justifies the use of SRTP. Also, WebRTC requires RTP and RTCP multiplexing onto a single flow. This simplifies the traversal of NATs and firewalls, since it requires only one binding. Additionally, there is a need of adapting media to the network variations and there are ongoing efforts on providing congestion control algorithms (**Perkins et al. 2016**), (**Grigorik 2013**).

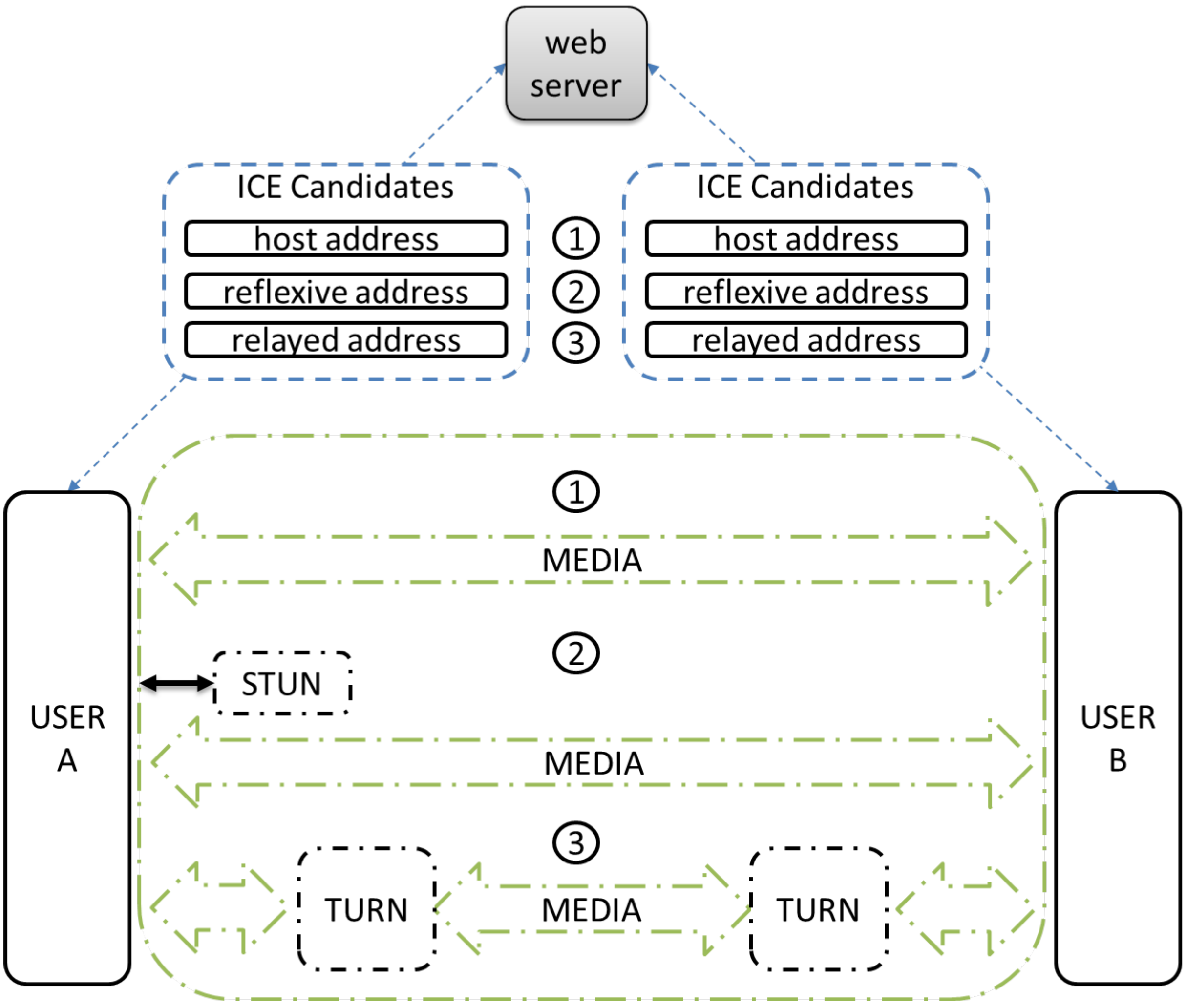


Figure 2.5 – Connection modes: 1.Direct with host addresses 2.Direct with reflexive address 3.Through media relay with relayed address

The advantage of using WebRTC is that all necessary media infrastructure is already implemented in a browser (**Grigorik 2013**). The developer's task is thus simplified.

Data Channel

Even though in this thesis we focus on audio and video communication, it is important to highlight that WebRTC enables also a transfer of non-media data. For this reason, Data Channel was implemented (**Jesup et al. 2015**).

Hence, a non-media data is sent by using *Stream Control Transmission Protocol* (SCTP) (**Stewart 2007**), encapsulated in *Datagram Transport Layer Security* (DTLS) (**Rescorla & Modadugu 2012**).

The non-media data transport is done parallel to the SRTP media and they can even share the same UDP port (**Jesup et al. 2015**).

Protocol stack

Given the above information it is possible to establish a simplified protocol stack, that is presented on the figure 2.6.

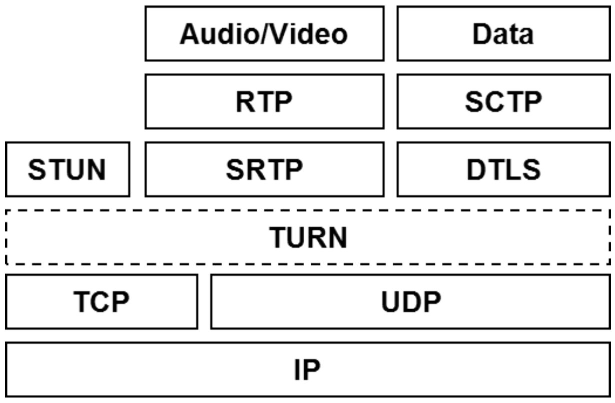


Figure 2.6 – Protocol stack

2.2.2 Application Programming Interfaces

W3C documentation defines the *Application Programming Interfaces* (APIs) to support the features specified by the IETF (**Bergkvist et al. 2016**).

There are three major APIs:

- getUserMedia,
- RTCPeerConnection,
- DataChannel.

getUserMedia

W3C defines a JavaScript API that enables to request local media, i.e. audio and video, from the platform. It also provides a definition of MediaStream API that allows controlling the multimedia stream (**Burnett et al. 2016**).

Generally speaking getUserMedia allows requesting access to a microphone and a camera. It also allows specifying mandatory and optional constraints, e.g. video resolution or frame rate, and it enables to request media streams that meet the given constraints.

RTCPeerConnection

RTCPeerConnection, as its name indicates, allows establishing P2P communications.

RTCPeerConnection supports the ICE functionality, like discovering new ICE candidates or creating SDP offers and answers, but also allows managing local and remote streams. It enables specifying configuration details, e.g. STUN and TURN servers, to establish a connection. It also enables specifying a preferred connection mode, e.g. it can force using a media relay. Furthermore, it features monitoring ICE connection state (**Bergkvist et al. 2016**), (**Grigorik 2013**).

RTCDataChannel

RTCDataChannel is a bi-directional data channel set up between two peers. It can be established once the PeerConnection is set up. Its behaviour is similar to WebSocket but has additional benefits like P2P aspects and more flexibility concerning the underlying transport, e.g. delivery and reliability (**Bergkvist et al. 2016**), (**Grigorik 2013**).

2.3 Focus on application layer mechanisms for quality improvements

As mentioned in Chapter 1, for OTT applications, application and network layers are independent and therefore Webcos use best-effort Internet delivery. Hence, they do not get any specific network assistance.

As a result, they rely on application layer mechanisms, i.e. mechanisms built into the endpoints, e.g. congestion control or adaptive codecs, and adapt to underlying network fluctuations.

In this section we will present the efforts of WebRTC community in the study of these mechanisms.

2.3.1 TRAM and TSVWG

TRAM

TURN Revised and Modernized (TRAM[5]) is an IETF working group created due to increased interest in ICE, STUN and TURN. It focuses on updating and improving STUN and TURN in order to adapt them to ongoing technology changes. Among others, this working group consists of various efforts concerning discovering underlying network characteristics with endpoint based messages, thus without any network assistance.

First, there is a draft (**Martinsen et al. 2016**) focusing on measuring *Round-Trip Time* (RTT) and fractional loss using STUN.

At the present time, if there are several possible addresses, ICE selects one address based on a static configuration. When taking this decision it does not consider any path characteristics. Therefore, a chosen interface may not be the most optimal for a given communication.

The draft (**Martinsen et al. 2016**) discusses a mechanism allowing an endpoint to use STUN messages in order to discover path characteristics. It introduces a new STUN attribute: TRANSACTION-TRANSMIT-COUNTER. With this attribute, STUN client can associate a STUN request with a corresponding STUN response and calculate the RTT. Also, thanks to this attribute, it is possible to avoid any confusion introduced by eventual retransmission.

Additionally, the mechanism should provide packet loss information. However, for now, it is not always possible to distinguish packet loss from packet reordering, or even to identify the direction in which the packet loss was observed. Hence, many measurements over a long time period should be done in order to try to detect a pattern.

Second, there is a draft (**Martinsen et al. 2015**) focusing on pre-call probing of a TURN server, in order to discover the maximum bandwidth along with maximum latency and bufferbloat of the aggregation of uplink and downlink.

Thus, after getting allocation and appropriate permission on a given TURN server, an endpoint can send data (TCP or UDP) to itself through this TURN server and perform the measurements.

[5]https://datatracker.ietf.org/wg/tram/documents/

TSVWG

Transport Area Working Group (TSVWG[6]) discusses transport topics that are not in the scope of existing groups but that do not necessarily need creating a specific working group. Within this group, a document concerning TURN extension to convey flow characteristics has been published (**Wing et al. 2014**).

The draft focuses on issues caused by overloading TURN server and the network in which it is hosted. Currently, a TURN server does not have any guidelines on how to act in case of too many flows, so it does not know which flows are more critical. Hence, the document suggests a mechanism allowing TURN client to provide flow characteristics to the TURN server.

More precisely, a TURN client can send flow characteristics, i.e. delay, loss and jitter tolerance, to the TURN server in order to demand a differentiated service. This mechanism is intended notably for long lived flows such as media streams and WebRTC data. When TURN server receives the request, it indicates to the client if it can accept its demand. It can also suggest another TURN server.

The document indicates that a TURN server can relay flow characteristics to other network elements but do not give any details. Additionally, there is no information about how TURN clients can exploit this information.

2.3.2 Audio and Video Codecs

Codecs play an important role in WebRTC communications, since even though hardware nowadays can capture high quality streams, CPU and bandwidth are not necessarily able to keep up (**Grigorik 2013**). The choice of media codecs has an impact of upper and lower limits of supported bitrates, but also on robustness in case of packet loss (**Perkins et al. 2016**).

The list of codecs that WebRTC endpoints are required to support is given in the draft (**Valin & Bran 2016**) for audio and (**Roach 2016**) for video. In current implementations, the Opus for audio and the VP8 for video are preferred (**Grigorik 2013**).

Opus

Opus (**Valin et al. 2012**), (**Opus nown**) is an interactive speech and audio codec, that can be used for wide range of applications, e.g. videoconferencing, Voice over IP, in-game chat or distributed music performances.

The codec can scale from low bitrate narrowband speech to high quality stereo music, making the bandwidth vary from 6 to 510 kbit/s. Opus supports constant and variable bitrate encoding and it also dynamically adjusts bitrate and audio bandwidth. In the Google Chrome browser, mono bitrate for Chrome is 32kbit/s and stereo bitrate is 64kbit/s (**Khan nown**).

Additionally, it implements *Forward Error Correction* (FEC) which offers more robust behaviour in case of packet loss.

Consequently, Opus can easily adapt to different content and network conditions.

[6]https://datatracker.ietf.org/wg/tsvwg/documents/

VP8

VP8 (Bankoski et al. 011a) is a video codec with variable and constant bitrate encoding options.

When developing VP8, its creators had in mind Internet and web-based applications. VP8 is supposed to work even in cases of low bandwidth but also on heterogeneous hardware. It implements advanced coding features so it can improve compression efficiency and decoding speed (Bankoski et al. 011b).

It is important to point out that the codec bitrate changes depending on the quality of the streams. For WebRTC the bitrate values are (Grigorik 2013):

- 1 - 2 Mb/s for 720p and 30 *frames per second* (FPS),
- 0.5 - 1 Mb/s for 360p and 30 FPS,
- 0.1 - 0.5 Mb/s for 180p and 30 FPS.

2.3.3 Handling packet loss for VP8

To handle packet loss for video flows and VP8 codec, WebRTC uses an adaptive hybrid approach combining *Forward Error Correction* (FEC) and packet retransmission based on *Negative Acknowledgment* (NACK). The FEC/NACK approach allows to balance redundancy cost (FEC) and delay cost (NACK) (Holmer et al. 2013).

FEC settings are based on network statistics, notably RTT measurements. If RTT is low, packets can be retransmitted faster, so FEC level of redundancy can be lower. In case of higher values of RTT, retransmission can cause too much delay, so there should be more redundancy. Additionally, in WebRTC there is an adaptive playback delay, meaning that while waiting for packets retransmission, the playback delay is modified in order to reduce the duration of frozen video (Holmer et al. 2013).

2.3.4 Congestion Control Algorithms

Real-time traffic, notably WebRTC traffic, has to share network links with other types of traffic that do not necessarily have the same type of behaviour. Additionally, there are heterogeneous network environments, with varying delays and available bitrates. Since for OTT communication best-effort traffic is used, network behaviour cannot be predicted. The quality of communication depends on many aspects, so it should adapt to the underlying networks.

WebRTC should use the available resources in the most optimal way. As a result, a congestion control algorithm is essential (Perkins et al. 2016). For this reason, the IETF *RTP Media Congestion Avoidance Techniques* (RMCAT[7]) working group was created.

In essence, this working group highlights the need of congestion control for interactive traffic. Furthermore, they define congestion control requirements for RTP flows and study interactions between different flows. With this in mind, they study, develop and evaluate the candidate congestion control mechanisms.

The working group has defined requirements for evaluation of congestion control for interactive real-time media. We quote the essential ones (Jesup & Sarker 2014):

- Provide delays as low as possible.

[7]https://datatracker.ietf.org/wg/rmcat/documents/

- Be resilient to various behaviour of underlying network, e.g. routing changes or interface changes, and adapt quickly to changing network conditions.
- Be fair to other flows, avoid building up queues when competing with other traffic and avoid starving it.
- Not require any network support and leverage mostly information like packet arrival time, packet loss, acknowledgements, etc.

Furthermore, the RMCAT working group highlights the lack of generally accepted congestion control algorithms that would be adapted to interactive real-time traffic.

In fact, in RMCAT group, at this time three congestion control algorithms for real-time media are being discussed:

- *Google Congestion Control Algorithm for Real-Time Communication* (GCC) (**Holmer et al. 2015**).
- *Unified Congestion Control Scheme for Real-Time Media* (NADA) (**Zhu et al. 2016**).
- *Self-Clocked Rate Adaptation for Multimedia* (SCReAM) (**Johansson & Sarker 2016**).

So far none of the congestion control algorithms has been chosen as a standard.

In this thesis we focus mainly on GCC congestion control algorithm as it is developed and used in Google Chrome since M23 and in Google Hangouts. Thus, we are able to evaluate its behaviour in our implementations.

Google Congestion Control Algorithm for Real-Time Communication

Google Congestion Control Algorithm for Real-time communication (GCC) draft (**Holmer et al. 2015**) describes two methods of congestion control for video flows, i.e. delay-based and loss-based that, used together, improve real-time communication performance.

The draft specifies that there are two ways to implement this algorithm: (1) with both controllers running at the sender side, or (2) with the delay controller running on the receiver side and the loss controller running on the sender side. However, in articles analysing GCC, e.g. (**Carlucci et al. 2016a**), the second approach is highlighted, thus we are going to focus on it. Nevertheless, regardless the placement of the delay-based controller, the principal behaviour is the same.

In brief, the delay-based controller on the receiver side calculates a sending rate. It provides it to the sender. On the sender side, the loss-based controller calculates a complementary sending rate. Later, both sending rates are compated and the sender chooses the lowest.

The simplified schema of GCC is given in figure 2.7 and will be explained below based on the existing literature (**Holmer et al. 2015**), (**Carlucci et al. 2016a**), (**Carlucci et al. 2014**).

The delay-based controller (Figure 2.8), implemented at the receiver side, calculates the maximum bitrate (A_r) based on packet arrival time. It consists of three main elements.

- Arrival-time filter

  The arrival-time filter uses timing of arriving packets to update estimates of network parameters. Particularly, it provides an estimate of the one way delay gradient ($m(t_i)$), where t_i is the time a i-th video frame is received.

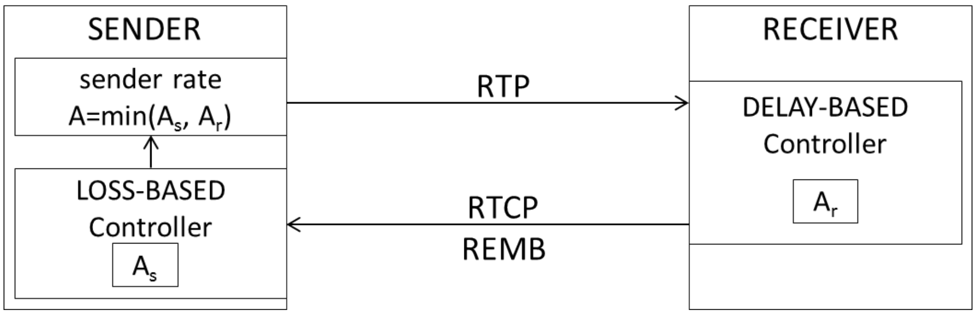


Figure 2.7 – GCC simplified schema

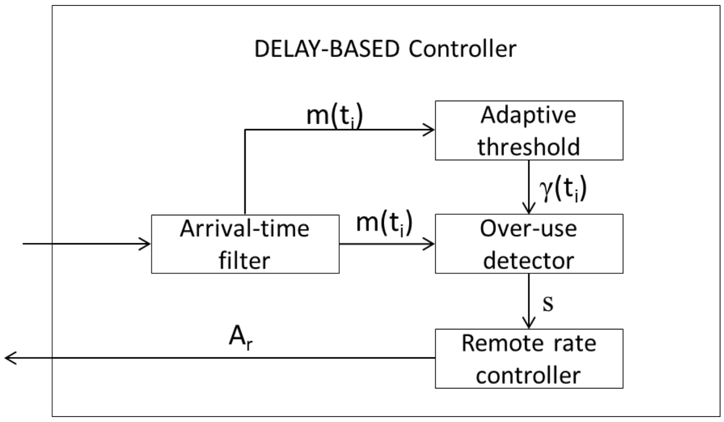


Figure 2.8 – Delay-based controller

The estimate is obtained using the measured one way delay variation ($d_m(t_i)$), calculated based on a time at which neighbouring video frames were sent and at which they were received, i.e.:

$d_m(t_i) = (t_i - t_{i-1}) - (T_i - T_{i-1})$

where:

t_i - the time of receiving the last packet of the i-th video frame,

T_i - the time of sending the first packet of the i-th video frame.

Since the measured values are affected by a jitter noise, a Kalman filter is used to filter out the one way delay gradient ($m(t_i)$). The Kalman filter is a set of mathematical equations, that based on series of measurements collected over a certain time, can produce a noise-free estimate (**Welch & Bishop 2006**).

This $m(t_i)$ estimate is helpful in determining the bottleneck link utilization: when the buffers build-up, $m(t_i)$ becomes positive, when the buffers are draining, $m(t_i)$ becomes negative and when the buffers are emptied, $m(t_i)$ is close to 0.

- Over-use detector

  The over-use detector compares the $m(t_i)$, obtained from the arrival-time filter, with a dynamically changing threshold $\gamma(t_i)$. The comparison is done every time a video frame is received. Later, the over-use detector generates one of the following signals, s: *underuse*, *normal*, *overuse*.

  The detection of link utilization is presented on the figure 2.9. If the estimate is above the threshold, an *overuse* is detected. However, it will be signalled to the rate

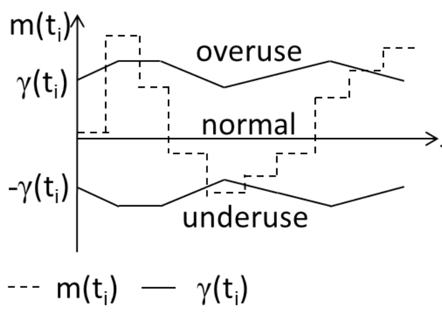


Figure 2.9 – Over-use detector - link utilization detection

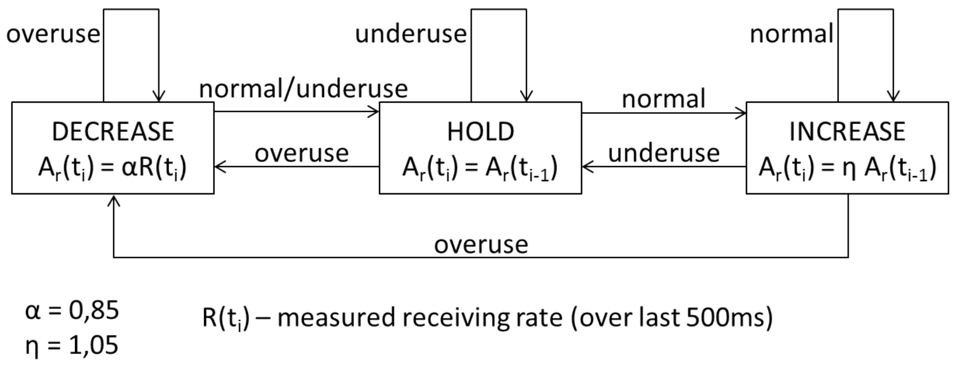
controller only if this state lasts for a given time period (i.e. 100ms based on the last publication on this subject (**Carlucci et al. 2016a**)). Similarly, if the estimate is below the negative value of the threshold an *underuse* is detected. Otherwise, the *normal* state is detected.

The γ threshold has an influence on the tolerated queuing delay and the algorithm's reactivity. Thus, with a larger threshold, a larger queueing delay is acceptable, whereas a smaller threshold allows better reactivity. This allows adapting to the eventual presence of other flows, e.g. increasing the threshold prevents flow starvation in the presence of TCP concurrent traffic. This threshold changes dynamically and it adapts its value by tracking the queuing delay variation (**Carlucci et al. 2014**).

- Remote rate controller

  The remote rate controller uses the signal s generated by the over-use detector. Based on this signal it provides the A_r value using the finite state machine shown on the figure 2.10.

  When the link is overused, it decreases the sending rate (decrease state). Hence, the buffers start draining, the underuse signal is generated and the sending rate remains stable (hold state). Later when the buffer is emptied, the normal signal is triggered and the remote controller increases its sending rate (increase state).



overuse
underuse
normal
normal/underuse
normal
DECREASE $A_r(t_i) = \alpha R(t_i)$
overuse
HOLD $A_r(t_i) = A_r(t_{i-1})$
underuse
INCREASE $A_r(t_i) = \eta A_r(t_{i-1})$
overuse
$\alpha = 0,85$
$\eta = 1,05$
$R(t_i)$ – measured receiving rate (over last 500ms)

Figure 2.10 – Remote controller states

The computed rate A_r is provided to the sender with RTCP REMB messages, i.e. RTCP message for Receiver Estimated Maximum Bitrate (**Alvestrand 2016b**).

This feedback is sent at least every 1s or when a congestion is detected, i.e. when the A_r values are decreased by over 3%.

Given the above description, we can conclude that the delay-based controller probes for available bandwidth by adapting its sending rate and aims at keeping the queuing delay small by assuring optimal link utilization.

The loss-based controller is activated every time a RTCP or RTCP REMB message is received. The recommended feedback interval is 30ms or once every received video frame. RTCP messages provide information about lost packets. The loss-based controller computes the loss-based sending rate A_s as follows:

- It increases A_s if packet loss during the last reporting interval is negligible (less than 2%),
- It does not change A_s if packet loss during the last reporting interval is low (between 2 and 10%),
- It decreases A_s if packet loss during the last reporting interval is high (more than 10%).

Thus, the input of loss-based controller is complementary to the delay based controller in the presence of packet loss.

The delay-based A_r is compared with the loss-based A_s. The minimum value is used as the sending rate.

2.3.5 Circuit Breakers

Additionally, the draft on RTP for WebRTC (**Perkins et al. 2016**) indicates that WebRTC solutions should implement RTP circuit breakers.

The draft (**Perkins & Singh 2016**) defines a set of RTP circuit breakers. The document gives an overview of network conditions to which an RTP sender should react by stopping the transmission of media data, for instance extreme congestion, media time out or RTCP timeout.

An RTP sender can cease to transmit, meaning that it can stop a single RTP flow or multiple bundled RTP flows. However, the flows should not be restarted automatically, unless the RTP sender is able to obtain the information about a given network issue being resolved.

The circuit breaker is used to enable applications to react to situations caused by extreme network congestion, but it does not replace a congestion control algorithm, since it is triggered only in case of certain network conditions.

2.4 Summary

This chapter focused on technological aspects of WebRTC and it included an overview of standardization efforts, notably concerning application layer mechanisms.

WebRTC is a rich collection of components enabling real-time communications. It not only allows to easily acquire media streams, but also takes care of media management.

The biggest advantage of WebRTC is its simplicity. The large part of functionalities is integrated in browsers. Additionally, P2P communications are privileged, so there is no need of implementing any particular infrastructure, except for usage of STUN and TURN servers.

WebRTC is under standardization, so all technological details are accessible and there is a community constantly working on improving it.

WebRTC is partially based on standards previously used in e.g. VoIP (ICE, TURN, STUN), but proposes a novel approach, by leveraging browser advantages.

It focuses on putting intelligence in the endpoints. The adaptive mechanisms used by WebRTC consist of probing networks and estimating network parameters. For instance, there is ongoing work on congestion control mechanisms. Thus, WebRTC does not count on any network assistance and adapts to the underlying best-effort network fluctuations.

Chapter 3

State of the art and motivations for QoS for communication services

Contents

3.1 Limits of best-effort delivery

The Internet aims to provide robust connectivity and is based on best-effort delivery. This best-effort delivery approach consists in processing packets as quickly as they arrive, but without any guarantees (**Statovci-Halimi & Franzl 2013**). However, this approach has some limits that will be detailed in this section.

3.1.1 Limits of network capacity

The Internet supports different types of applications that share finite network resources. This causes congestion creation that can significantly perturb packets delivery. This issues were addressed in various studies, notably (**Bauer et al. 2009**) and (**Campedel 2007**).

In the existing literature, e.g. (**Leavitt 2010**) and (**Kreibich et al. 2010**), the edges of the network, i.e. user access networks and peering points, are indicated as areas with limited network resources. Hence, we are going to analyse these aspects in this section.

Access networks

Internet users experience is influenced by the type of access network provided by their NSPs. Regarding wireline access networks, even with the ongoing fiber implementation, the majority of users still use older technologies that offer more limited bandwidth. For instance, according to ARCEP, (**Arcep 2016**), in France in second quarter of 2016, out of 27.220 million broadband subscribers, there were 1.765 million end-to-end fiber subscriptions. It represents 27% of households eligible to *Fiber to the Home* (FttH). At the same time, there were 22.450 million ADSL subscriptions. Furthermore, in Orange France, based on the internal documentation, 80% of ADSL subscriptions correspond to users that obtain 1Mbit/s or less for uplink.

Furthermore, Bauer in his study on the evolution of Internet congestion (**Bauer et al. 2009**) indicates that after a transition to broadband Internet and the emergence of more powerful multimedia capable devices, users started using more bandwidth hungry applications, e.g. streaming, P2P, file sharing, interactive gaming. Thus, the aggregated traffic has grown in wireline access networks, since the number of devices on a single link increased. For example in France in 2014, one third of households is equipped with at least 4 devices, i.e. TV, PC, smartphone and tablet (**Offremedia.com 2015**).

Another key point is that mobile radio resources are also limited. They are shared by a number of users in a cell and there has been a significant growth of mobile users. This growth, together with flat rate pricing and omnipresence of smartphones, has caused the increase of mobile traffic volume. Cisco predicts that between 2015 and 2020, global mobile data traffic will increase eightfold (**Cisco 2016**). However, overprovisioning in cellular networks would be uneconomical (**Ekstrom 2009**).

Peering points

Congestion can be created at peering points between different NSPs, not necessarily because of technical, but because of economic aspects.

Peering agreements between different NSPs have an important impact on routing, meaning that the final path does not always have the shortest latency (**Briscoe et al. 2014**). Moreover, at interconnections, congestion is caused by conflicts between different actors, especially when traffic at the peering points is too asymmetric (**Clark et al. 2014**).

Usually different providers have free-peering agreements, assuming that they exchange the same amount of data. However, it happens that peering agreements are abused, like for example in the case of Netflix sending its traffic via Cogent to Verizon, when Verizon refused to upgrade its infrastructure and claimed that it had to accept too much traffic from Cogent (**Brodkin 2014**). This limit cannot be easily overcome and mostly demands some business negotiations.

3.1.2 Limits of overprovisioning

Current solutions to avoid congestion problems are mostly based on overprovisioning. This approach assumes that quality of networks depends strongly on the allocated amount of resources. As a result, allocating enough network resources helps to prevent creating congestion. Additionally, network architecture is kept simple, since there is no differentiation between flows and no particular flow treatment (**Gozdecki et al. 2003**).

Furthermore, this approach allows using flat pricing for Internet data plans, i.e. using a fixed fee regardless used resources or applications. When benefiting from flat pricing, users do not limit themselves when consuming network resources (**Meddeb 2010**). There is often no additional cost when using OTT communication services that rely on adaptive application layer mechanisms to maintain a certain level of services in case of network problems (**Teitelbaum & Shalunov 2002**). In fact, OTT take advantage of the flat rate data plans, that put no restriction on used data volume and so they can offer competitive communication services (**Statovci-Halimi & Franzl 2013**).

However, overprovisioning has some disadvantages. With the increase of bandwidth, usage grows and it is not always possible to predict required capacity, notably for rich-media services. These aspects were discussed in the follwoing studies: (**Gozdecki et al. 2003**), (**Statovci-Halimi & Franzl 2013**), (**Bauer et al. 2009**). Furthermore, the speed of growing network capacity demands is not necessarily the same as the speed of increasing network resources.

Even though overprovisioning is believed to be less expensive than investing into QoS or premium services (**Teitelbaum & Shalunov 2002**), it still demands an effort from NSPs, that have to constantly invest in their network infrastructures. NSPs investments and marketing strategies have an impact on available network capacity. Also infrastructure investments should be planned in advance because these investments take time (**Bauer et al. 2009**). However, as we have shown in the Chapter 1 and as it is discussed in the studies (**Gozdecki et al. 2003**), (**Statovci-Halimi & Franzl 2013**), NSPs do not have a direct revenue from investing into increasing network resources, so they may decide that it is not profitable.

3.1.3 Unpredictability of OTT traffic

Another important point is that NSPs lack full visibility over OTT traffic. Furthermore, OTT traffic changes over time and is unpredictable. Together with varying Internet capacity, it can be a cause of congestions. To illustrate this, (**Bauer et al. 2009**) discusses that at first, traffic peaks were created during business hours, so easier to predict, whereas now, provisioning decisions are based on residential customers patterns that are less predictable.

As mentioned above, the number of devices in home networks has increased. These devices connect with different technologies and some of them are completely independent from NSPs. As a result, it has become complex for NSPs to monitor the traffic and to allow performance diagnostic of home networks (**Aouini et al. 2015**).

The increase of content demand, notably video streaming, has been a reason of implementing *Content Delivery Networks* (CDN). CDNs are for instance used by YouTube or Netflix. CDNs deploy distributed server infrastructures in order to place their content in locations all over the Internet. The services using CDNs are successful and as a result, the CDN traffic represents a large part of Internet traffic. However the overlay network that they create has an impact on the Internet best-effort delivery. It can cause important traffic shifts within a short period of time (**Poese et al. 2012**), since CDN providers use the application layer routing management, so the routing decisions are based on their own criteria (**Bécot et al. 2015**).

P2P services also create a traffic that is difficult to predict for NPSs. This traffic is problematic because its paths can change rapidly, even on hourly basis, and create congestion on unexpected links (**Lundqvist 2011**).

3.1.4 Different nature of real-time traffic

Last but not least, it is important to highlight that there is a growing number of new challenging applications, notably time sensitive applications. Multimedia applications, i.e. real-time communication or on demand video, are growing fast and demanding more network resources and better transport quality (**Statovci-Halimi & Franzl 2013**). Nevertheless, the widely used best-effort delivery is more beneficial for typical non real-time applications, e.g. TCP traffic that is less sensitive to delays (**Meddeb 2010**).

Indeed, the Internet has been concerned more about throughput and it has been common to use large buffers aiming to improve the network utilization and minimize the loss. Buffers are needed in packet networks. However, when they are too large, unmanaged and frequently full, they become a cause of important delays, as discussed in (**Gettys & Nichols 2011**) and (**Kuhn et al. 2014**).

Based on (**Grigorescu et al. 2013**), the queuing algorithm mostly used in routers is *First In First Out* (FIFO). It uses a droptail, so packets are queued in a buffer until they are transmitted. However when this buffers becomes full, packets are dropped. Loss-based TCP congestion control mechanisms probe for capacity and increase the rate until they detect a packet loss. Hence they fill the buffers, causing high queuing delay and in the end they penalize the real-time-traffic .

This negative side of buffers is known as the bufferbloat. Overbuffering is especially a problem of end user access devices, notably DSL (**Kreibich et al. 2010**).

Indeed, there are studies indicating that latency, not bandwidth, leads to performance bottleneck (**Grigorik 2013**). This is especially true for real-time services that have different nature and requirements than non real-time TCP flows.

Real-time applications value predictable and consistent delivery, so they do not behave well in the presence of network fluctuations (**Lundqvist 2011**). Videoconferencing is especially sensitive to delay, which has a major impact on convenience of communication. Hence for real-time applications more bandwidth does not necessarily provide better performance (**Statovci-Halimi & Franzl 2013**).

As a result, the different nature of real-time traffic is the motivation for investing in service differentiation (**Statovci-Halimi & Franzl 2013**). Hence, we create and assessment of the characteristics and requirements of the OTT real-time traffic in the chapter 5.

3.2 QoS solutions for communication services

Quality of Service (QoS) aims to meet application or end user requirements by providing the ability to manage network performance (**Meddeb 2010**). QoS has different notions, it can be evaluated based on measurements of objective parameters, but it can also reflect customers experience (**Gozdecki et al. 2003**).

There is large spectrum of aspects linked with QoS. In this thesis we are focusing on QoS approaches for communication services

In this section we will present several QoS approaches, notably managed VoIP solutions and solutions that require coupling of application and network layers.

3.2.1 Managed VoIP

In general, users, when using the Internet, expect it to provide connectivity, whether it is best-effort or not. For them QoS is difficult to understand. Furthermore, subjective perception of quality may change depending on users, application, etc. As a result, QoS is difficult to monetize in a *Business to Client* (B2C) model (**Statovci-Halimi & Franzl 2013**).

NSPs sell QoS implicitly, so as a part of certain services, notably VoIP (**Meddeb 2010**). These services become more attractive for customers and can be used to differentiate from other NSPs. Thus, the revenue is not directly linked with QoS but it is an indirect result of e.g. increased number of subscribers.

In this section, we present an overview on how NSPs provide real-time communication services for their subscribers. The description is limited to mass market VoIP solutions. We focus on a common practice among the network providers as the standardization does not describe the full implementation of these solutions (**Janczukowicz et al. 2014**).

Telco CSPs based their services on traditional telephony. Thus, each NSP provides a service to its own subscribers and later interconnects with other NSPs. The simplified schema of managed VoIP for different network segments is presented on figure 3.1.

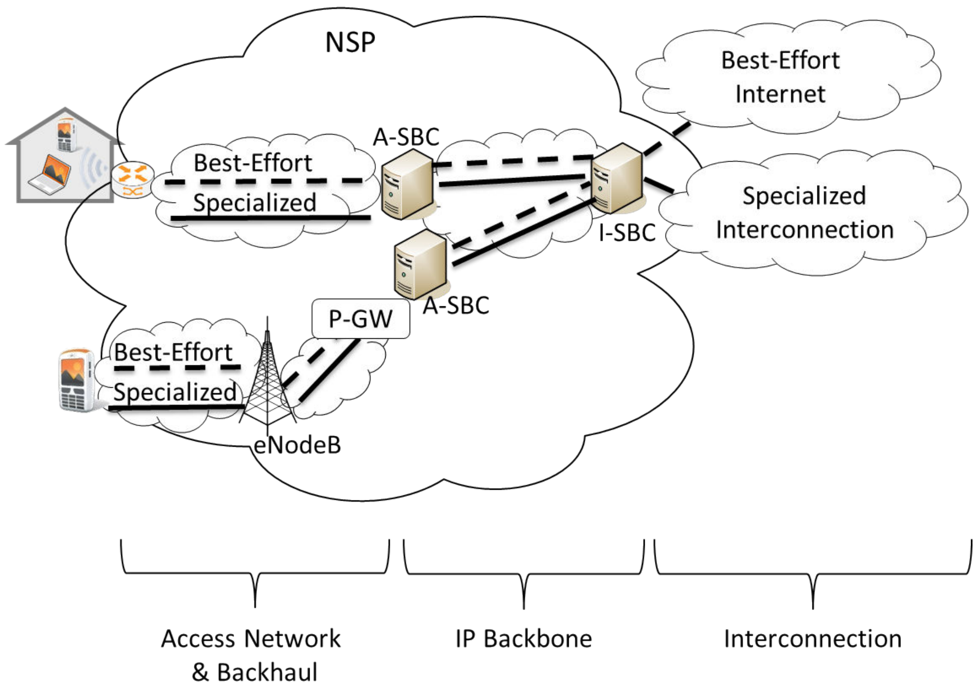


Figure 3.1 – Managed VoIP simplified schema

Access networks

Managed VoIP flows are separated from best-effort flows at the access level. Furthermore, in access network we can highlight two main cases:

- Wireline access networks, i.e. *Digital Subscriber Line* (DSL) or *Fiber To The...* (FTTx).

- Mobile access networks, i.e. *Long-Term Evolution* (LTE).

These cases are based on different network architecture, so they use different QoS mechanisms.

In wireline access networks, the *Local Area Network* (LAN) cannot be controlled by NSPs. The LAN is managed by its local administrator or in majority residential cases, left by default. The first treatment controlled by NSPs is done at the *Customer Premises Equipment* (CPE), e.g. a home gateway.

At the CPE level two traffic differentiation mechanisms are possible for the upstream traffic: VC-based and DSCP-based (**Janczukowicz et al. 2014**).

VC-based mechanism is used only in DSL. It uses ATM multiVC solution and a CPE with multiple local IP interfaces. Hence, there is one *Virtual Circuit* (VC) for Internet and a VC dedicated to conversational traffic. Packets generated by the trusted managed VoIP software are inserted in the conversational VC.

DSCP-based mechanisms are used in DSL and FTTx. Packets generated by VoIP software authorized by NSP are marked with an appropriate *Differentiated Services Code Point* (DSCP) tag. Later, at the CPE level they are separated from the best-effort traffic. Only authorized DSCP marking is accepted, any other marking is ignored, as not all end devices can be trusted by NSPs.

To control DSCP marking, the CPE can use a specific configuration. The CPE has a list of authorized destination addresses (belonging to *Session Border Controllers* (SBC), that will be explained later) and corresponding DSCP markings. As a result, the CPE can ignore any marking that is not preconfigured.

Another way to ensure trusted DSCP marking is to use the “heritage” mechanism. It is based on trusting downstream DSCP marking, i.e. marking coming from the network. The network is fully controlled by the NSP, so CPE can trust it. It tracks the connection and if it receives a traffic marked from network, the corresponding upstream flow inherits this marking.

When it comes to downstream traffic, the VoIP differentiated treatment is applied based on the fact that the packet comes from a known SBC or a dedicated IP-VPN, whose access is granted to flows having crossed a known SBC.

Historically, network operators were using VC-based mechanism because it could ensure more reliable and detailed QoS. Later they switched to IP mechanisms that are simpler and less expensive, but in exchange less powerful when it comes to QoS management. ATM multiVC is still used in DSL, because DSCP-based may introduce too big delay for conversational traffic in case of long lines with a weak bitrate. It is due to the fact that IP packets are bigger than ATM cells. In case of a weak bitrate it would take too much time to transfer an IP packet and it would influence other packets in the queue, whereas for short ATM packets, even when the speed is limited, the cells are smaller so packet forwarding takes less time and the delay is less bothering.

For the overview of mechanisms used in mobile access networks, we focus on *Long Term Evolution* (LTE) technology, that is explained in the books (**Bouguen et al. 2012**), (**Cox 2014**).

We can highlight three parts for this type of mobile access network: user equipment, radio access network and *Evolved Packet Core* (EPC).

The radio access network consists of base stations, called eNodeBs, that communicate with the user equipment.

The EPC, among other entities, contains a *Packet Data Network Gateway* (PDN-GW or P-GW). We focus on the P-GW, because it ensures the connectivity, but also plays an important role in the QoS implementation, since it participates in the *Evolved Packet System)* (EPS) bearer creation.

LTE support different types of services, e.g. voice, video streaming or non real-time data. Each of these services requires different network characteristics. To ensure an appropriate treatment of different services, along with a certain QoS, LTE introduces the concept of EPS bearers.

An EPS bearer is a connection between the user equipment and a P-GW. It can be viewed as a pipe with negotiated characteristics that allows more efficient resource allocation. It is characterized by a *QoS Class Identifier* (QCI) associated with certain network characteristics, notably packet delay budget and packet error loss rate.

A default bearer is established for each user equipment, when it registers to the network. It is used to provide a general connectivity. Later, additional bearers, called dedicated bearers, can be established for traffic that requires a certain level of QoS, notably voice services. Dedicated bearers can be created at any time, but after the default bearer establishment.

To identify the flows and associate them with appropriate bearers, traffic filters are used. These filters constitute a *Traffic Flow Template* (TFT). For uplink, TFT is applied by the user equipment and for downlink by the P-GW.

IP backbone

The IP backbone is similar for both mobile and wireline technologies. The traffic is separated between IP-VPNs so that VoIP media is separated from other traffic. In current implementations SBCs are used for traffic differentiation. There are two types of SBCs: *access* (A-SBC) and *interconnect* (I-SBC). These SBCs transfer the traffic and control access to dedicated IP-VPNs, so that only flows authorized to cross the SBCs can access the IP-VPNs.

Traffic differentiation in the IP backbone is not done for prioritization, since IP backbone can be easily provisioned compared to other network segments. However, this approach allows topology hiding and implementation of security solutions, e.g. an *Access Control List* (ACL) management. This enables protecting VoIP media plane equipment, like devices, media servers and gateways. It also simplifies traffic management and ensures fast rerouting in case of connection failures (**Janczukowicz et al. 2014**).

Interconnection

Interconnection is also similar for wireline and mobile. Typical Internet interconnection is best-effort. However, for managed VoIP, NSPs use specialized interconnections thanks to I-SBCs. Hence they can manage network capacity and traffic quality.

There are two types of specialized interconnections (**Janczukowicz et al. 2014**):

- Direct interconnection between Telco-CSPs.
- Interconnection with multiple networks by using connection hubs such as IPX with a given *Service Level Agreement* (SLA), so ensuring QoS.

To summarize, each NSP manages its own network resources and federates with other NSPs in order to offer a global managed VoIP solution. The coupling of application and network layers plays an important role, so that network has a visibility of flows entitled to benefit from specialized treatment. Managed VoIP traffic is also more predictable, so it is possible to anticipate necessary network capacity.

3.2.2 Coupling of application and network layers

We have already discussed in chapter 2 section 2.3 application layer mechanisms, used notably by WebRTC, that adapt to best-effort delivery in order to improve the quality of real-time applications.

Apart from application layer mechanism, there are various OTT approaches in providing QoS for communication services. We give several examples of efforts in coupling of application and network layers aiming at improving QoS and bandwidth utilization.

Unified Communications

Unified Communications (UC) is a set of products consisting of real-time communication solutions, e.g. VoIP, video conferencing and non real-time communication solutions, e.g. email, voicemail. This is essentially used within business environments.

There exist different UC solutions provided by companies like Microsoft and Cisco. Each company has its own way to improve QoS, however they are only limited to a given enterprise environment.

For instance Cisco, to improve network performance for its Unified Communication Networking, requires enabling QoS mechanisms in Cisco switches and routers throughout the network (**Cisco 2012**). Another example is Microsoft's Skype for Business that has a *Software Defined Networking* (SDN) interface to monitor the underlying network and analyze network traffic. Collected information enables optimizing media stream quality (**Skype 2016**).

Furthermore, UC vendors have created a UCI forum, that has integrated the *International Multimedia Telecommunications Consortium* (IMTC[1]). It was created to improve interoperability between different UC solutions and focuses on offering use cases and best practices. Moreover, IMTC *Real-Time Media Software Defined Networks* (RTM SDN[2]) works on improving the quality by providing APIs allowing interacting with the underlying network.

LLT

Latency Loss Tradeoff (LLT) draft focuses on separating IP packets in two classes of services: (`Lo`) low-loss service and (`La`) low-latency service (**You et al. 2016**).

In fact, the draft discusses that best-effort Internet privileges high utilization of bottleneck links and that it causes high queuing delay. It highlights that it may work for non real-time applications but penalizes interactive ones. As an improvement, a *Per-Hop Behavior* LLT group allows application to choose between low-loss or low-latency service. Thus, it enables trading loss for delay and the other way round.

However the document does not give any particular implementation solution for LLT.

[1]http://www.imtc.org/
[2]http://www.imtc.org/uc/ucsdn-work-group/

Devices APIs

W3C standardization organization members have shown an interest in working on APIs to access information about the underlying network (**W3C 2014a**) or users' data plans (**W3C 2014b**). These types of APIs would be helpful in providing interactive services. There were also discussions about how difficult it is to deploy this types of APIs, e.g. a need to have a specific contract with NSPs (**W3C 2014c**).

In fact, as a part of its W3C Device API working group, there was an effort of providing a Network Information API, hence an interface enabling accessing connection information of the device. Two properties were chosen to be exposed, i.e. bandwidth and information whether a given connection is metered.

However, the working group has encountered difficulties with useful connection information. Thus, the work has been discontinued for now (**Lamouri 2014**).

3.3 Summary

The aim of this chapter was to discuss the limits of the best-effort delivery. It also included an overview of certain QoS solutions, notably managed VoIP provided by NSPs.

Currently, best-effort delivery is mostly used, along with overprovisioning that aims at assuring appropriate network capacity. However, increased bandwidth is not necessarily essential for real-time applications. For them, latency is more essential and it has become a new bottleneck.

There is a growing number of challenging interactive applications provided by OTT communication service providers. They create real-time traffic that has different characteristics than typical non real-time data traffic. As a result, it would benefit from IP delivery adapted to its needs.

So far there is no global QoS solution for real-time applications. NSPs offer managed VoIP based on legacy telephony service approach. Hence they provide a service to their own subscribers and need agreements with other NSPs in order to expand it.

On the other side, OTT solutions mostly use adaptive mechanisms built into the endpoints. Other QoS efforts are either limited to networks that they control, e.g. enterprise networks, either demand assistance of device vendors and NSPs in order to be developed.

Chapter 4

Loose coupling strategies for improving OTT communication service quality

Contents

4.1 Research opportunities

4.1.1 Can managed VoIP principles apply to WebRTC?

In Chapter 3, we have shown how NSPs ensure differentiated traffic treatment while providing the managed VoIP solutions.

However, offering differentiated treatment to OTT communication services, notably WebRTC, is not straightforward.

Since managed VoIP benefits from differentiated treatment we compared its design with WebRTC technological choices. The study revealed the following issues preventing from directly applying principles of managed VoIP to WebRTC solutions (**Janczukowicz et al. 2014**):

- **Scalability issue**: The Internet is very heterogeneous. As a result, offering QoS services is complex. In managed VoIP each NSP has interface with its own network resources and federates with other NSPs to increase the reach of its service. However, in case of WebRTC solutions privileging P2P connections, there may be multiple, not necessarily cooperating, entities in the communication path. Additionally, endpoints can belong to different actors and be connected to various networks, e.g. mobile, wireline. Hence a number of participating actors and necessary interfaces would increase significantly. The scalability of this solution would be very difficult to manage.
- **Flow visibility issue**: Each NSP needs to distinguish the flows that should benefit from differentiated treatment. In managed VoIP this differentiation is ensured thanks to SBC based media steering distinct from best-effort routing at the user-network interface and network-network interface. In WebRTC, flow identification is complicated because of privileging P2P connection and mandatory encryption. Furthermore, there is no standardized signaling that could provide assistance in enabling flow visibility.
- **Troubleshooting and supervision issue**: In managed VoIP, troubleshooting and supervision of cross layer issues between the VoIP service and network resources, is internal and proprietary to each NSP. Enabling OTT communication service providers to offer a full troubleshooting and supervision service to its customers would require standard and open interfaces with multiple NSPs.
- **Capacity management issue**: In managed VoIP the management of network capacity depends on VoIP forecasts that are possible to predict. However, forecasts for OTT services are less accessible and difficult to predict. Hence, a more elastic capacity management would be needed.

In summary, NSP QoS mechanisms depend strongly on underlying network and cannot be easily provided to OTT actors. Furthermore, NSPs cannot give control over their networks to any third party, as it could cause security issues and would be too difficult to manage. Thus, as it was mentioned before, OTT can only provide QoS in networks they control, e.g. enterprise networks in case of business solutions, or use application layer mechanisms that do not interact with NSPs resources. NSPs ignore any unauthorized interactions with their networks, e.g. unauthorized packet marking explained in 3.2.1.

4.1.2 Research approaches

So far there has been no happy medium between OTT and NSP approaches. Generally speaking, OTT communication services are global and developer friendly, but are less reliable and vulnerable to changing network conditions. Whereas NSPs are known for services similar to traditional telephony, i.e. reliable, but not global and less attractive than OTT services.

Within the scope of reThink, an European project that focuses on designing non telecom centric service architecture, we have discussed two approaches of providing a solution for improving the quality of OTT communication services (**Copeland et al. 2015**):

- Over-The-Top approach - focusing on exploiting diversity of best-effort paths and collecting statistics in order to select the most advantageous path. This approach does not require NSPs assistance. OTT services can work independently by leveraging information provided by the end points and devices in the path to which they have access, e.g. TURN and STUN servers.
- In-Network approach - focusing on exploiting paths with differentiated traffic treatment. This approach requires cooperation of application and network layers. OTT services can benefit from specialized treatment offered by NSPs.

In this thesis we focus on the in-network approach. Therefore we study the possibilities of offering collaborative solutions. This study will allow us to test the feasibility of this concept but also to provide recommendations for improving WebRTC performance by coupling of application and network layers.

4.2 Driving forces

“How to improve the quality of OTT communication services by leveraging network mechanisms?” is an important question, not only from a technical point of view but also from an economic one.

Indeed, it is important to work on QoS mechanisms and evaluate their impact on different traffic types. Nevertheless, it is also essential to understand the driving forces for improving the quality of communication services. In fact, identifying who benefits from the quality improvements, can be a starting point for identifying means of providing an appropriate solution.

There are four principle actors that can be a driving force for providing quality solutions:

- User,
- OTT players,
- NSP,
- Regulatory.

The focus on issues and expectations of different actors, i.e. user, OTT and NSP, is done in the sections below.

We do not cover the regulatory aspect, since it is specific to a given country. Nevertheless, it is important to highlight that QoS could be used for public safety and emergency services. Already, regulations, e.g. in France (**Arcep 2007**), impose that NSPs provide free emergency calls and it could be possible to require a certain quality to ensure that these call can be made.

QoS solutions require investments, e.g. investing into additional network resources or equipment. Thus, they can increase the cost of service. Different actors may take a financial part in these solutions.

NSPs are directly impacted with a cost of these investments, since they manage network resources and implement QoS mechanisms.

Therefore, NSPs can decide to cover the investment cost by themselves and provide *neutral* services that improve the quality in general, regardless of the application.

Nevertheless, NSPs can decide to improve a quality only to certain actors. Hence a *monetized* approach can be identified, so NSPs can offer premium services to certain OTTs or to users.

4.2.1 NSP

NSPs would benefit from enhanced quality, since it would make their networks more attractive and that would lead to increase market share by increasing satisfaction of users.

NSPs would benefit from a solution that could not only improve the quality of real-time traffic, but also that could improve management of network resources, so overprovisioning would not be the only feasible mean of solving a problem of limited capacity.

Possible monetization of QoS could reward investments into network infrastructures.

4.2.2 OTT

OTT services can benefit from enhanced quality by offering a better communication service and differentiating from other OTTs, by eventually gaining more loyal users. As a result it would improve their market share.

Quality is important to OTT services, since there is a large number of OTT communication services, so if a user is not satisfied with one service, he can easily find a new one. This may impact the value of a given service, since it is linked to a number of loyal customers. A stable base of users has a direct influence on a growth of service, e.g. by giving access to potential new users, like friends and family (**Berezan et al. 2016**).

4.2.3 Users

Users would directly benefit from enhanced quality of communication services, as it would positively impact the comfort of their online communications.

It was believed that users were less demanding when using OTT communication services, due to the fact that these services are offered for free or for a very low price, e.g. the authors analysing the interest of developing premium services (**Teitelbaum & Shalunov 2002**) concluded that users would not notice failures until they were catastrophic.

However, more recent studies, e.g. (**De Pessemier et al. 2015**), indicate that poor quality and unreliability are the major disadvantages of web real-time communications. Also users accustomed to the very good quality of traditional telephony, expect the same performance from other types of communication services. Moreover, the perceived quality has a direct influence on users behaviour, e.g. call duration.

In fact, users require good and stable audio and video quality and fast call set up. As mentioned, users are accustomed to the reliability of traditional telephony. However, it is difficult to give precise characteristics of expected quality, since it depends strongly on the context, application used and subjective opinions of users (**De Pessemier et al. 2015**). Thus it is difficult to generalize users behaviour. Some customers may value reliability, whereas others may privilege lower costs.

For users, notably residential ones, it is not straightforward to demand a certain quality. Nevertheless, enterprises can be also considered end users. Furthermore, enterprises often have a network infrastructure adapted to their needs or a network administrator able to take care of different network configurations. Hence, they may be willing to invest in a solution providing communication services adapted to their needs .

4.3 Proposal of loose coupling strategies

Loose coupling is a term commonly used in software engineering, notably for service design when using web technologies. Pautasso, in (**Pautasso & Wilde 2009**), indicates that one of the main goals of using web technologies is to achieve loose coupling. According to his survey, in a loosely coupled service-oriented system, different elements can evolve in an independent way: a change to one service does not heavily impact the rest of the system, as long as it continues to provide the same functionality, described in its interface. As a result, a loosely coupled system can evolve and scale easily.

In the context of this thesis, loose coupling means that network mechanisms can be provided without imposing strong adaptation to NSP ecosystem, so the proposed strategies are in line with technological choices of web companies. Hence, even if network mechanisms evolve, they do not heavily affect the OTT services.

We have identified three strategies enabling loose coupling of OTT and NSP approaches, in order to provide solutions for improving quality of OTT communication services by leveraging network mechanisms. We focus mainly on new types of interactions between different actors.

4.3.1 NSP driven

In a NSP driven strategy (also called neutral strategy), the NSPs can improve the quality in general for all subscribers by providing appropriate treatment to different kinds of traffic, notably real-time traffic.

This strategy should take into account the following aspects when choosing a technical solution:

- Focus on a given NSPs network and be transparent to OTT services and users.
- Allow differentiaton of real-time and non real-time traffic.

4.3.2 OTT driven

In on OTT driven strategy, NSPs can improve the quality for a particular OTT service, so for a specific real-time application.

This strategy should take into account the following aspects when choosing a technical solution:

- Ensure a global reach by establishing a relationship between a given OTT service and NSPs.
- Allow differentiation of communication services per OTT service.

4.3.3 User/Enterprise driven

In User/Enterprise driven strategy, NSPs can improve the quality for given users, for a specific service or regardless of the application.

This strategy should take into account the following aspects when choosing a technical solution:

- Ensure a reach of a household or an enterprise network by establishing a relationship between a given user/enteprise and NSP.
- Allow differentiation of communication services:
  - per user,
  - per application.

  We propose these two possibilities notably for an enterprise use case, where the differentiation type would depend on a company policy. For instance, a company can prefer its employees to use certain applications over others. It can also decide to give different privileges per user, depending on employees' tasks.

4.4 Implementation examples for WebRTC

Given the identified strategies, in this section we propose different concepts, which can be used to enable loose coupling of OTT and NSP mechanisms and consequently enable improving the quality of OTT WebRTC based applications with the network's assistance.

4.4.1 NSP driven: neutral solutions

We define neutral solutions as mechanisms that aim to improve the quality in general for any type of application. These mechanisms should focus on the optimal treatment of different types of traffic, notably WebRTC, that would replace the widespread best-effort delivery.

Neutral solutions can consist in changing implementations in network equipment with algorithms that enable optimal packet treatment by trying to balance low delay and high throughput, e.g. *Active Queue Management* explained in section 5.4.2.

Another solution would be to give an application the possibility to explicitly choose a prefered treatment, for instance low-loss versus low-latency as explained in the LLT draft in section 3.2.2.

4.4.2 OTT driven: TURN-based architecture with brokering

In an OTT driven solution, we aim to improve the quality of a particular OTT service. To achieve it, its flows need to be isolated from the best-effort traffic. When isolated, this traffic can benefit from a differentiated service.

TURN-based architecture

The main requirement of the TURN-based approach is to ensure identification of flows that are eligible to benefit from a better quality. To achieve that, we leverage an existing managed VoIP solution explained in section 3.2.1.

SBC media steering plays an essential role in managed VoIP, since it ensures media steering of VoIP traffic. A TURN server, serving as a media relay, has some similarities with an SBC. So far, TURN servers are only used to relay the traffic and do not make any flow differentiation. Nevertheless, eligible media relays could be used to identify WebRTC-generated traffic, so that it can benefit from differentiated traffic treatment. Thus, TURN servers can be fundamental in designing a global OTT driven solution.

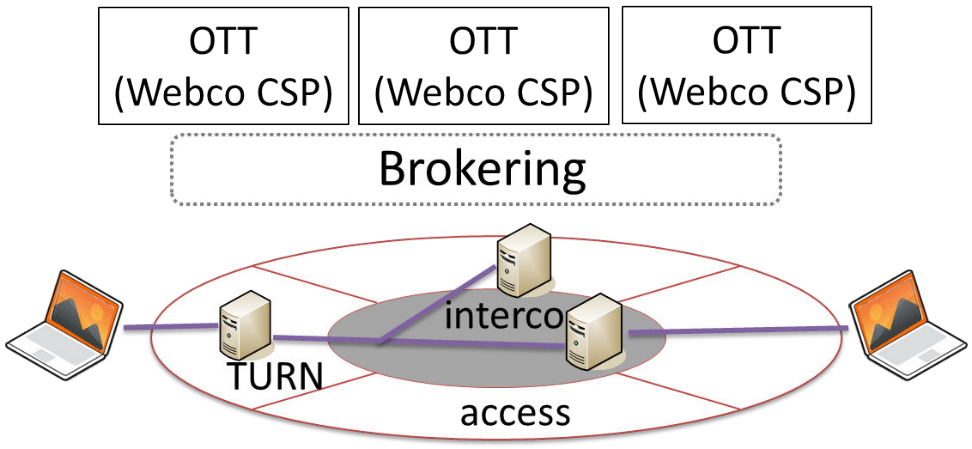


Figure 4.1 – TURN-based architecture concept

Figure 4.1 presents a TURN-based architecture (Janczukowicz et al. 2015). As it can be seen it impacts access and interconnection network segments.

- In access networks, a TURN server can be used for flow identification, i.e. all flows relayed by an eligibile TURN server can benefit from differentiated treatment. A TURN would act analogically to an A-SBC.
- For interconnection, a TURN overlay network can be created. This would allow creating peering agreements specific to eligible WebRTC flows. A TURN server would act analogically to an I-SBC.

A TURN-based architecture can be used by NSPs that want to improve the quality of real-time traffic of eligible OTTs and consequently their subscribers. Since OTTs subscribers do not necessarily belong to the same NSPs, this solution requires cooperation between OTTs and multiple NSPs. To achieve that, a single contact point is essential. This can be solved by providing a brokering service.

Given these points, there are three research areas in this architecture:

- access networks,
- interconnection,
- brokering.

In this thesis we focus on access networks and brokering concepts. We do not detail the interconnection aspects as peering agreements are concerned mostly by NSP agreements and economic aspects.

Brokering

There is a need of a solution for establishing a relationship between NSPs and OTTs, which would allow offering of global QoS mechanisms.

There can be a large number of NSPs concerned with a global OTT service, but an OTT cannot contact each NSP to ask for a specific QoS mechanism. Thus, to enable an establishment of relationship between different NSPs and OTTs, there is a need of a single contact point. To achieve that, an abstraction layer, i.e. a broker, has to be created, as it is presented on the figure 4.2.

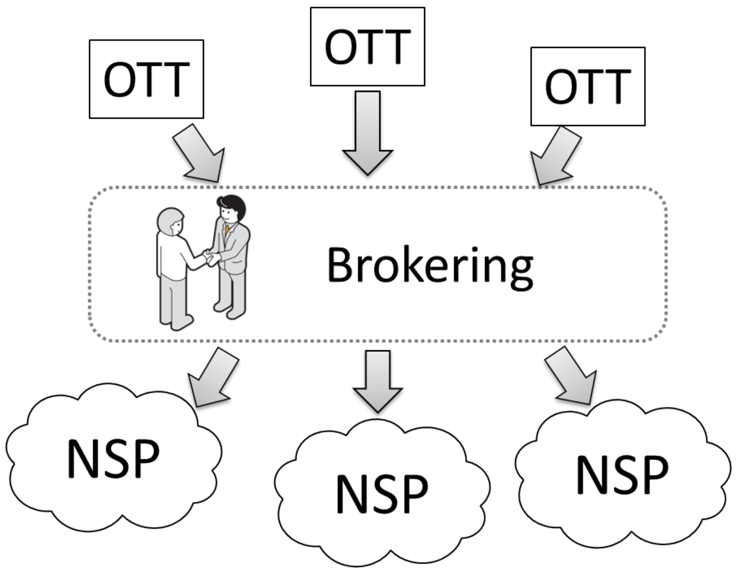


Figure 4.2 – Brokering concept

The aim of a broker is to enable OTT services to benefit from a differentiated treatment. In the TURN-based architecture context, it provides to a given OTT service the information concerning an eligible TURN server. Later, an OTT, by relaying its traffic through this TURN server, can benefit from the same QoS mechanisms used in wireline and mobile managed VoIP presented in the section 3.2.1.

The functionality of a broker is presented on the figure 4.3, only one side of a communication is presented to simplify the schema.

- 0: The OTT communication service provider subscribes to benefit from a differentiated treatment, before launching the WebRTC calls.
- 1: The User A connects to the OTT service and starts a WebRTC communication establishment.

- 2: The WebRTC application contacts the Broker to get information about a TURN server to use.
- 3: The Broker provides information about the TURN server to use. Different criteria of choosing a TURN server can be envisioned, e.g. a distance from a user.
- 4: The WebRTC flow is relayed through the TURN server and can benefit from a differentiated treatment.

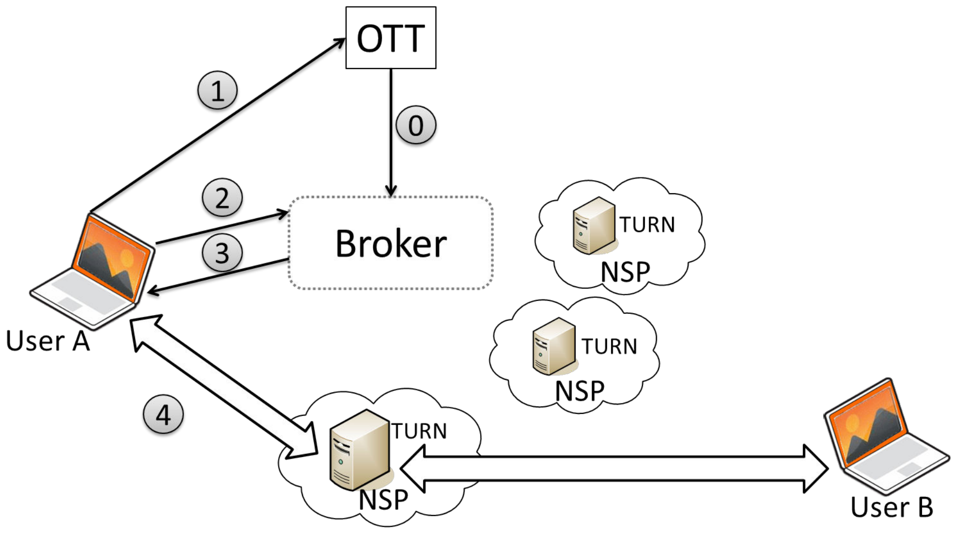


Figure 4.3 – Broker functionality

This brokering concept does not require major changes to WebRTC mechanisms. A WebRTC application needs to call a broker API to get TURN server information. Apart from that, there are no changes when configuring a TURN server to use.

The broker was studied and implemented by Orange Labs in the scope of the reThink European Project.

Wireline access networks

In this thesis we chose to focus on implementation and evaluation of TURN-based solution in wireline access networks. This choice was made for feasibility reasons, since this concept was implemented as a prototype.

Figure 4.4 is a schema of flow identification in wireline access networks (**Janczukowicz et al. 2015**).

WebRTC flows are marked with a given DSCP by an application. This application DSCP marking is only valid within a given *Local Area Network* (LAN). There exist recommendations for DSCP browser marking for WebRTC, that aim at improving WebRTC quality in (1) private, wide area networks, (2) residential networks and (3) wireless networks (**Jones et al. 2016**).

Application DSCP marking can be taken into account in an NSP network, i.e. based on this marking, the NSP can apply its own marking policy. For this reason, the application marking needs to be verified at the CPE level, e.g. the home gateway.

To validate the application DSCP marking, mechanisms known from managed VoIP can be used, i.e. CPE static configuration or heritage mechanism, explained in section

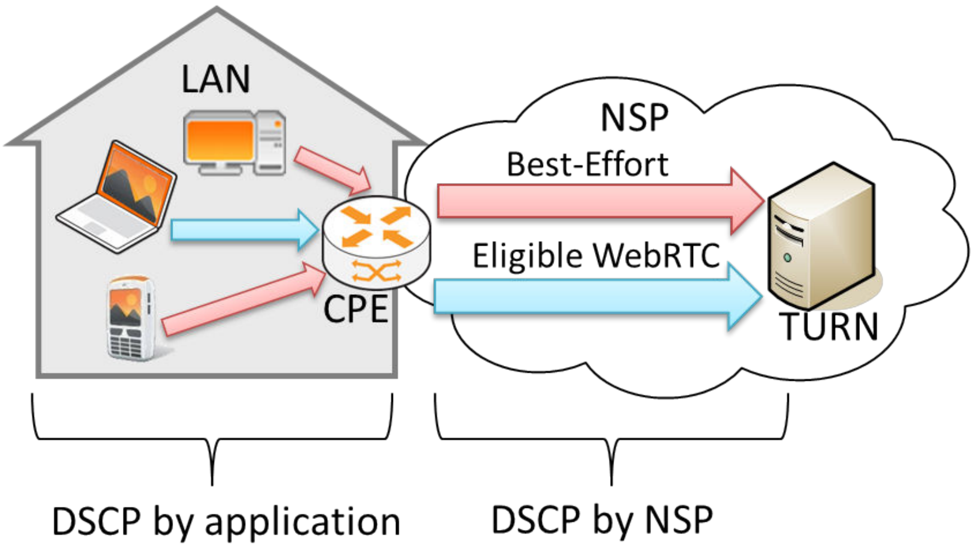


Figure 4.4 – Wireline access networks in TURN-based architecture

3.2.1. However, instead of using SBC as authorization element, we use eligible TURN servers. Hence, we assume that if a flow is relayed by an eligible TURN server, it can benefit from differentiated treatment and improved quality. Credentials or OAuth mechanism are used to enable authorization to access a TURN server (**Reddy et al. 2015**).

Consequently, at the CPE level, eligible WebRTC flows are separated from best-effort flows and put into different queues. Hence eligible WebRTC flows benefit from differentiated traffic treatment adapted to real-time traffic, e.g. different queuing mechanisms as it will be explained in the next chapter 5. Best-effort traffic and not eligible WebRTC flows are treated like a regular traffic.

4.4.3 User/Enterprise driven: enterprise policies

In User/Enterprise driven solution we aim to improve quality for a particular user or for a particular application. Here, eligible real-time traffic also needs to be isolated from the best-effort traffic in order to benefit from differentiated treatment.

The concept of enterprise policies solution is presented on the figure 4.5.

Enterprise policies solution gives an enterprise a possibility to manage OTT communication services used by their employees. This can be achieved by enabling the browser and mobile operating systems customization.

Hence, enterprise policies need to be coupled with browsers or operating systems used by employees, e.g. it is possible to modify the Google Chrome code to trigger an appropriate DSCP marking for WebRTC flows depending on application or on a particular user. Moreover, the NSP needs to map this marking to traffic management at the CPE level, e.g. certain flows will be prioritized over other.

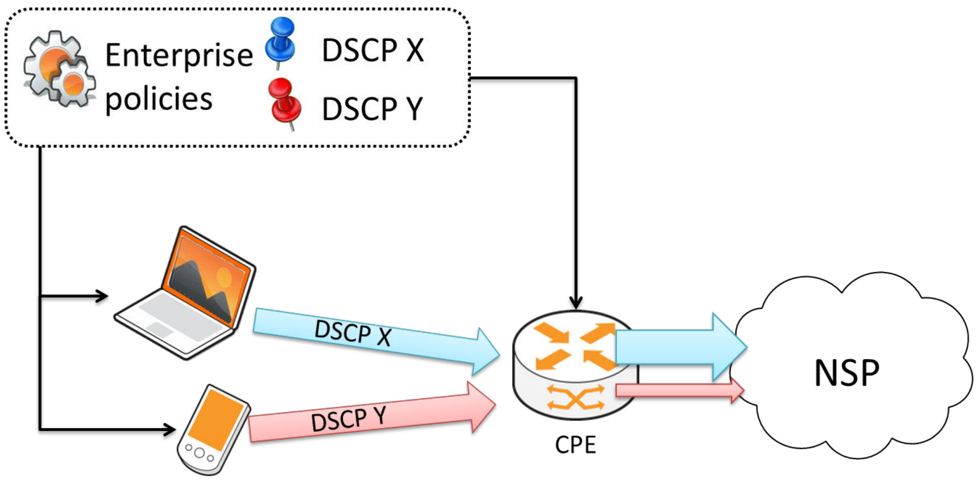


Figure 4.5 – Enterprise policies concept

4.5 Summary

In this chapter, we have shown that managed VoIP principles cannot directly apply to WebRTC. However, we have discussed that certain concepts can be reused, e.g. DSCP marking or SBC-like media steering.

We have also presented two research approaches to improve OTT communication services, i.e. in-network and over-the-top. We have focused on in-network approach that leverages network assistance when working on OTT solutions quality improvement.

Furthermore, we have identified three loose coupling strategies for improving OTT communication service quality: NSP driven, OTT driven and User/Enterprise driven.

Based on these strategies, we have proposed implementation examples for WebRTC quality improvement. These concepts show different levels of coupling between application and network layers.

The advantage of the proposed implementations is that they leverage existing mechanisms. Thus, they do not impose any important changes to the WebRTC mechanisms by adapting it to NSP ecosystem. On the contrary, they are in line with web technologies and WebRTC standardisation efforts. As a result, they demand little or no change to existing WebRTC applications.

Chapter 5

Traffic management solutions adapted to WebRTC

Contents

5.1 Current Internet engineering practices

5.1.1 Widespread best-effort delivery

We have already mentioned, in section 3.1.4, that the widespread best-effort delivery is not appropriate for real-time traffic. We are going to detail these aspects in this section.

The main reason of current Internet engineering practices is that historically, networks were aiming at privileging high utilization, i.e. maximizing throughput as presented in the study on reducing Internet latency (**Briscoe et al. 2014**). For this reason, it has been common to use large buffers aiming at improving the network utilization and minimize the packet loss. Even though, buffers are essential in packet networking, e.g. to absorb data bursts, when they are too large and frequently full, they can cause important delays, as discussed in (**Gettys & Nichols 2011**) and (**Baker & Fairhurst 2015**).

Based on existing bibliography (**Baker & Fairhurst 2015**), (**Grigorescu et al. 2013**), in best-effort delivery a traditional queueing mechanism is used: FIFO. So the packets are queued in a buffer, in order that they arrive, until they are transmitted. When the buffer becomes full, the arriving packets are dropped.

In network equipment, certain buffer lengths are configured, and by minimizing packet loss, they privilege throughput. However, these buffers are often large and, given the growth of Internet and the increased amount of traffic sharing the same resources, this approach is a source of different issues.

First of all, it can perturb loss-based congestion control, notably for TCP flows. Loss-based TCP congestion control mechanisms probe for capacity and increase their rate until they detect a packet loss. Long queues postpone packet loss, so cannot signal a congestion problem. Consequently, the congestion control mechanisms do not know that they should decrease their sending rate and they end up filling the buffers.

This can cause an excessive queueing delays that negatively influence the delay for other flows sharing these buffers. This is especially the case when large packet bursts are received. Moreover, more aggressive flows can cause starvation of other traffic. Since real-time traffic is less greedy and does not use aggressive congestion control algorithms, it struggles to provide good performance when competing with concurrent traffic, notably in the presence of TCP flows.

The study (**Briscoe et al. 2014**), indicates that delay, caused by packet queuing at network devices, has a major influence on the overall end-to-end path delay. Moreover, the queuing mechanisms have an essential impact on coexistence of different types of flows.

Focus on wireline access networks

In section 3.1.1, we have identified that quality problems are mostly caused in access networks and peering points.

Peering areas, as discussed before, are in a large part concerned by economic aspects and inter NSPs agreements, so they are not detailed in this thesis that deals with more technical concepts.

In wireline access networks, it is common to see quality issues and congestion problems. The study (**Briscoe et al. 2014**) indicates that large buffers can cause delays at any congestion point in the end-to-end path, but it also emphasizes that currently this issue is mainly noticed at the edge of the network. In fact, for most users, their experience is determined by their edge networks provided by NSP, as it impacts service connectivity, availability and reliability, as explained in (**Kreibich et al. 2010**).

A wireline access link is shared between different flows. How much is going to be attributed to a given real-time flow depends on the total link capacity and on the type of the concurrent traffic (**Briscoe et al. 2014**). Hence, user experience is directly linked with the increased number of devices on aggregated access link, but also with the fact that, as mentioned above, best-effort delivery is not adapted to real-time traffic requirements and works better for non real-time traffic like TCP.

In fact, overbuffering, causing excessive delays, is a known problem in wireline access networks. Netalyzr measurements, (**Kreibich et al. 2010**), confirmed overbuffering in end user access devices for uplink. The most common buffer sizes in this study were 128kB and 256kB. Such large buffers for 1Mbit/s uplink, during e.g. a file transfer, can introduce a delay of over 1s.

Therefore, in our study we focus on access networks. We decided to restrict the thesis to wireline aspects and the scenarios used for evaluation were done for uplink emulation, since uplink bandwidth is usually more limited than downlink, notably in ADSL. This choice was made for feasibility reasons. Thus, we were able to test a wide range of technical aspects and evaluate identified hypothesis in close to real world conditions.

5.1.2 Overview of TCP congestion control

TCP by design, based on (**Grigorik 2013**) and (**Briscoe et al. 2014**), guarantees reliable delivery of packets and it privileges an accurate delivery even if it is done at the cost of delay. TCP traffic is designed to efficiently use the underlying network. In fact, TCP probes for network capacity, so it continuously tries to increase its sending rate until it detects packet loss.

To determine the available capacity and appropriate sending rate, TCP uses the Slow Start concept. At this stage, the sender initializes a congestion window. It has a limit of data that it can send, i.e. that can be in flight, before it receives an acknowledgment. Later, the sender sends the data to the receiver. For each received acknowledgment, it increases its congestion window. Hence, in general it doubles the congestion window for every RTT. This is done, until it detects congestion or reaches a Slow Start threshold.

Then, during the congestion avoidance state, standard TCP uses an additive-increase, multiplicative-decrease model. However, this was considered ineffective, because it takes too much time to recover after every packet loss. In case of congestion, it decreases the congestion window size by two, but later increases it only by one segment per RTT.

To improve TCP performance, different algorithms were proposed. Their aim is to make congestion window reduction smaller. Moreover, they assume faster recovery.

For instance, Linux uses TCP Cubic, (**Ha et al. 2008**), that aims at maximizing throughput. It changes the way the congestion window behaves. When a packet loss occurs, it decreases the congestion window by a defined, constant factor. At the same time, it uses a cubic function for a better congestion window recovery.

Given these points, it can be concluded that TCP congestion control is agressive and aims at assuring high bandwidth utilization.

Furthermore, at the bottleneck, the traffic from one flow impacts the delay and packet loss of other flows. Thus, as it is indicated in the survey (**Briscoe et al. 2014**), delay-based congestion controllers risk to be starved in the presence of loss-based congestion control mechanisms, that end up filling up the buffers.

5.2 Network requirements for WebRTC

As discussed, best-effort delivery is not adapted to real-time traffic, but is optimised for non real-time traffic, notably TCP.

Real-time traffic is not considered as greedy, since it is not aggressive towards other traffic, but it needs a particular level of protection (**Lundqvist 2011**). However it is not easy to achieve, especially in the presence of coexisting TCP flows that tend to be greedy and bursty. In case of congestion or link saturation caused by TCP traffic, real-time services suffer from quality degradation caused by delay and packet loss, whereas TCP traffic goes into recovery stage, meaning it decreases its sending rate and sends the postponed packets when the network recovers (**Kim et al. 2009**).

The relevant network requirements for real-time traffic, notably WebRTC, may vary depending on the context and application that is used. However, there are several parameters that are traditionally associated with real-time applications, i.e. bandwidth, latency, jitter and packet loss ratio (**Campedel 2007**).

5.2.1 Variable bandwidth

WebRTC media flow contains packets that have more or less fixed sizes. Generally, voice packets are relatively small, whereas video packets can be bigger (**Lundqvist 2011**). Moreover, flow characteristics can quickly change because of muting and unmuting, activating and deactivating video, changing video input (e.g. using a different webcam or switching to desktop sharing) or because of the impact of congestion control mechanisms built in applications. Hence, there is a certain unpredictability associated with real-time application behaviour.

In fact, bandwidth in Opus and VP8, both used in WebRTC, can vary significantly, as discussed in section 2.3.2.

According to measurements in the existing literature, in Google Chrome browser, for audio, the Opus mono bitrate is 32kbit/s and the stereo bitrate is 64kbit/.

Regarding the video, in Google Chrome browser the starting video bitrate is 300kbit/s, the minimum bitrate is 50kbit/s and the maximum is about 2Mbit/s (**Khan nown**).

5.2.2 Delay and jitter

The study written by Chong and Matthews (**Chong & Matthews 2004**) highlights that for speech, one-way delays below 150ms are not noticed by the human ear, whereas delays between 150 and 250ms are acceptable, but noticeable to the human ear. Moreover, ITU-T provides the E-model (**ITU-T 2015**), which is a computational model that can be used to estimate the impact of one-way delay on the quality of conversational speech. However, there are no similar models for non-speech interactive applications.

ITU-T recommendations, given in (**ITU-T 2003**), indicate that for general network planning purposed, the one-way delay should be kept under 400ms. Nevertheless, the same document highlights that one-way delays below 100ms can impact the highly interactive tasks, like video conferencing and any interactive applications in general. Generally, to provide transparent interactivity for speech and non-speech applications, the one way delay should be kept below 150ms.

To our knowledge, there is no explicit information about the impact of delay on WebRTC perceived quality. However, there is a study on performance of WebRTC congestion control for video flows in the presence of delay (**Singh et al. 2013**). It has shown that in the case of the Google Chrome browser, the GCC congestion algorithm performs well when the latency does not exceed 200ms. But, for latencies that exceed 200ms, it does not provide good bandwidth utilization.

Jitter is defined as a variance in the latency. It is created since there is a difference between the transmission delays of different packets belonging to the same flow (**Campedel 2007**). It is caused by the fact that there may be varying queue lengths or even routes encountered by these packets (**ITU-T 2003**). Jitter causes sound distortions and its recommended acceptable maximum value is 75ms (**Chong & Matthews 2004**).

5.2.3 Packet loss

In most cases, WebRTC media relies on UDP, a protocol that does not provide a guarantee of message delivery, so there are no acknowledgements or retransmissions (**Grigorik 2013**).

Acceptable packet loss values depend on used codecs and concealment algorithm. However, generally for audio, it should be less than 3% and for video less than 1% (**ITU-T 2001**).

To handle packet loss for video flows in WebRTC, the hybrid FEC/NACK approach is used, that was explained in section 2.3.3. However, to our knowledge there is no explicit information about the impact of packet loss on WebRTC perceived quality.

Nevertheless, it is important to highlight that loss based controller in GCC reacts, i.e. decreases sending rate, when packet loss for video is more than 10%, as it was detailed in section 2.3.4. Furthermore, the existing evaluation studies of WebRTC implementation in Google Chrome browser have shown that the GCC congestion algorithm for video performs well in the presence of packet loss, even when it reaches 10%. However, the performance drops in the presence of both, packet loss and latency (**Singh et al. 2013**).

5.3 Focus on queuing mechanisms

At the bottleneck, mutliple flows compete for network resources. As a result, traffic from one flow impacts the performance of other flows.

In this thesis, we study how different queueing mechanisms may be helpful in providing optimal and differentiated treatment of different types of traffic.

We have chosen Linux to perform this study in order to ensure the feasibility of it, since it enables configuration of different queuing mechanisms.

5.3.1 Queuing mechanisms definitions

First of all, before going into details, it is important to explain what queuing mechanisms are and how they are defined in the literature. The vocabulary, that is going to be used throughout this thesis, is given in this section.

Wallace, in (**Wallace 2004**), defines *queueing mechanism* as a congestion management. It specifies how to send traffic from multiple streams out of a bottleneck link.

Briscoe et al., in (**Briscoe et al. 2014**), define *packet (or queue) scheduling* as a mechanism a network device uses in order to decide which packet should be send next in case of a queue of multiple packets. They divide queuing mechanisms into:

- Class based, that classifies packets into different classes associated with specific treatment.
- Flow based, that orders packets based on flow characteristics.
- Latency specific, that schedules packets to obtain a given latency.
- Hierarchical, that allows creating a hierarchy of treatment.

The same authors also defined *queue management* as a mechanism allowing network devices to monitor a queue size and taking actions when this queue is filling up, e.g. by dropping packets. It divides queuing management into:

- Passive, that drop packets from the tail (drop tail) or from the front (drop front) of the queue, when a given queue is full.
- Active, that act proactively to achieve certain queue characteristics.

In Linux, for traffic management purposes, there is *traffic control*. Brown, in (**Brown 2006**), defines it as a set of queuing mechanisms and systems that are used when packets are received and transmitted on a router. According to (**Ubuntu nown**) and (**Brown 2006**), it consists of:

- Shaping, that allows controlling a transmission rate and can be used to smooth out the bursts of traffic. It consists in delaying packets to meet a certain output rate, so it occurs on egress.
- Scheduling, that allows arranging the transmission of packets, i.e. it orders the packets between the input and output of a given queue.
- Classifying, that allows separating packets into different queues.
- Policing, that allows measuring and limiting traffic of a given queue. It accepts packets until a given rate, so it occurs on ingress.
- Dropping, that allows discarding packets.

Additionally, Linux defines a *queuing discipline*, qdisc, (**Brown 2006**) as a scheduler that orders packets to be sent according to given rules. There are two types of qdiscs:

- Classful, i.e. a scheduler that can contain classes to which it can attach other qdiscs.
- Classless, i.e. a simple scheduler, not containing any classes.

5.4 Overview of existing queuing mechanisms

5.4.1 Classless

In this section, we give an overview of the classless queuing mechanisms, which are fundamental in Linux (**Brown 2006**).

We have chosen two widely known and simple mechanisms: FIFO and SFQ.

FIFO

First In, First Out (**Lundqvist 2011**) is widely known and is the most basic queuing discipline. As a consequence, it is often used as a default mechanism. FIFO is based on the principle that all packets are put in the same queue and treated equally. The order in which packets are placed in the queue indicates the order in which packets are served. When the queue is full, any new arriving packets are dropped.

There are two variants of FIFO: (1) managed in packets, i.e. *packet FIFO* (pFIFO) and (2) managed in bytes, i.e. *byte FIFO* (bFIFO) (**Brown 2006**).

The only parameter of FIFO is its length, i.e. limit. FIFO is available in Linux[1].

SFQ

Stochastic Fairness Queueing (SFQ) (**McKenney 1990**), (**WirelessConnect nown**) is a queuing discipline that aims to ensure fairness between flows.

It uses a hashing algorithm to divide traffic into a number of FIFO queues. To avoid that different flows end up being associated to the same queue, the hashing algorithm is run every perturb value of seconds. Furthermore, the round robin is used to dequeue packets.

SFQ has an important number of parameters that can be configured. Nevertheless, all of them are provided with default values. SFQ is available in Linux[2].

5.4.2 AQM

We focus on *Active Queue Management* (AQM) mechanisms, because there is an increased interest in this domain, linked with an increased concern about the importance of network latency.

AQM mechanisms, explained in (**Khademi et al. 2013**) and (**Kulatunga et al. 2015**), are used to solve problems of excessive delays due to large buffers and they aim at absorbing burst of packets but without creating long standing queues. This is achieved by dropping packets before a given buffer is filled up.

AQM mechanisms have been known for two decades, notably *Random Early Detection* (RED). Nevertheless, they were never widely developed, because of performance problems and complex tuning. However, newer mechanisms claim to overcome this issue by focusing on parameter less, also known as "no-knobs" schemes, as discussed in (**Kulatunga et al. 2015**), (**Nichols & Jacobson 2012**).

RED

Random Early Detection (RED) (**Floyd & Jacobson 1993**) is an AQM mechanism based on average queue size. It tracks queue size and drops packets gradually (from the end of the queue) with a certain probability.

Packets are not dropped if the average queue size is smaller than a configured minimum threshold (min). However, when average queue size becomes bigger than min, packets start to be dropped. Packets are dropped with a changing probability, which increases linearly along with the average queue size value. It increases until the average queue length reaches a configured maximum threshold (max). At this point the probability reaches its maximal

[1]http://manpages.ubuntu.com/manpages/trusty/man8/tc-bfifo.8.html
[2]http://manpages.ubuntu.com/manpages/trusty/man8/tc-sfq.8.html

possible value, i.e. predefined max-probability. It is possible that the average queue size becomes larger than max.

In RED, keeping the average queue size close to a target value requires a complex tuning. RED performance is difficult to predict as it depends on the congestion level. For instance, when the link is congested, but max-probability is too small, the average queue size can end up above max threshold. However, when there is a light congestion, but max-probability is big, the average queue size will be kept close to min threshold. As a result, without precise tuning parameters, it is complex to ensure that a certain average queue size will be achieved.

RED is available in Linux[3].

ARED

Adaptive Random Early Detection (ARED) (**Floyd et al. 2001**), (**Khademi et al. 2013**) was created in order to improve RED by providing the ability of keeping an average queue length around a certain target value.

ARED allows adjusting max-probability so that the average queue size is kept between a given range of values, i.e. half way between max and min. Max-probability changes from 1% to 50% and it increases additively and decreases multiplicatively.

ARED requires fewer configurations than RED. Thresholds (min and max) can be configured or calculated based on target queue length (limit). It is available in Linux, as an option to RED.

Codel

Controlled Delay (Codel) (**Nichols et al. 2016**), (**Khademi et al. 2013**) is based on the principle that a queue is *bad* if it persists for several RTTs. Hence, it avoids building up the queue and it ensures that the link is not underutilized.

Codel tracks how much time packets spend in a buffer, i.e. the minimum queuing delay packets experienced for a given interval of time. It compares this queueing delay with a defined acceptable target delay in order to detect a persistent queuing delay.

Interval should be chosen appropriately, so that packets bursts can be absorbed. It should provide a balance in detecting a persistent queue, so avoid false detection but be able to react fast enough. It is suggested to use interval equal to the worse RTT, i.e. the highest RTT of all connections that share a buffer. It is also suggested to use target that equals 5%-10% of RTT as it allows good performance and does not cause link underutilisation.

No packets are dropped if the minimum queueing delay is less than target. However, if the minimum queing delay is greater than target for at least interval of time, Codel enters the dropping state and drops a packet. Packets are dropped at the dequeue, i.e. head of the queue, at the calculated dropping rate, that starts at 1 packet per RTT (this value can be derived from interval) and slowly increases until the minimum queueing delay becomes smaller than target. When it happens, Codel exits the dropping state. Moreover packets are not dropped if the queue risks emptying out completely or containing less than MTU bytes.

Although, Codel was designed to require no configuration, its settings can be modified. It is available in Linux[4].

[3]http://manpages.ubuntu.com/manpages/wily/man8/tc-red.8.html

[4]http://manpages.ubuntu.com/manpages/xenial/man8/tc-codel.8.html

FQ Codel

Fair (or Flow) Queue Controlled Delay (FQ Codel) (Hoeiland-Joergensen et al. 2015) is a packet scheduler and AQM algorithm at the same time. It isolates sensitive traffic, e.g. interactive traffic and it improves network performance thanks to preserving short queue lengths.

FQ Codel classifies packets into different FIFO queues based on IP 5-tuples, i.e. source IP address, destination IP address, source port, destination port and protocol. Additionally, there is a Codel scheme associated with each queue. A modified *Deficit Round Robin* (DRR) scheduler is used and every iteration, each queue can send up to a given number of bytes, i.e. quantum of bytes.

The scheduler can distinguish between *new* and *old* queues, that is, the queues that do not build up from the queues that are present for more than one iteration. The *new* queue is emptied up within an iteration, i.e. it builds up less than quantum of bytes before the scheduler's visit. Moreover, the *new* queue has a priority over the *old* one. The scheduler manages an order of iteration by keeping a separate list for *new* and *old* queues.

Furthermore, the DRR scheduler tracks deficit on each queue. The initial deficit value equals the value of quantum. When a queue dequeues packets, it decreases deficit by the size of dequeued packets until it becomes negative. When it happens, the queue's iteration ends and deficit is increased by quantum's value. Later, if the queue was in the list of *new* queues, it is moved at the end of the list of *old* queues. And if the queue was already on the list of *old* queues, it is moved at its end.

Each packet goes through the following stages. First, when a packet is enqueued, it is classified into an existing queue or initialises a new queue. Then it is added to the tail of the given queue. If the total of enqueued packets is bigger than the configured limit, a packet is dropped from the head (beginning) of the queue, regardless the queue.

Second, the scheduler selects a queue to dequeue from. It starts from the list of the *new* queues and later moves on to *old* queues. When a queue to dequeue is chosen, Codel algorithm is used on it and a packet is dequeued while deficit is updated. If the queue is emptied, it is either (1) moved to the end of the list of *old* queues if it was on the list of *new* queues before, or (2) removed, if it was already on the list of *old* queues. It prevents a queue from reappearing every iteration and makes sure that packets from the *old* queues also get a chance to be sent.

As a result, in FQ Codel packets are dropped from the queues that are building up. Also thanks to DRR, flows with large packets are not penalizing other flows, since for the same quantum a flow with small packet sizes will be able to send more packets than a flow with large packet sizes.

FQ Codel requires nearly no configuration and is available in Linux[5].

PIE

Proportional Integral controller Enhanced (PIE) (Pan et al. 2016) is a lightweight AQM scheme that drops packets randomly with a probability calculated based on a trend of queuing delay.

Packet queuing delays are calculated based on their dequeue rate and the queue length. Later, dropping probability is systematically updated with a defined frequency, called tupadte. It is calculated based on queuing delay compared to configured expected target

[5]http://manpages.ubuntu.com/manpages/xenial/man8/tc-fq_codel.8.html

delay and its trend, i.e. whether it increases or decreases. For this purpose, it uses a *Proportional Integral* (PI) controller, which is is enhanced with adjusting calculation to network congestion issues, e.g. dropping probability is decreasing exponentially instead of linearly when congestion ends (details are in pseudocode given in (**Pan et al. 2013**)). Packets are dropped with a calculated probability before enqueuing, so as they arrive.

PIE is defined to require no configuration, but its parameters can be modified. It is available in Linux[6].

5.4.3 Classful

Classful queuing mechanisms allow separating traffic into different classes. We have chosen to focus on *Hierarchical Token Bucket* (HTB), because as according to (**Brown 2006**), it enables complex and granular traffic control, thanks to a variety of sophisticated techniques. Also according to HTB evaluation studies, (**Ivancic et al. 2005**), it is believed to be faster, more intuitive and more precise than other queuing mechanisms of this type developed in Linux.

HTB

HTB, defined in (**Devera 2002a**), (**Devera 2002b**) and (**WirelessConnect nown**), allows handling different types of traffic by organizing them in classes with defined allocated bandwidth.

HTB is based on token buckets principle. A token bucket consists of generating tokens with a certain rate and placing tokens in a bucket. So whenever a packet arrives and there is an available token, it can be sent. However, if there are no available tokens, i.e. the bucket is empty, the given packet is held until a token is available (**Brown 2006**).

Furthermore, HTB is also a classful queuing discipline. It means that it allows creating different classes, i.e. sub queues to which it can attach other queuing disciplines. To put packets into appropriate classes it uses filters, e.g. based on DSCP tag.

In order to use HTB a tree needs to be created. The root class corresponds to the whole link. The root class and any lower classes can have child classes attached.

The lowest classes are known as leaf classes. Each leaf class has a queuing discipline qdisc attached to it, since it is positioned at the beginning of packet treatment, so it enqueues and dequeues packets. The default qdisc attached to leaf classes in HTB is FIFO.

Each class has the following principal parameters:

- Ceil - maximum rate that class can use.
- Rate - guaranteed rate of class.
- Prio - priority of class used to decide on order of packets.

Moreover, each class can be in one of three modes:

- Green - class has not used its guaranteed rate yet. Class can send.
- Yellow - class has used its guaranteed rate, but it is allowed to borrow from its parent. It has not reached its ceil yet. Class can send.

[6]http://manpages.ubuntu.com/manpages/wily/man8/tc-pie.8.html

- Red – class has reached its ceil, so it cannot borrow anymore. Class cannot send.

Each class can send until it reaches its rate and can borrow until it reaches its ceil. A class can send quantum of bytes before other classes are served. It is important to note, that classes that are in a green mode can always send before classes that are in a yellow mode, no matter the prio, since guaranteed rates have to be satisfied. Prio is used to determine the order in which leaf classes are served. Also the excess bandwidth is divided to lower classes based on their prio.

HTB configuration is required. First of all, a tree needs to be created, so parent and child classes have to be indicated. Later, rate and prio are mandatory to be configured, whereas other parameters can be configured or left to default values. HTB is available in Linux[7].

It is important to highlight that classful HTB can be used together with classless queuing disciplines, e.g. presented above FIFO, SFQ or ARED. Classless queuing disciplines can be attached to leaf classes. This configuration can enable separating traffic into different classes and providing them with differentiated treatment.

5.5 Proposal of traffic management solutions adapted to WebRTC

We have explained why the current Internet engineering practices are not adapted to real-time traffic.

Here, we focus on evaluation of traffic management solutions adapted to real-time traffic, notably to WebRTC communication services. We focus on mechanisms that privilege low latency and that, at the same time, ensure the coexistence of WebRTC traffic and TCP flows.

There are two possibilities of addressing these aspects. Either by decreasing queuing delay for all traffic, either by isolating the sensitive traffic. Thus, we identify two directions in choosing appropriate queuing mechanisms:

- *Target delay based queuing mechanisms*, that consist in monitoring queuing delay and taking actions when a queue is filling up.
- *Rate and queue length based classful queuing mechanisms*, that consist in separating traffic with different characteristics into different classes.

For each of these directions, we have selected queuing disciplines or a combination of queuing disciplines. These solutions will be evaluated later. We are going to study how they interact with WebRTC and the GCC algorithm in the presence of typical TCP loss-based congestion control.

5.5.1 Target delay based queuing mechanisms

In the target delay based approach, we focus on mechanisms that balance between low delay and high throughput, while using delay as principal criteria.

[7]http://manpages.ubuntu.com/manpages/trusty/man8/tc-htb.8.html

The recent work in AQM focuses on creating dropping policy based on queuing delay measurements or estimations and not on the queue size or average queue. This approach is fundamental for schemes like Codel, FQ Codel and PIE.

We have decided to evaluate FQ Codel and PIE. We have chosen FQ Codel for its capability of differentiating between *old* and *new* queues. We have also selected PIE for its adaptive dropping probability that is calculated based on a trend of queuing delay.

The details of this configuration are given in the following section 6.3.5.

5.5.2 Rate and queue length based classful queuing mechanisms

In the rate and queue length based classful approach, we focus on mechanisms that allow explicitly separating traffic into different classes. These classes are adapted to the characteristics of traffic they contain. Hence, they are configured to offer a certain rate and queue length.

In this context, we use HTB for its ability to classify packets into different sub queues that are configured with different queuing disciplines.

We define two classes: 1) one class for best-effort traffic and 2) one class for WebRTC traffic.

In best-effort class we use mainly the FIFO queuing discipline, since it corresponds to the current practices. However, we also use ARED since we wanted to check best-effort traffic in the presence of queuing discipline different than FIFO, but which still uses a queue size as a principal criteria.

In WebRTC we use SFQ, because it divides traffic into FIFO queues, while assuring the fairness between flows. We have chosen it, because in case of multiple WebRTC flows, all of them can receive the same treatment.

The details of this configuration are given in the following section 6.3.6.

5.6 Summary

In this chapter, we have discussed that current engineering practices, notably queuing mechanisms, are not adapted to the real-time traffic.

We have given an overview of network requirements for real-time traffic, based on WebRTC example. We have also created a survey of existing queuing mechanisms that could be reused in our solution.

Therefore, we have identified two possibilities to offer the traffic management adapted to real-time traffic, i.e. 1) aiming at assuring lower delay regardless the traffic or 2) isolating the sensitive traffic.

Accordingly, we have identified two possible directions when defining the traffic management adapted to WebRTC:

- Target delay based queuing mechanisms,
- Rate and queue length based classful queuing mechanisms.

For each of these directions, we have selected queuing disciplines or a combination of queuing disciplines. They will be implemented and evaluated in the following chapters.

In our study, the wireline access networks were chosen for implementation, because they give a possibility of testing our hypothesis in close to real world conditions.

Chapter 6

Proof of Concept

Contents

6.1 Test environment

In the previous chapter 5, we have presented our proposals of traffic management adapted to the WebRTC traffic. We have identified two approaches in proposing appropriate queuing mechanisms:

- Target delay based queuing mechanisms,
- Rate and queue length based classful queuing mechanisms.

In this chapter we describe a *Proof of Concept* that will be used for evaluation of different traffic management approaches. It will allow us to analyse how the identified solutions impact the WebRTC traffic, notably how they interact with WebRTC congestion control algorithm in the presence of long-lived TCP flows.

As it was mentioned in the section 5.1.1, the PoC is set for emulation of uplink wireline access networks and we will emulate two technologies: ADSL and fiber.

WebRTC traffic

The main aim of the PoC is to evaluate the impact of different configurations on real-time traffic. In our study we have focused on WebRTC solutions, since it is well documented. As a result, we are able to retrieve information about WebRTC statistics.

The WebRTC traffic characteristics were discussed in section 5.2.

Concurrent traffic

TCP traffic was chosen as concurrent traffic, as it is widely used for non real-time traffic. Its typical behaviour was discussed in the section 5.1.2.

We have decided to use long-lived TCP connections, for instance a file upload, since we wanted a connection that would impact the bottleneck buffer for a time that is long enough to observe its effect on WebRTC.

6.1.1 Testbed architecture

The testbed used for tests in wireline access network is presented on the figure 6.1. This testbed configuration enables emulating uplink wireline access network.

The testbed consists of the following elements:

- The CPE acts as a gateway between the local network and the Internet. In case of excessive traffic on the access link, the bottleneck is created at the CPE level. Furthermore, identified queuing mechanisms can be implemented on the output interface for the upstream traffic of the CPE, with Linux qdisc. Additionally, Internet access network is emulated with netem.
- WebRTC Server is used to provide the WebRTC application and to exchange WebRTC call establishment information between WebRTC Users.
- WebRTC User A and WebRTC User B are used to perform a WebRTC call. User A is in the local network, whereas User B acts as a remote user.
- Upload Server is a simple web server that allows TCP file uploads.
- Concurrent Traffic represents any traffic that is launched in addition to the analysed WebRTC call.

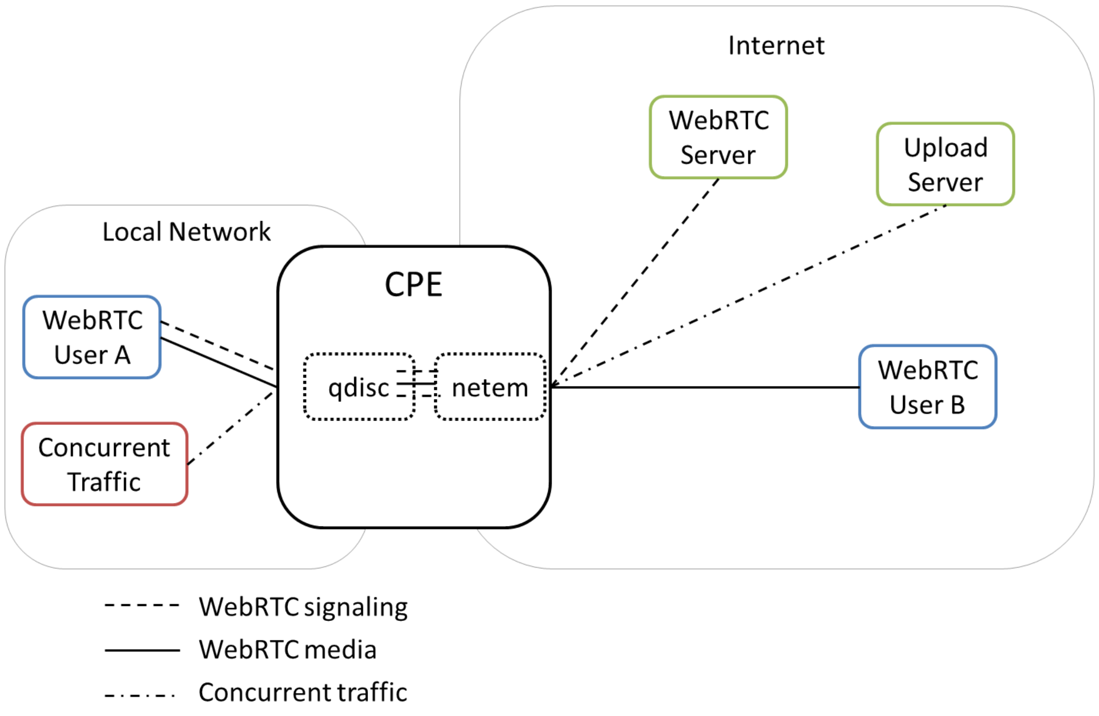


Figure 6.1 – Testbed

In short, the presented testbed allows us to create test scenarios that consist in establishing a WebRTC communication, along with injecting a concurrent traffic.

Therefore, it is possible to observe the impact of the concurrent traffic on WebRTC media flows, especially in case of congestion. Additionally, it allows observing how different CPE configurations impact the WebRTC and concurrent traffic.

6.1.2 CPE and Internet access functionalities

A Linux machine with iptables is used to provide CPE functionalities. Iptables[1] configuration is used for routing packets between the local network and the Internet. Furthermore, tested queuing mechanisms are executed on the output interface of CPE for the upstream traffic.

Linux Traffic Control

As it was mentions before, Linux was chosen fo feasibility reasons, because it enables configuration of different queuing mechanisms.

Siemon, in (**Siemon 2013**), presents an overview of queuing in the Linux network stack. The simplified schema of Linux network stack is given on the figure 6.2.

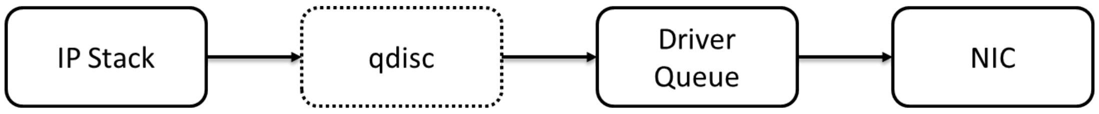


Figure 6.2 – Simplified Linux network stack

[1]http://www.netfilter.org/projects/iptables/index.html

The IP Stack provides complete IP packets to be transmitted by the NIC, i.e. *Network Interface Controller*.

The Driver Queue is used to hold packets so they can be available to the NIC for transmission. It uses a simple FIFO queuing mechanism, since it simplifies NIC driver design.

Siemon, in (**Siemon 2013**), indicates two problems caused by Driver Queue size: (1) starvation and (2) increased latency.

NIC starvation, i.e. reduced system throughput, happens if the NIC drains packets faster than packets are queued, so at some point, it drains the queue and if it has nothing to dequeue, it misses the transmission opportunity. Large queue size is used to ensure high throughput and avoid starvation, but they also introduce higher latency.

Thus, the qdisc, i.e. queuing discipline, placed between IP Stack and Driver Queue, is used for traffic management, notably traffic classification, shaping and prioritization. For this purpose Linux *traffic control* (TC[2]) is used. This command allows configuring queuing mechanisms presented in previous chapter 5, without introducing complex changes to Driver Queue or NIC.

CPE interface configuration

In our PoC, two actions need to be taken on the output interface of CPE: (1) access network and Internet emulation and (2) identified queuing mechanisms configuration.

Netem, network emulator, is used for Internet and access network emulation. However, since it is not possible to configure netem and other qdisc on the same interface, we created a virtual bridge that we attached to the physical interface. The configuration is presented on the figure 6.3.

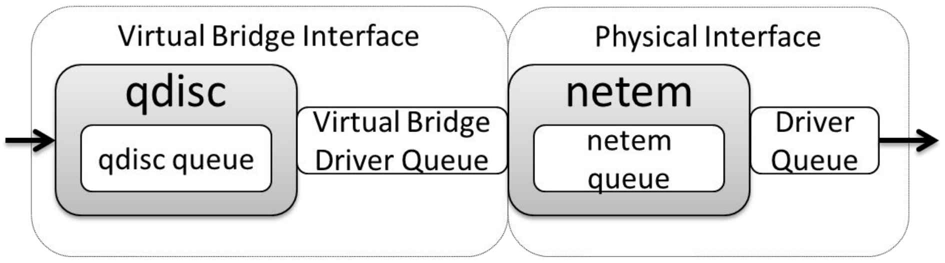


Figure 6.3 – CPE interface configuration

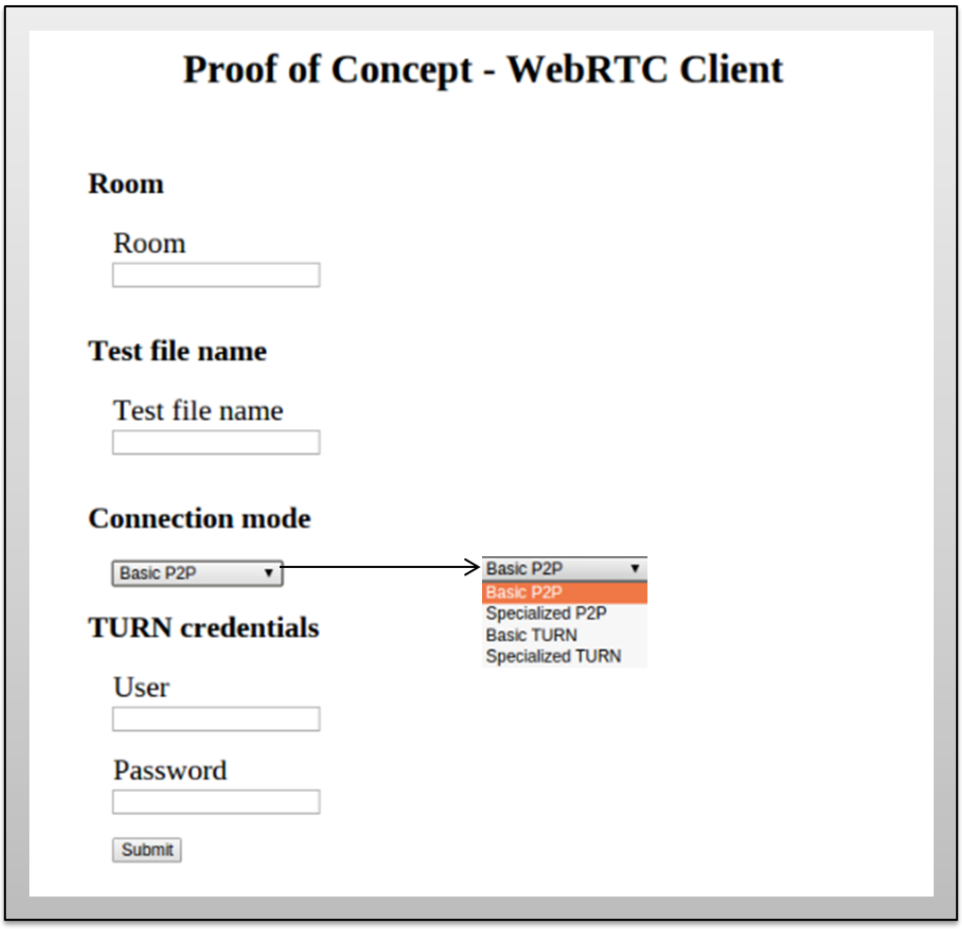
- The Physical Interface is used for Internet and access network emulation. Netem[3] is used to emulate the rate of an access link and a delay of packets in the Internet. The length of netem's queue is set to 100 packets, so it would not introduce unmanaged packet drops. The Driver Queue length is left by default (1000 packets), since it is negligible given the physical interface speed, especially in the presence of netem configuration.
- The Virtual Bridge Interface is used for CPE queuing mechanisms configuration, i.e. qdisc configuration managed by TC. The Virtual Bridge Driver Queue length by default equals 0, so when defining qdisc the queue length must be explicitly configured depending on tested queuing mechanism.

[2]http://manpages.ubuntu.com/manpages/wily/man8/tc.8.html

[3]http://manpages.ubuntu.com/manpages/trusty/man8/tc-netem.8.html

6.1.3 WebRTC application

We developed a basic WebRTC application that allows establishing video calls between two users, WebRTC User A and WebRTC User B. The interface is given on the figure 6.4.



Proof of Concept - WebRTC Client

Room

Room

Test file name

Test file name

Connection mode

Basic P2P

TURN credentials

User

Password

Submit

Figure 6.4 – WebRTC application interface

.

WebRTC Users use Chrome version 47 running on Linux machines. We chose Chrome since it provides the most advanced WebRTC features, i.e. GCC implementation, DSCP marking and rich statistics information. We chose version 47, as it was the most up to date when we started the test campaign.

Each WebRTC User, before launching a call, can specify the communication type (P2P or with TURN server) and whether DSCP marking should be activated or deactivated (basic or specialized).

To enable or disable DSCP marking we use the googDscp option in Chrome when creating the connection. Pfeiffer, in her blog post (**Pfeiffer 2016**), gives a description of DSCP activation in Google Chrome.

As we have tested, Chrome decides which DSCP marking will be used based on flow characteristics: *Best Effort* (BE) if googDscp is off and *Assured Forwarding* (AF) or *Expedited Forwarding* if googDscp is on. However, for our test purposes it is enough to know that when googDscp is deactivated, DSCP value equals zero and when it is activated, DSCP value is different than zero.

The WebRTC Server runs on a Linux machine and implemented with nodejs[4], i.e. a JavaScript runtime that allows rapidly implementing network applications. This WebRTC Server provides the WebRTC application to the users and enables the information exchange (ICE candidates, SDP messages) between users. It also allows connecting to a data base in order to save browser statistics per test case, which will be explained below.

6.1.4 Concurrent traffic

Concurrent Traffic is any traffic that is launched in addition to the observed WebRTC call, e.g. any TCP traffic or another WebRTC call.

Generating WebRTC traffic is straightforward, as the above WebRTC server can be used.

There are two possibilities of generating TCP traffic. First, we have developed Upload Server, also using nodejs. It runs on a Linux machine and provides a web interface for file uploading. It gives the progress of the update and measures the total upload time. It also connects to a data base in order to save measured upload time per test case, that will be explained later.

Second, TCP traffic can be generated with iperf[5], that is generally used for active measurements of maximum bandwidth on a given link. However, in this PoC it is used for generating TCP traffic between two Linux machines on different sides of the CPE.

6.2 Data collection for evaluation

We have based evaluation on three types of measurements. The measurements are collected at the application level, network level and they are enriched with a subjective opinion about perceived quality.

The data collection approach is presented on the figure 6.5.

Application and network layers measurements are saved in one data base.

6.2.1 Network layer measurements

The Wireshark[6] protocol analyser is used to collect data at the network layer. Wireshark captures are performed on the outgoing interface of the CPE.

From these captures, the bitrate of WebRTC and concurrent traffic is extracted. Obtained values are saved in the data base. Later, they are used to calculate channel utilization.

Channel utilization is the average bitrate obtained by Wireshark for a given traffic type and divided by available bandwidth. It can be calculated for WebRTC and for TCP traffic.

6.2.2 Application layer measurements

Application layer measurements can be obtained from the upload application and WebRTC application.

[4]http://nodejs.org
[5]http://iperf.fr
[6]http://www.wireshark.org

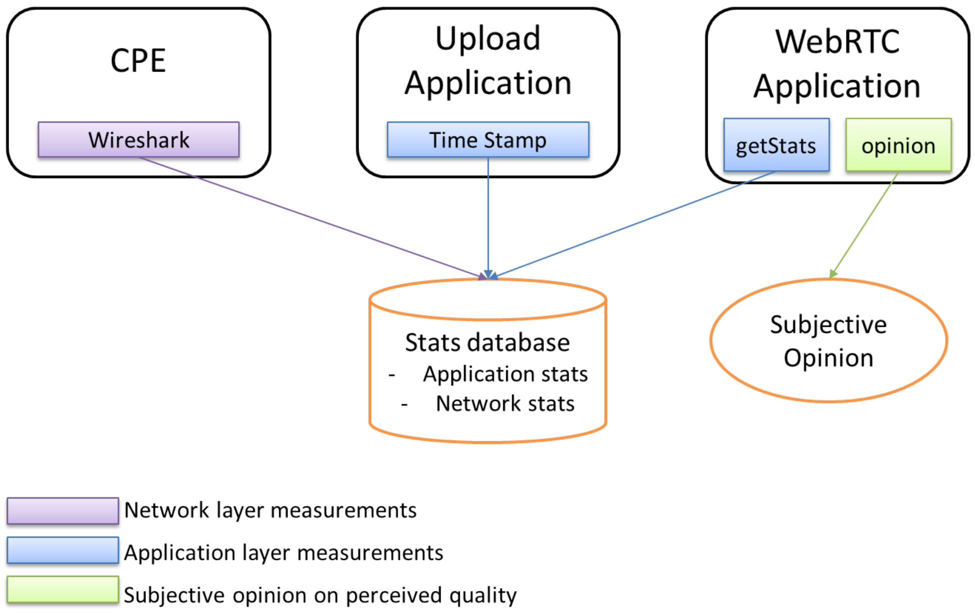


Figure 6.5 – Data collection of measurements for evaluation

Upload application provides information about the upload time of a given file. For this purpose a simple time stamp is used. The obtained data is saved in the data base.

WebRTC application measurements are obtained with getStats, i.e. an API that enables accessing statistical information of a given WebRTC call. GetStats is started whenever a WebRTC call is launched and is called every 2 seconds.

GetStats allows obtaining a large number of statistics. The detailed list of available statistical information is managed by W3C (**W3C 2016**). In the PoC, we have focused on general stream statistics, media stream statistics and transport statistics.

We have selected the *Key Performance Indicators* (KPI) that we identified as relevant and that are automatically saved to the data base. The selected KPIs are given in the table 6.1.

Audio KPIs	**Video KPIs**
number of audio packets sent	number of video packets sent
number of audio packets received	number of video packets received
audio RTT	video RTT
audio packet loss	video packet loss
audio bitrate sent	video bitrate sent
audio bitrate received	video bitrate received
audio jitter	video frame rate sent
	video frame rate received
	video frame width/height sent
	video frame width/height received

Table 6.1 – Selected KPIs from WebRTC getStats

However, since in the evaluation presented in this thesis, we have focused on the assessment of GCC WebRTC performance, we needed to use metrics applicable for video. Thus, in our evaluation we focused on the following metrics selected from available KPIs:

- video bitrate,
- video packet loss ratio, i.e. number of lost packets divided by number of sent packets,
- video frame rate,
- video frame width, i.e. a size of a video as observed by the WebRTC user.

6.2.3 Subjective opinion on perceived quality

PoC tests are attended. As a result, it is possible to observe subjective, perceived quality of experience. It is important since not all information can be obtained from analysis of measured statistics, e.g. image and sound quality or general comprehension.

It is an important aspect that can enrich the evaluation when compared to related studies (presented in section 7.4) that do not take subjective opinion into account.

So far, subjective opinion was done while performing the tests. However, it would be interesting to create more specific user tests, taking into account opinions of a larger focus group or using tools specific for quality of experience measurements.

6.3 ADSL networks - PoC configuration

6.3.1 ADSL emulation settings

In our tests, we focused mainly on ADSL networks, since they correspond to the majority of subscriptions in France.

According to ARCEP, in the second quarter of 2016 in France there were 27.220 million broadband (high-speed Internet) subscriptions, including 22.450 million ADSL subscriptions (**Arcep 2016**). Furthermore, for Orange France, based on an internal documentation, 80% of ADSL subscriptions correspond to users that benefit from 1Mbit/s or less for uplink.

As it was mentioned before, we used `netem` to emulate access networks. Since in PoC we are going to principally focus on use cases concerning ADSL uplink, we configure the rate of 1Mbit/s. Furthermore, based on existing ARCEP quality measurements of ADSL wired networks, we have chosen to configure `netem's` delay to 20ms (**Arcep 2015**).

6.3.2 WebRTC call

There is a need to guarantee reproducibility of performed tests. Hence, instead of using a live webcam and microphone capture, video and audio files are injected in the WebRTC call.

We used a file that consists of several sequences, each lasting 30 seconds. In each 30 second sequence, there is one speaker. The speaker changes between the sequences, along with the background that changes from simple to more complex.

We decided to use different sequences of videos connected together, because it allowed us to observe WebRTC behaviour in the presence of bitrate fluctuations, caused by sequence changes. As a result, we could evaluate if WebRTC is able to gain back its bitrate in the presence of concurrent TCP flow. Furthermore, the sequence changes may correspond to switching between cameras or switching to desktop sharing.

The chosen video has a 4CIF (704 x 576) resolution and frame rate of 25 fps. The Opus codec for audio and the VP8 codec for video are used.

The Chrome browser allows inserting files in order to use them as webcam and microphone flows, however video and audio need to be placed in separate files. The video file needs to be in the *.y4m* format, whereas the audio file needs to be in the *.wav* format.

6.3.3 Concurrent traffic

For ADSL network emulation, we use the upload server to upload a file with TCP. The file that we upload is a simple 5MB photo. This upload can correspond to a typical use case, e.g. uploading a picture to the cloud, sending an e-mail with an attachment, etc.

6.3.4 Configuration for droptail queuing mechanisms

The Droptail configuration corresponds to the current IP network engineering that is widely used for best-effort delivery. It assumes that in network devices, buffers are configured with simple FIFO queuing discipline. Hence, only a limit, i.e. a length of a given buffer needs to be configured. When a buffer is full, any packet above the defined limit is discarded.

As mentioned above, overbuffering in uplink access networks can cause excessive delays. In current networks, different sizes of buffers are present. The study using the Netalyzr tool, (**Kreibich et al. 2010**), (**Gettys 2011**), have shown that for ADSL uplink of 1MBit/s, buffer size can vary from 4kB to 500kB.

We have decided to study the impact of different buffer sizes on WebRTC communication and more precisely on the behaviour of congestion control for video. We have selected five different buffer sizes for the tests: 16kB, 32kB, 64kB, 125kB, 250kB. Additionally, we have tested a buffer size of 1000 packets that corresponds to the default Linux value for FIFO.

In Linux TC, when configuring FIFO[7], `limit` is the only parameter to change.

Configuration summary

The summary of evaluated configurations for droptail is given in the table 6.2.

6.3.5 Configuration for target delay based queuing mechanisms

For target delay based queuing mechanisms, we have chosen to evaluate FQ Codel and PIE mechanisms. Both of them require nearly no configuration, however their parameters can be modified if needed. In our configurations, we left most of the parameters to default, in order to evaluate the impact of suggested values on WebRTC communication.

[7]http://manpages.ubuntu.com/manpages/trusty/man8/tc-bfifo.8.html

CPE config name	TC qdisc	Parameters settings
Droptail 16kB	fifo	limit=16kB
Droptail 32kB	fifo	limit=32kB
Droptail 64kB	fifo	limit=64kB
Droptail 125kB	fifo	limit=125kB
Droptail 250kB	fifo	limit=250kB
Droptail 1000p	fifo	limit=1000 packets (default Linux)

Table 6.2 – Droptail configuration

For FQ Codel[8], we use the default values, notably for target delay and interval. However, we disabled ecn marking. ECN stands for *Explicit Congestion Notification*, that enables notification of network congestions rather than dropping packets (**Ramakrishnan et al. 2001**). Since, ECN is still marked as future work for GCC, (**Holmer et al. 2015**), we decided to disable this option.

For PIE[9], we also use the default values, notably for target delay. We also disabled the ecn option. Additionally, we activated the bytemode parameter. As a result, the drop probability is scaled proporionally to packet size.

Configuration summary

The summary of evaluated configurations for target delay based queuing mechanisms is given in the table 6.3.

CPE config name	TC qdisc	Parameters settings

FQ Codel	fq_codel	limit=10240 packets (default)
		flows=1024 (default)
		target=5ms (default)
		interval=100ms (default)
		quantum=1514 (default)
		noecn (ecn turned off)

PIE	pie	limit=1000 packets (default)
		target=20ms (default)
		tupdate=30ms (default)
		noecn (ecn turned off)
		bytemode on

Table 6.3 – Target delay based configuration

Note: According to the best practices for Linux configurations[10], when configuring FQ Codel and PIE, first, it is advised to use HTB on the whole interface, to limit connections

[8]http://manpages.ubuntu.com/manpages/xenial/man8/tc-fq_codel.8.html
[9]http://manpages.ubuntu.com/manpages/wily/man8/tc-pie.8.html
[10]http://www.bufferbloat.net/projects/codel/wiki/Best_practices_for_benchmarking_Codel_and_FQ_Codel/

to the link speed, i.e. 1Mbit/s in our case, so any overbuffering on the next device will not have a negative impact.

6.3.6 Configuration for rate and queue length based classful queuing mechanisms

We have configured HTB to classify packets into separate classes, i.e. one queue for best-effort, notably the concurrent traffic, and another for eligible WebRTC traffic. Each queue has a queuing mechanisms adapted to the traffic characteristics it contains.

Dependencies between HTB classes

In HTB[11] there are three main parameters to configure: rate, ceil and prio. Furthermore, there can be different dependencies between best-effort and WebRTC classes. Hence, we propose two configurations:

- HTB 204

  In this configuration, higher priority (Prio) is given to the eligible WebRTC traffic than to the best-effort traffic. Moreover, WebRTC is guaranteed to obtain a minimal bandwidth necessary to establish a video call.

  For eligible WebRTC traffic, Ceil is set to available bandwidth, i.e. 1Mbit/s and Rate is set to 204kbit/s, i.e. the sum of:

  – VP8 codec minimum bandwidth[12] coresponding to a 180p video stream resolution (100k for codec rate + 50k overhead[13] for 180p video stream),
  – Optimal bandwidth for Opus[12] (32k codec rate + 22k overhead).

  At the same time, for the best-effort traffic, Ceil is set to available bandwidth, i.e. 1Mbit/s, whereas the guaranteed Rate of the best-effort traffic is limited to a minimum, i.e. 8bit/s in our setup.

  In this configuration, the class corresponding to WebRTC, can be in Green and Yellow HTB sending mode, whereas best-effort traffic will be most of the time in Yellow HTB sending mode.

  Since the eligible WebRTC traffic has higher priority, it will be always served before best-effort traffic (in Green and Yellow HTB sending mode). It may be essential for the links with a very low bandwidth in order to ensure a minimal acceptable quality of WebRTC communication. It may be also used in cases when communication services are supposed to be privileged over other traffic.

- HTB 654

  In this configuration, higher priority (Prio) is given to the best-effort traffic than to the eligible traffic, but the WebRTC class is configured to provide a bandwidth sufficient to provide an optimal quality of communication.

  For eligible WebRTC traffic, Ceil is set to available bandwidth, i.e. 1Mbit/s and Rate is set to 654kbit/s, i.e. the sum of:

[11] http://manpages.ubuntu.com/manpages/trusty/man8/tc-htb.8.html
[12] http://www.webrtc-experiment.com/webrtcpedia/
[13] The values of overhead were obtained by measuring network bitrate and codec bitrate of a reference WebRTC call performed on the testbed.

  - VP8 codec min bandwidth for 360p video stream resolution[12] coresponding to a medium quality video (500k codec rate + 100k overhead for 360p video stream),
  - Optimal bandwidth for Opus[12] (32k codec rate + 22k overhead).

At the same time, for the best-effort traffic, Ceil is set to available bandwidth, i.e. 1Mbit/s, whereas the guaranteed Rate of the best-effort traffic is limited to minimum, i.e. 8bit/s in our set up.

In this configuration, analogically the class corresponding to WebRTC, can be in Green and Yellow HTB sending mode, whereas most of the time, best-effort traffic will be in Yellow HTB sending mode.

As a result, WebRTC will obtain guaranteed rate (Green HTB sending mode) to be able to ensure an optimal communication quality. However, concerning the remaining bandwidth (when both classes are in Yellow HTB sending mode), the best-effort traffic will be served first since it has higher priority.

The summary of the above configurations is given on the figure 6.6.

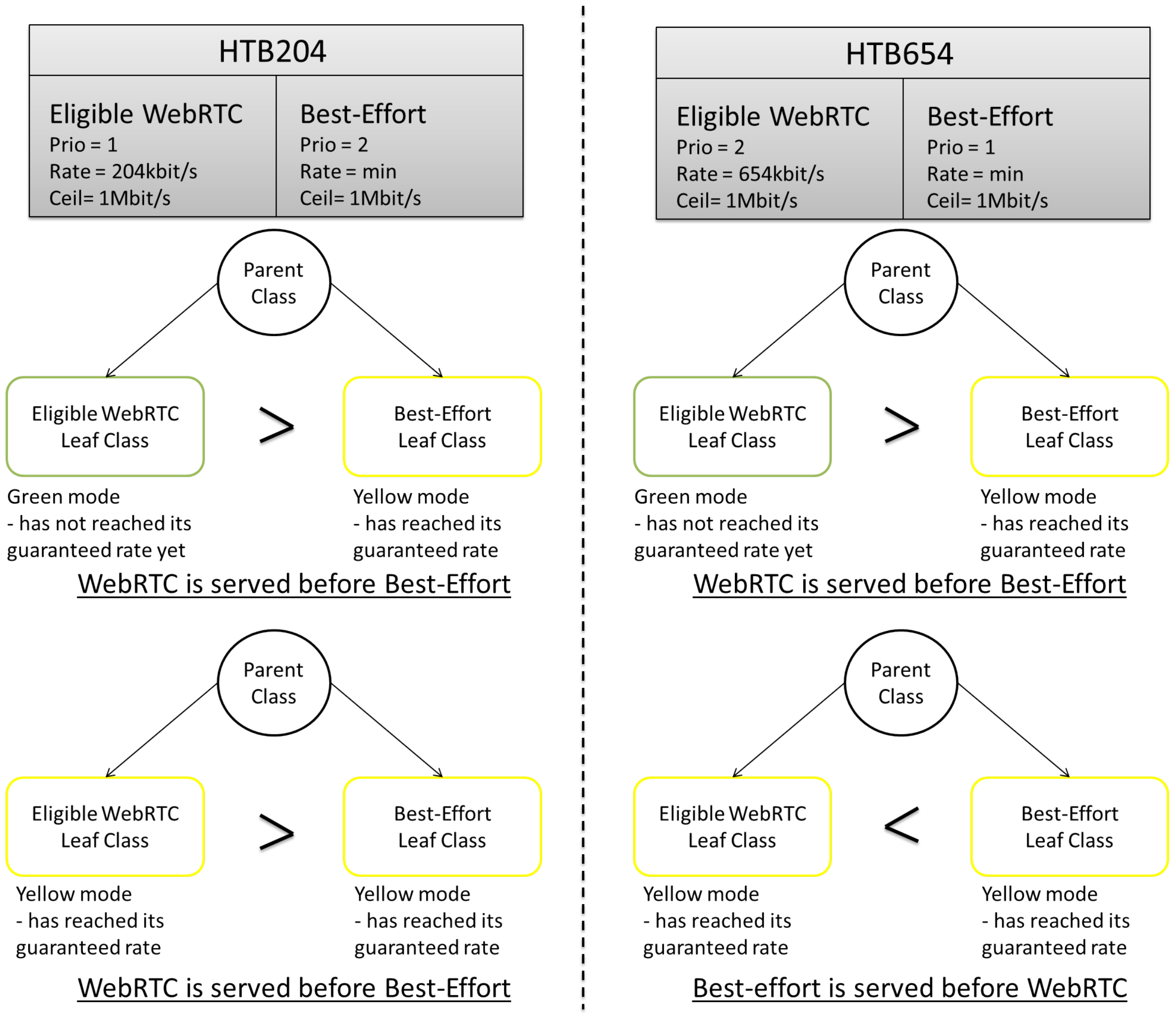

HTB204
Eligible WebRTC
Prio = 1
Rate = 204kbit/s
Ceil= 1Mbit/s
Best-Effort
Prio = 2
Rate = min
Ceil= 1Mbit/s
Parent Class
Eligible WebRTC Leaf Class
>
Best-Effort Leaf Class
Green mode - has not reached its guaranteed rate yet
Yellow mode - has reached its guaranteed rate
WebRTC is served before Best-Effort
Parent Class
Eligible WebRTC Leaf Class
>
Best-Effort Leaf Class
Yellow mode - has reached its guaranteed rate
Yellow mode - has reached its guaranteed rate
WebRTC is served before Best-Effort
HTB654
Eligible WebRTC
Prio = 2
Rate = 654kbit/s
Ceil= 1Mbit/s
Best-Effort
Prio = 1
Rate = min
Ceil= 1Mbit/s
Parent Class
Eligible WebRTC Leaf Class
>
Best-Effort Leaf Class
Green mode - has not reached its guaranteed rate yet
Yellow mode - has reached its guaranteed rate
WebRTC is served before Best-Effort
Parent Class
Eligible WebRTC Leaf Class
<
Best-Effort Leaf Class
Yellow mode - has reached its guaranteed rate
Yellow mode - has reached its guaranteed rate
Best-effort is served before WebRTC

Figure 6.6 – Dependencies between classes for HTB204 and HTB654 configurations

Each HTB configuration contains one queue for eligible WebRTC traffic and one for best-effort traffic. Each of these classes has different queuing discipline attached, as presented on the figure 6.7.

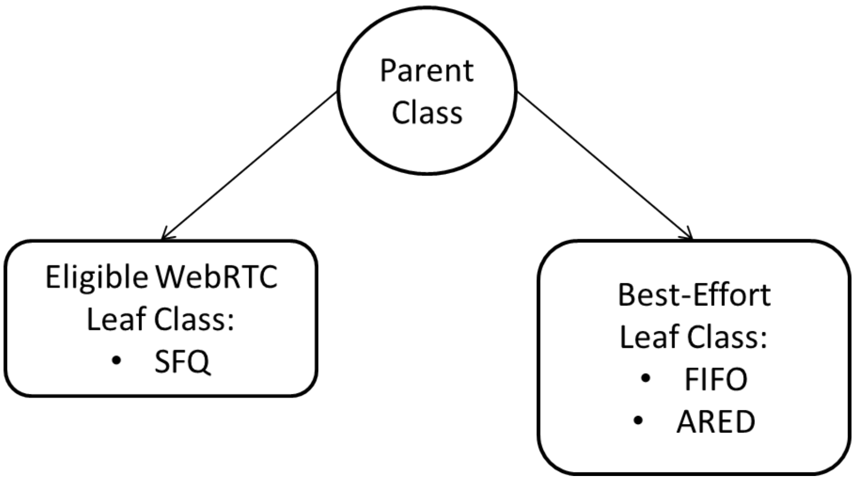


Figure 6.7 – HTB classes and their queuing disciplines

WebRTC class configuration

For WebRTC class, we use SFQ[14]. As explained in the previous chapter (section 5.4.1), it divides traffic into FIFO queues, while assuring the fairness between flows and in case of multiple WebRTC flows, all of them can receive the same treatment.

SFQ has a large number of parameters to configure. We have left most of them by default, but we have modified the length of the queue, limit, to a value that we consider more adapted to WebRTC traffic. We also modified the perturb value.

To choose the queue length, we have based our decision on existing studies. Evaluations of GCC indicated that it performs well for one-way latencies lower than 200ms (**Singh et al. 2013**). Thus, we assumed that a maximum acceptable queuing delay for WebRTC is 200ms. We obtained a buffer size of 25kB, since it will take 200ms to empty up this buffer at the rate of 1Mb/s.

Since SFQ requires to indicate limit in packets, we measured with Wireshark that an average packet size of WebRTC communication (audio and video) is 610B. Thus, a queue length of 25kB corresponds to a queue length of 40 packets.

Perturb is by default set to 0. However, it is not recommended since it assumes that the hashing algorithm is never run, as it was explained in the section 5.4.1. Hence, we use a suggested value of 10s (**Hubert nown**).

Best-effort class configuration

The best-effort class is used for any traffic other than eligible WebRTC traffic, notably the concurrent TCP traffic. For this class, we have chosen two queuing disciplines.

First, we have evaluated FIFO, i.e. a default queuing discipline in HTB. By default its limit is set to 1000 packets in our Linux distribution. However, apart from the default value, we tested FIFO queuing discipline of 125kB, so quite a large buffer.

[14]http://manpages.ubuntu.com/manpages/trusty/man8/tc-sfq.8.html

Second, we have evaluated ARED[15] to verify how the best-effort traffic behaves in the presence of mechanisms different than FIFO. To configure ARED, Linux TC RED queuing discipline is used with activated adaptive option.

As mentioned before, according to Netalyzr, (**Kreibich et al. 2010**), buffer sizes can vary between 4kB and 500kB. For the evaluation of ARED we have decided to set an average queue to a value that is not too extreme, i.e. a compromise between a very large and a very short buffer.

We wanted a buffer size analogical to SFQ, so a buffer of about 40 packets. We assumed that in the best-effort class, the maximum size of packets is 1514, i.e. 1500 Ethernet MTU along with the hardware header length. We obtained the value of 60,560kB. This value is used as the average queue and is used to calculate ARED parameters, e.g. thresholds min, max and limit, according to TC suggestions.

We have also set harddrop option, since we want packets to be dropped instead of using ECN marking.

Configuration summary

The summary of evaluated configurations for rate and queue length based classful queuing mechanisms is given in the table 6.4. Given the large number of parameters and for better visibility, only essential parameters are presented.

6.4 Fiber networks - PoC configuration

6.4.1 Fiber emulation settings

We have done some additional tests for emulation of fiber access networks.

According to ARCEP, in the second quarter of 2016 in France there were 1.765 million end-to-end fiber subscriptions out of 27.220 million subscriptions. However, this number is still increasing, and so far represents 27% of all households eligible to FttH.

Here we also used netem for network emulation purposes. Based on existing ARCEP quality measurements of FttH wired networks, we have chosen the most common configuration and we have set the uplink to 200Mbit/s and we configure the delay to 10ms (**Arcep 2015**).

6.4.2 WebRTC call

For fiber we used the sames settings for WebRTC call as we did for ADSL networks. Hence, a 4CIF video of 120s is injected into Chrome browser and used a webcam and microphone flows.

6.4.3 Concurrent traffic

For fiber network emulation, it is more difficult to create congestion because of resources oversizing. We could not use a simple file upload, like for ADSL. As mentioned in section 5.1.2, TCP congestion control probes for network resources and in the Slow Start phase, it doubles the congestion window every RTT and it takes time to fill the link. Hence, in case of large link, the upload would finish before the TCP acquired all of the available

[15]http://manpages.ubuntu.com/manpages/wily/man8/tc-red.8.html

CPE config name	TC qdisc	Parameters settings

HTB204 fifo def	WebRTC -> sfq	limit=40 packets
		perturb=10

	Best-Effort -> fifo	limit=1000 packets (default)

HTB654 fifo def	WebRTC -> sfq	limit=40 packets
		perturb=10

	Best-Effort -> fifo	limit=1000 packets (default)

HTB204 fifo 125kB	WebRTC -> sfq	limit=40 packets
		perturb=10

	Best-Effort -> fifo	limit=125kB

HTB654 fifo 125kB	WebRTC -> sfq	limit=40 packets
		perturb=10

	Best-Effort -> fifo	limit=125kB

HTB204 ared	WebRTC -> sfq	limit=40 packets
		perturb=10

	Best-Effort -> adaptive red	limit=726720B
		min=60560B
		max=181680B
		hardrop on

HTB654 ared	WebRTC -> sfq	limit=40 packets
		perturb=10

	Best-Effort -> adaptive red	limit=726720B
		min=60560B
		max=181680B
		hardrop on

Table 6.4 – Rate and queue length based classful configuration

capacity. In brief, TCP is limited by the size of the congestion window and also by the size of the receiver window.

Thus, instead of using simple file upload, we have used the iperf traffic generator. With iperf we can generate a large number of parallel TCP streams for a defined period of time, that together can more efficiently fill out the available capacity. For the test purposes we generated 1, 5 and 100 of parallel TCP streams.

6.4.4 Configuration for droptail queuing mechanisms

We have tested a droptail configuration. However, for fiber we have selected only one buffer size, i.e. 16kB, as it corresponds to the existing practices of network operators.

CPE config name	TC qdisc	Parameters settings
Droptail Fiber	fifo	limit=16kB

HTB Fiber	WebRTC -> sfq	limit=40 packets
		perturb=10

	Best-Effort -> fifo	limit=16kB

FQ Codel	fq_codel	limit=10240 packets (default)
		flows=1024 (default)
		target=5ms (default)
		interval=100ms (default)
		quantum=1514 (default)
		noecn (ecn turned off)

PIE	pie	limit=1000 packets (default)
		target=20ms (default)
		tupdate=30ms (default)
		noecn (ecn turned off)
		bytemode on

Table 6.5 – Fiber configuration

6.4.5 Configuration for target delays based queuing mechanisms

We have tested two AQM mechanisms, PIE and FQ Codel. We have also reused the ADSL configuration.

6.4.6 Configuration for rate and queue length based classful queuing mechanisms

The current WebRTC video flow bitrate, when not constrained by network bandwidth, is typically around 2Mbit/s. It is a small fraction of available fiber bandwidth, i.e. 200Mbit/s. Hence, given TCP's inefficiency and dependence on RTT, it is difficult to create congestion big enough to perturb WebRTC traffic.

Nevertheless, we have tested one configuration in which we provided a given rate to WebRTC traffic.

We have used the rate and queue length based classful configuration and we have configured the guaranteed rate of WebRTC to 2,5Mbit/s, which is more than enough to cover bandwidth requirements of WebRTC. WebRTC traffic has also higher priority than the best-effort traffic. For best-effort traffic, we decided to use the default FIFO queuing discipline.

6.4.7 Configuration summary

The summary of evaluated configurations for fiber is given in the table 6.5.

6.5 Summary

In this chapter we have given an overview of the implemented PoC and the chosen test environment.

We have given a detailed description of the testbed architecture. We have also presented the chosen KPIs for further evaluation.

We have discussed the PoC configurations of identified traffic management solutions used for ADSL and fiber evaluation.

The advantage of the proposed test environment is that it allows collecting various measurements essential for a detailed analysis of evolution of the WebRTC communication.

We decided to collect KPIs from different levels, i.e. we provided network and application layer measurements enriched by subjective opinion. We also introduced a collection of KPIs that are strongly linked with quality, e.g. frame rate and frame width. It is possible to collect these metrics, because WebRTC audio and video engines can adapt media stream bitrate to underlying network link between the WebRTC users. Consequently, the application can update the media constraints, notably the frame rate and video resolution (**Grigorik 2013**).

Chapter 7

Evaluation of proposed traffic management solutions

Contents

7.1 Evaluation methodology

In this section, we give an overview of our evaluation methodology. The detailed test descriptions and results assessment will be given in the sections 7.2 and 7.3.

The aim of this evaluation is to create an assessment of the impact of different traffic management approaches on WebRTC and its congestion control algorithm, i.e. GCC in our case.

This assessment is necessary to verify current best practices in network configuration used in the best-effort delivery. It is also essential in verifying the pertinence of proposed traffic management adapted to WebRTC.

The obtained results are used for the identification of possible improvements and the definition of recommendations.

We evaluated three traffic management approaches:

- Droptail approach, i.e. a configuration based on the current IP network engineering,
- Target delay based approach, i.e. a configuration aiming at assuring lower queuing delays regardless the traffic,
- Rate and queue length based classful approach, i.e. a configuration aiming at isolating the sensitive traffic.

The table 7.1 gives a summary of sections covering queuing mechanisms overview and corresponding PoC configurations of identified traffic management approaches chosen for evaluation.

Traffic management	**Queuing mechanisms overview**	**PoC Configuration**	
		ADSL	**Fiber**
Droptail	Section 5.1.1	Section 6.3.4	Section 6.4.4
Target delay based	Section 5.5.1	Section 6.3.5	Section 6.4.5
Rate and queue length based classful	Section 5.5.2	Section 6.3.6	Section 6.4.6

Table 7.1 – Summary of sections covering queuing mechanisms overview and corresponding PoC configurations

The chosen use cases concern uplink wireline access networks, ADSL and fiber. We analyse WebRTC traffic by itself, but also in the presence of long-lived TCP flows.

KPIs used for analysis were explained in the section 6.2 and their overview is given in the table 7.2.

We mainly focus on video KPIs since GCC mechanism is only implemented for video flows.

Measurement name	Measurement source	Measurement type
Video bitrate	getStats	Application layer
Video frame rate	getStats	Application layer
Video frame width	getStats	Application layer
Video packet loss	getStats	Application layer
Video RTT	getStats	Application layer
Upload time	Upload application	Application layer
Channel utilisation	Wireshark	Network layer

Table 7.2 – Overview of KPIs used for evaluation

7.1.1 Test evaluation methodology for 1Mbit/s ADSL uplink

For a 1Mbit/s ADSL uplink we evaluate an important number of various queuing mechanisms. We start by an evaluation of all test results per traffic management configuration. Later, we choose the most representative test results of each configuration and we create an overall comparison.

Assessment per configuration

For each of the traffic management approaches, we follow the same procedure.

First, we analyse the WebRTC traffic by itself, i.e. without any concurrent traffic.
We compare the obtained results with the reference measurements of WebRTC traffic for a generic configuration and without any concurrent traffic. Furthermore, for the two identified approaches, i.e. the target delay based and the rate and queue length based classful approach, we compare them with a typical droptail configuration.

We analyse application layer KPIs for the WebRTC traffic.

- We observe the evolution of the video bitrate in time. We compare the obtained bitrate with the reference.
- We observe the evolution of the video frame rate in time. We compare the obtained frame rate with the reference.
- We observe the evolution of the video frame width in time. We compare the obtained frame width with the reference.
- We observe the overall RTT measurements. We compare the obtained RTT with the reference.
- We observe the overall packet loss measurements. We compare the obtained packet loss with the reference.

The table 7.3 gives a summary of figures used for evaluation of WebRTC traffic without any concurrent traffic.
We compare the obtained KPI values with the reference and we identify any differences or anomalies. We provide our hypothesis to explain them.

KPI	**Droptail**	**Target delay based**	**Rate and queue length based classful**
Video bitrate	Figure 7.1a	Figure 7.7a	Figure 7.13a
Video frame rate/width	Figure 7.1b	Figure 7.7b	Figure 7.13b
Video RTT	Figure 7.2a	Figure 7.8a	Figure 7.14a
Video packet loss	Figure 7.2b	Figure 7.8b	Figure 7.14b

Table 7.3 – List of figures for each KPI per configuration - ADSL, WebRTC traffic without any concurrent traffic

Second, we analyse the WebRTC traffic in the presence of TCP concurrent traffic.
We compare the obtained results with the reference measurements of WebRTC and TCP traffic for a generic configuration and without any perturbations. Moreover, for the two identified approaches, i.e. target delay based and rate and queue length based classful approach, we compare them with a typical droptail configuration.

We analyse the application layer KPIs for WebRTC traffic in the same manner as it was done for WebRTC communication without any concurrent traffic. We compare the obtained KPI values with the reference and we identify any differences or anomalies. We provide our hypothesis to explain them.

Later, we analyse the impact of different traffic management configurations on the concurrent traffic, i.e. TCP file upload.

- We analyse the application layer KPI, i.e. we analyse the upload time. We compare the obtained results with the reference.
- We analyse the network layer KPIs, i.e. we analyse the channel utilization of each traffic in order to observe how the resources are shared between TCP and WebRTC flows.

The table 7.4 gives a summary of figures used for evaluation of WebRTC traffic in the presence of TCP concurrent traffic.

KPI	**Droptail**	**Target delay based**	**Rate and queue length based classful**
Video bitrate	Figure 7.3a	Figure 7.9a	Figure 7.15a
Video frame rate/width	Figure 7.3b	Figure 7.9b	Figure 7.15b
Video RTT	Figure 7.4a	Figure 7.10a	Figure 7.16a
Video packet loss	Figure 7.4b	Figure 7.10b	Figure 7.16b
Upload time	Figure 7.5	Figure 7.11	Figure 7.17
Channel utilisation	Figure 7.6	Figure 7.12	Figure 7.18

Table 7.4 – List of figures for each KPI per configuration - ADSL, WebRTC traffic in the presence of TCP concurrent traffic

Comparison of the main results for WebRTC communication in the presence of TCP concurrent traffic

In order to simplify the final assessment for a 1Mbit/s ADSL uplink, we compare the best performance obtained from two identified approaches, i.e. target delay based approach and rate and queue length based classful approach. We compare them with a typical droptail configuration and with the reference measurements of WebRTC and TCP traffic for a generic configuration and without any perturbations.

First, we analyse the application layer KPIs for WebRTC traffic. Second, we analyse the impact of different configurations on the concurrent traffic, i.e. TCP file upload. Finally, we present the subjective opinion about the perceived quality of WebRTC communication.

The table 7.5 gives a summary of figures used for comparison of principal results for WebRTC communication in the presence of TCP concurrent traffic.

KPI	**Comparison**
Video bitrate	Figure 7.19a
Video frame rate/width	Figure 7.19b
Video RTT	Figure 7.20a
Video packet loss	Figure 7.20b
Upload time	Figure 7.21
Channel utilisation	Figure 7.22

Table 7.5 – List of figures for each KPI for all configuration - ADSL, WebRTC traffic in the presence of concurrent traffic

7.1.2 Test evaluation methodology for 200Mbit/s fiber uplink

For a 200Mbit/s fiber uplink we evaluate a smaller number of various queuing mechanisms than in case of ADSL. Hence, we can proceed directly to comparison of different traffic management approaches.

We do not perform the analysis of WebRTC without any concurrent traffic, since the link is oversized and without any concurrent traffic it does not have any impact on WebRTC.

We also do not analyse any impact on the concurrent traffic. First of all, our application layer measurements do not apply to TCP traffic used in fiber evaluation, where we used `iperf` to generate traffic instead of TCP file upload, as it was explained in the section 6.4.3.

Moreover, channel utilization graphs would be illegible since the link is oversized and WebRTC traffic uses a very small fraction of the available link.

Thus, we directly proceed to the analysis of application layer KPIs for WebRTC traffic in the presence of long-lived TCP concurrent traffic.

- We observe the evolution of the video bitrate in time. We compare the obtained bitrate with the reference.
- We observe the evolution of the video frame rate in time. We compare the obtained frame rate with the reference.
- We observe the evolution of the video frame width in time. We compare the obtained frame width with the reference.
- We observe the overall RTT measurements. We compare the obtained RTT with the reference.
- We observe the overall packet loss measurements. We compare the obtained packet loss with the reference.

The table 7.6 gives a summary of figures used for comparison of different configurations based on results for WebRTC communication in the presence of TCP concurrent traffic.

KPI	**Comparison**
Video bitrate	Figure 7.23a
Video frame rate/width	Figure 7.23b
Video RTT	Figure 7.24a
Video packet loss	Figure 7.24b

Table 7.6 – List of figures for each KPI for all configuration - Fiber, WebRTC traffic in the presence of concurrent traffic

We compare all measurements and identify any differences and anomalies. We provide our hypothesis to explain them.

Finally, we present the subjective opinion about the perceived quality of WebRTC communication.

7.2 Results assessment for 1Mbit/s ADSL uplink

7.2.1 Test description

First, we run have the ADSL test cases. We started by the evaluation of the droptail configuration. Later, we tested the target delay based queuing mechanisms and rate and queue length based classful queuing mechanisms. We compared the results obtained with the ones assessed for droptail configuration.

Each test case corresponds to a 120 second WebRTC communication. Each test case is repeated 5 times for every CPE configuration. Since the tests were attended, running too many tests would be time consuming. In order to have enough data, we decided that 5 tests were sufficient. There are two types of test cases:

- Test case name that ends with com - Only WebRTC calls, 120s long WebRTC call without any concurrent traffic.
- Test case name that ends with ct - WebRTC call along with a concurrent traffic that is launched 30s after WebRTC call establishment.

Additionally, we have run two reference measurements: one for WebRTC traffic without any concurrent traffic and another one for file upload concurrent traffic. They are run in Droptail 1000p configuration, that corresponds to the default Linux value for FIFO (see the table 6.2).

- RefWebRTC is a 120s long WebRTC call without any concurrent traffic in Droptail 1000p configuration (that corresponds to 610kB buffer, given the fact that average WebRTC packet size is 610B).
- RefUpload is a 5MB file upload, so a TCP traffic without any concurrent or WebRTC traffic in Droptail 1000p configuration (that corresponds to 1500kB buffer).

The measurements used for evaluation are given in the table 7.7. The details about collecting these statistics were given in the section 6.2.

Measurement name	**Measurement source**	**Measurement type**
Video bitrate	getStats	Application layer
Video frame rate	getStats	Application layer
Video frame width	getStats	Application layer
Video packet loss	getStats	Application layer
Video RTT	getStats	Application layer
Upload time	Upload application	Application layer
Channel utilisation	Wireshark	Network layer

Table 7.7 – Measurements used for ADSL evaluation

We have assessed application layer measurements for the video, since the GCC algorithm concerns only video and not audio. Nevertheless, network layer measurements concern both, audio and video, since we used the overall bitrate captured by Wireshark in order to calculate channel utilization. In addition, subjective opinion is used to enrich the results evaluation.

7.2.2 Evaluation of droptail configuration

In this section we present the evaluation of the droptail configuration, i.e. a configuration based on the current IP network engineering, as discussed in the section 5.1.1. The summary of the test cases used for evaluation of droptail configuration is given in table 7.8.

The droptail test cases are compared with the reference measurements, i.e. RefWebRTC and RefUpload.

Test case name	CPE config	Concurrent Traffic
RefWebRTC (reference)	Droptail 1000p	none
RefUpload (reference)	Droptail 1000p	none
DT16kB com	Droptail 16kB	none
DT32kB com	Droptail 32kB	none
DT64kB com	Droptail 64kB	none
DT125kB com	Droptail 125kB	none
DT250kB com	Droptail 250kB	none
DT16kB ct	Droptail 16kB	TCP file upload
DT32kB ct	Droptail 32kB	TCP file upload
DT64kB ct	Droptail 64kB	TCP file upload
DT125kB ct	Droptail 125kB	TCP file upload
DT250kB ct	Droptail 250kB	TCP file upload
DTdefault ct	Droptail 1000p	TCP file upload

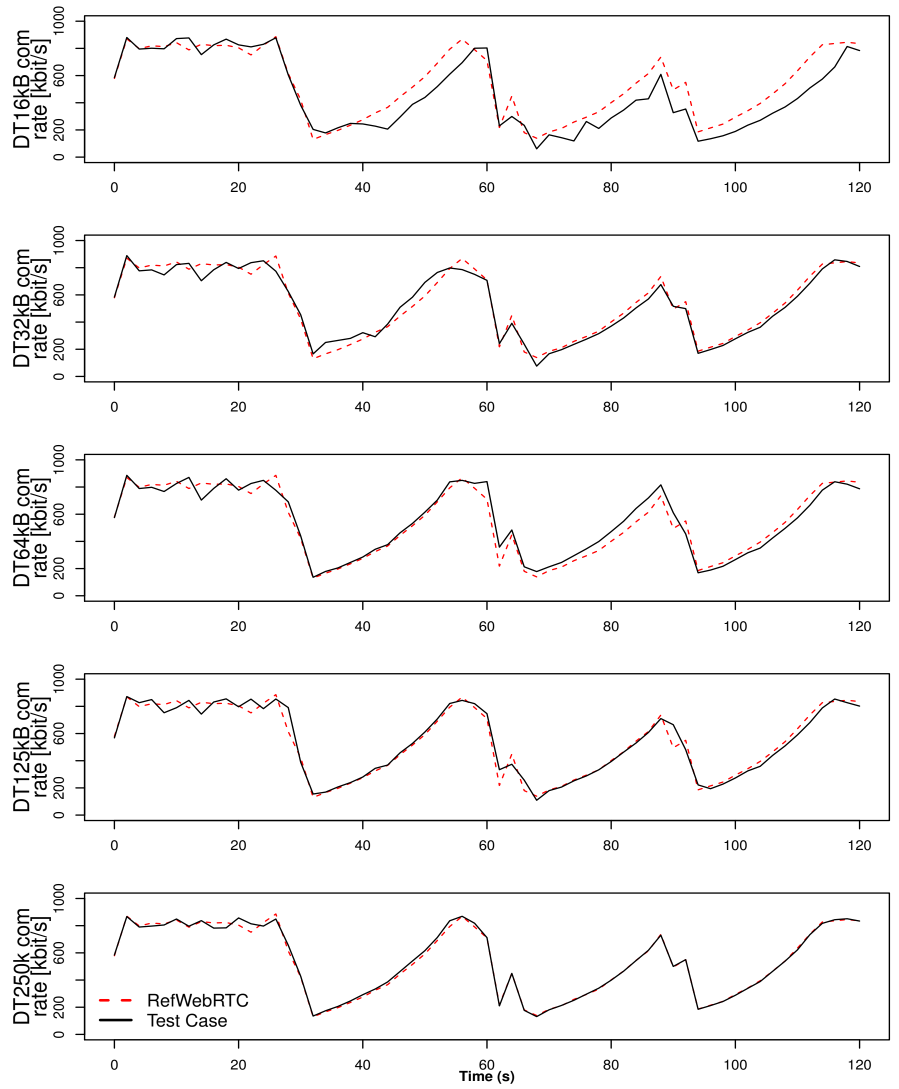
Table 7.8 – Test matrix ADSL uplink droptail configuration

Impact of droptail configuration on WebRTC traffic without any concurrent traffic

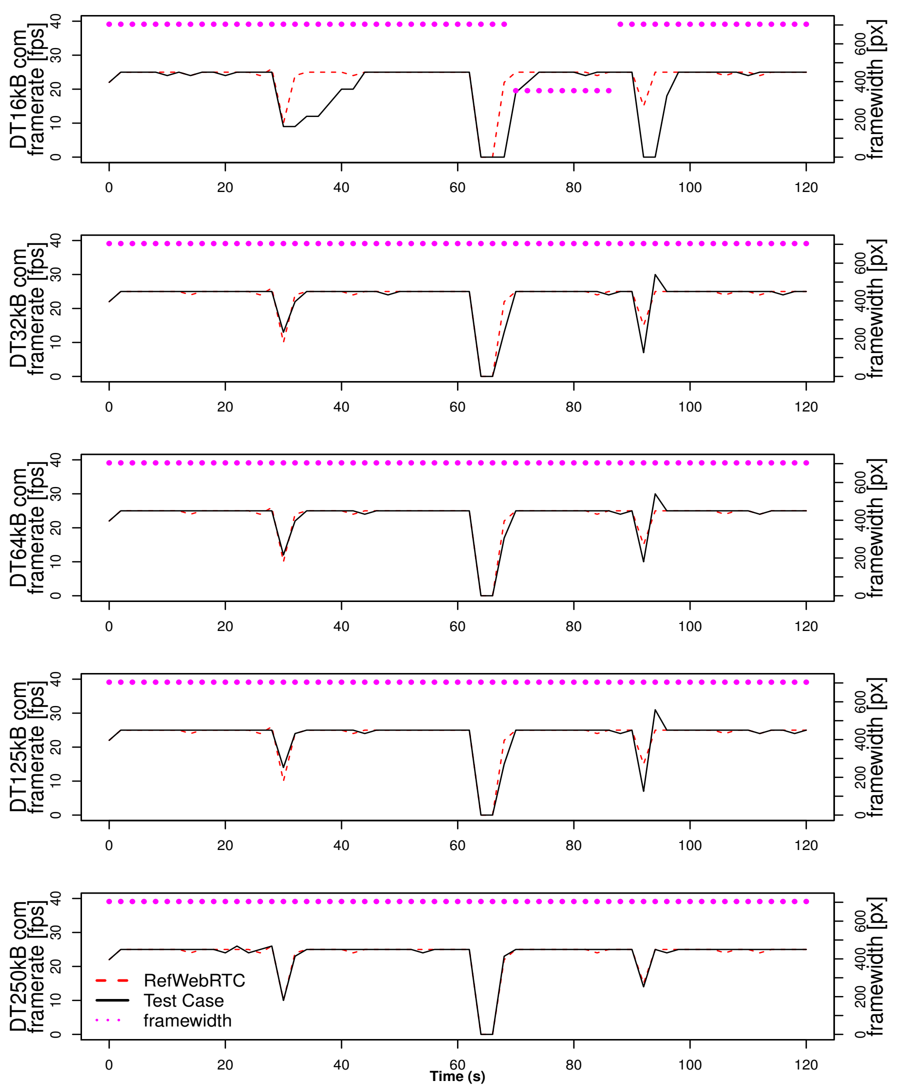
Figures 7.1 and 7.2 represent a set of schemas allowing to evaluate the impact of droptail configuration on the WebRTC communication without any concurrent traffic.

Figure 7.1a shows video bitrate, and figure 7.1b shows the frame rate and frame width. For better readability, only one test case per configuration is given. Each test case is superposed with the reference RefWebRTC.

Figure 7.2 shows the measurements of RTT and packet loss per configuration, but this time for all test cases, including the reference RefWebRTC.

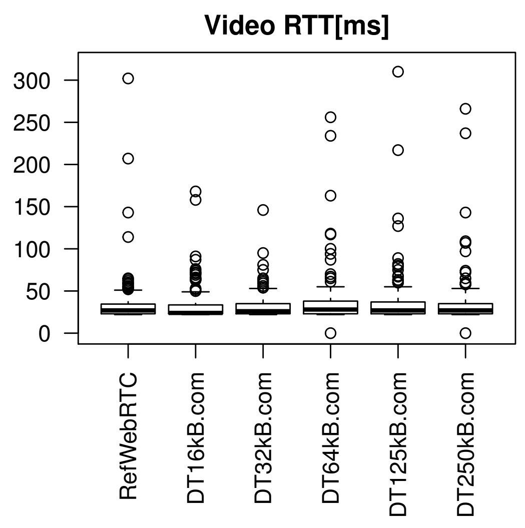


(a) Video bitrate for one test case per configuration superposed with RefWebRTC.

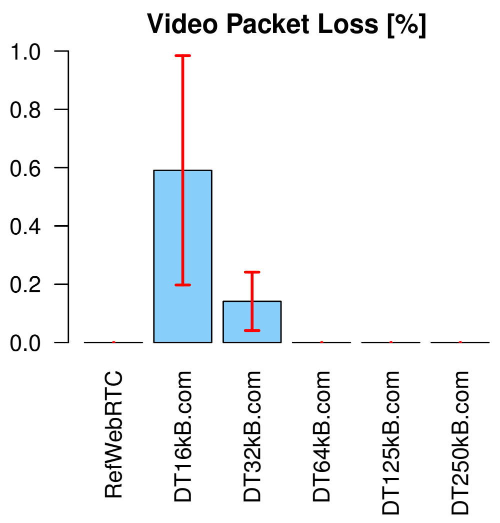


(b) Video frame rate (left axis) and frame width (right axis) for one test case per configuration superposed with RefWebRTC.

Figure 7.1 – Impact of droptail configuration on WebRTC traffic (bitrate, frame rate and frame width) without any concurrent traffic



(a) Video RTT - Boxplot of all test cases per configuration



(b) Video packet loss - Mean and standard deviation of all test cases per configuration

Figure 7.2 – Impact of droptail configuration on RTT and packet loss of WebRTC traffic without any concurrent traffic

For most of the configurations, the bitrate and frame rate are very close to the reference one, see figure 7.1. However, in the case of the shortest buffer, test case DT16kB com, the bitrate and frame rate differ from RefWebRTC. It is also the only configuration for which the frame width becomes smaller.

Our hypothesis is that the 16kB queue, test case DT16kB ct, is very short, so it does not introduce much of a queuing delay variation. Hence, the queuing delay is kept relatively short, so GCC does not detect when the queue builds up or drains, and consequently, it does not detect the overuse of the link. The overuse is detected when packets are already lost and the loss-based controller takes over.

Given these points, in case of 16kB, the delay-based controller is not sensitive enough. As a result, the WebRTC flow does not estimate the available link capacity with enough granularity which leads to filling up the buffer and consequently to packet loss, which has an impact on the quality.

We believe it is also the reason why we observe more packet loss for shorter buffers, i.e. 16kB and 32kB.

As seen on the figure 7.2a, the RTT does not change between the configurations, but it can be also seen that for the shortest buffers, it never exceeds 200ms.

Impact of droptail configuration on WebRTC traffic with TCP concurrent traffic

Figures 7.3 and 7.4 represent a set of schemas allowing to evaluate the impact of a droptail configuration on the WebRTC communication in the presence of a TCP concurrent traffic.

Figure 7.3a shows video bitrate, and figure 7.3b shows the frame rate and frame width for a chosen test case superposed with RefWebRTC.

Figure 7.4 shows the measurements of RTT and packet loss per configuration for all test cases, including the reference RefWebRTC.

Figure 7.5 represents the impact of droptail configuration on the concurrent traffic, i.e. the upload time of a file over TCP.

Figure 7.6 shows channel utilization of two observed traffics, i.e. WebRTC and TCP.

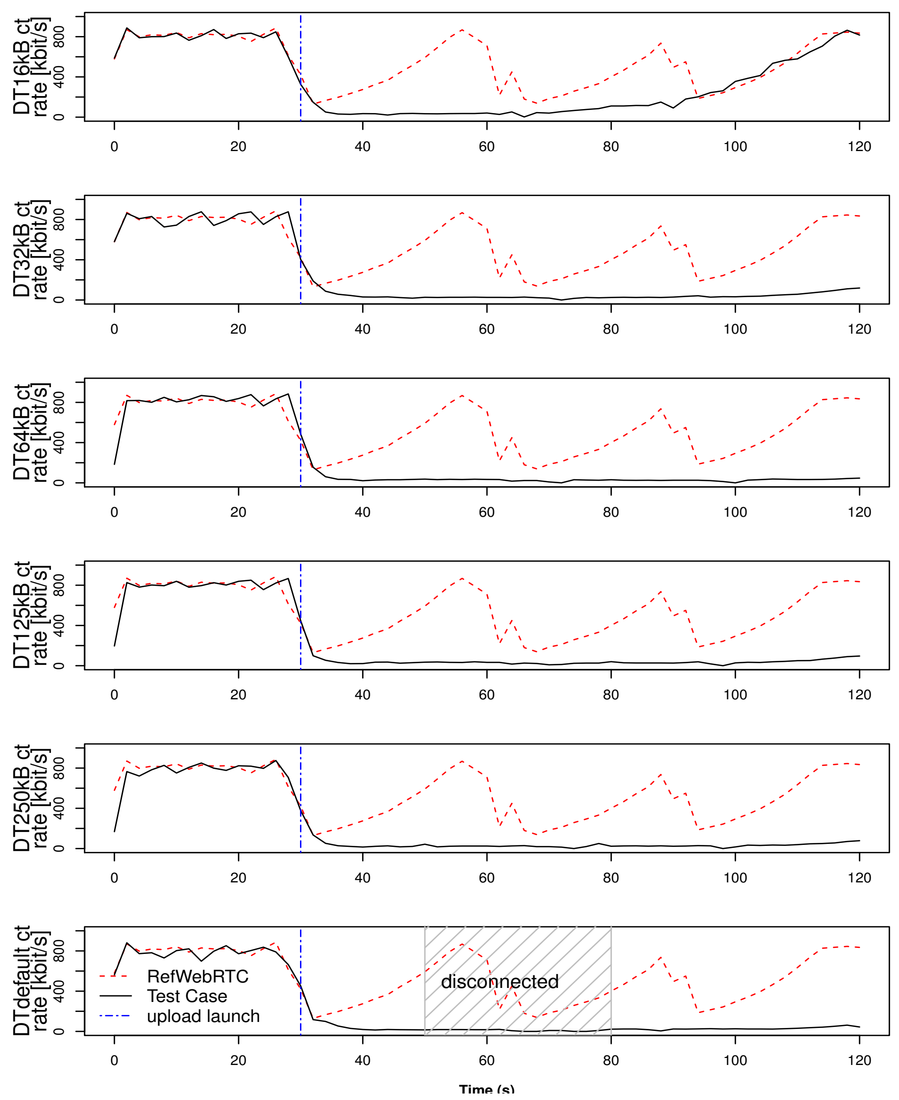
It can be observed on figure 7.3a that when a concurrent traffic is launched, the WebRTC bitrate drops to very low values, so it is starved by TCP traffic. For the shortest buffer, test case DT16kB ct, after the upload is finished (about 50s after the upload launch, since it corresponds to a general upload time, as seen on the figure 7.5), the bitrate goes back to normal. However, it is not the case for the longer buffers. In case of long buffers, even after the upload is finished, WebRTC bitrate stays very low. The worst case is observed for the longest buffer, test case DTdefault ct: the delay is so excessive that it causes a disconnection of WebRTC traffic.

The frame rate is heavily impacted as seen on the figure 7.3b. There are a lot of video freezes since the frame rate is often close to zero. Additionally, the frame width becomes very small. Again, for the shortest buffer, the video goes back to normal after the concurrent traffic is finished, whereas for longer buffers, it needs more time to recover.

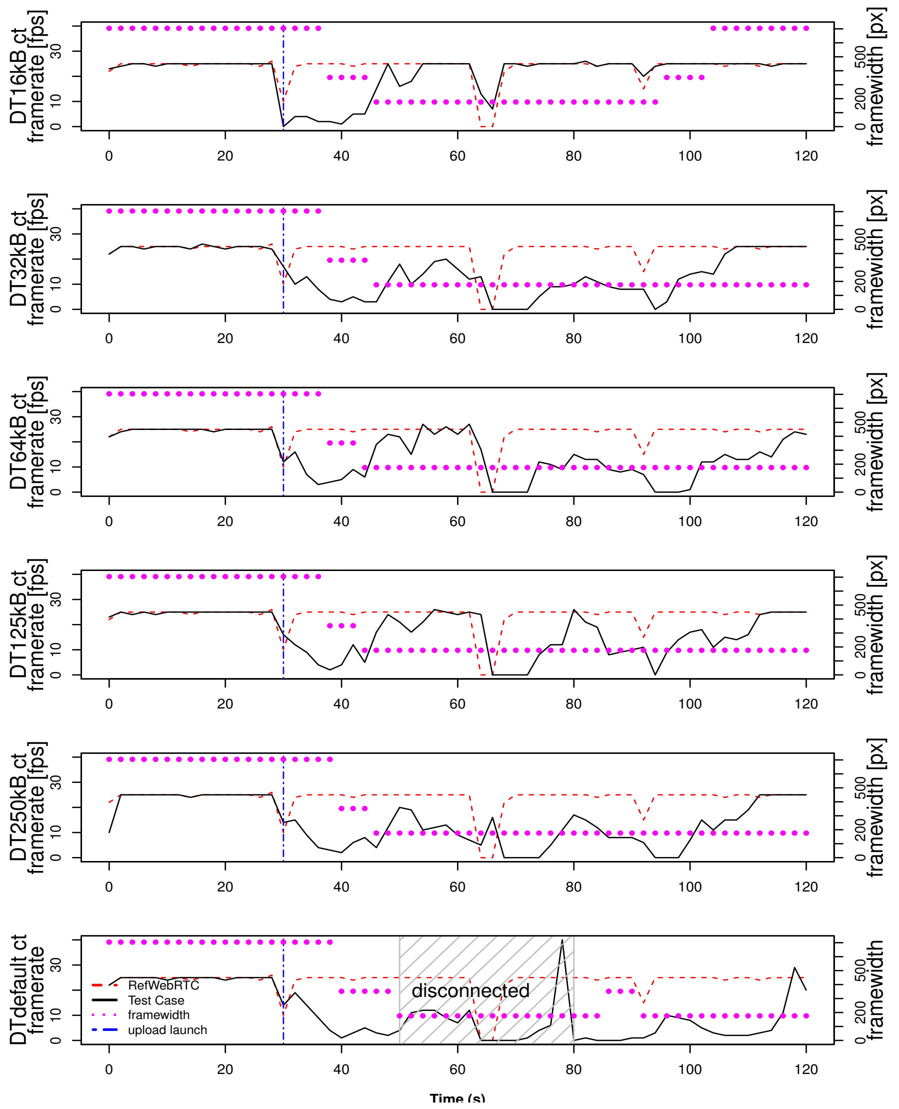
In case of the shortest buffer, test case DT16kB ct, the bitrate and frame rate degradation is the lowest. In our opinion, it is linked with the buffer size. Shorter buffer introduces less queuing delay, but more packet loss. It can also have an impact on the TCP flow, which is sensible to packet loss that triggers a decrease in its sending rate. As it can be seen on the figure 7.5, when the shortest buffer is used, the upload time is slightly longer.

Concerning the RTT, figure 7.4a, the longer the buffers, the higher RTT is obtained. However, the packet loss is higher for shorter buffers and kept low when longer buffers are used. In our opinion, this is caused by lack of the reactivity of delay-based controller in case of very short buffers, as was discussed in the previous section.

It is important to highlight that excessive delays are observed for the test case DT125kB ct and the test case DT250kB ct. The values for the test case DTdefault ct are not shown since they are too high, because of the disconnection.

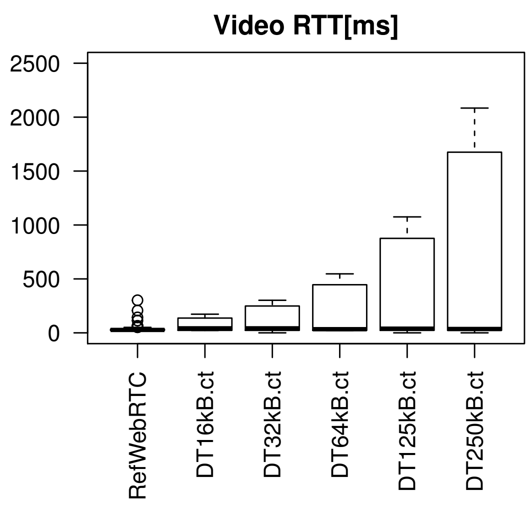


(a) Video bitrate for one test per configuration superposed with RefWebRTC.

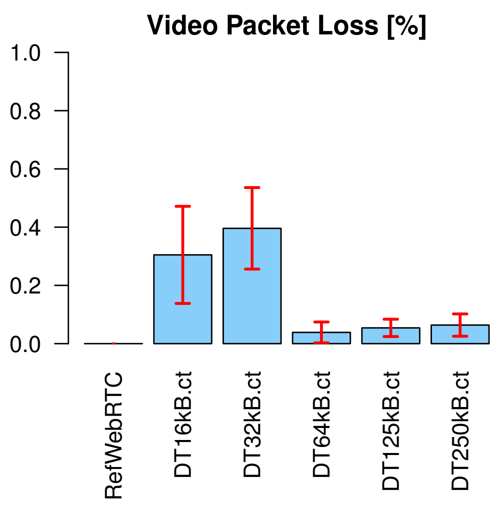


(b) Video frame rate (left axis) and frame width (right axis) for one test per configuration superposed with RefWebRTC.

Figure 7.3 – Impact of droptail configuration on WebRTC traffic (bitrate, frame rate and frame width) with TCP concurrent traffic



(a) Video RTT - Boxplot of all test cases per configuration



(b) Video packet loss - Mean and standard deviation of all test cases per configuration

Figure 7.4 – Impact of droptail configuration on RTT and packet loss of WebRTC traffic with TCP concurrent traffic

In this context, it is important to analyse the amount of time the WebRTC traffic takes to recover from starvation. It concerns both the bitrate and the frame rate along with the frame width.

In our hypothesis, it is linked with the reaction time of delay-based controller. As it could be seen, when the upload was launched and the delays increased, the GCC reacted rapidly and the bitrate and frame rate decreased to very low values. When the upload was finished, WebRTC could go back to the initial rate. The amount of time WebRTC took to recover is linked with the GCC design. The GCC does not increase the sending rate immediately after the congestion is finished, i.e. when the underuse signal is generated. Instead it goes into the *hold* state and waits for buffers to stabilize. It goes into the *increase* state, when the estimate of the one way delay gradient ($m(t_i)$) is close to 0 and a normal signal is triggered .

Moreover, the estimate ($m(t_i)$) is calculated for every i-th video frame. In test case DT16kB ct, it can be seen that the frame rate is higher and close to the reference frame rate, so frames are received more frequently, than in the case of large buffers. Hence, the delay-based controller can react faster.

Regarding the interaction with the concurrent traffic, the difference of buffer sizes has less impact on the concurrent traffic as seen on the figure 7.5. The upload takes slightly more time than when there is no concurrent traffic, test case RefUpload, but the upload time does not vary much between different configurations. Similarly, according to the figure 7.6, channel utilization stays similar between different configurations and in all cases the TCP traffic takes the large part of the link.

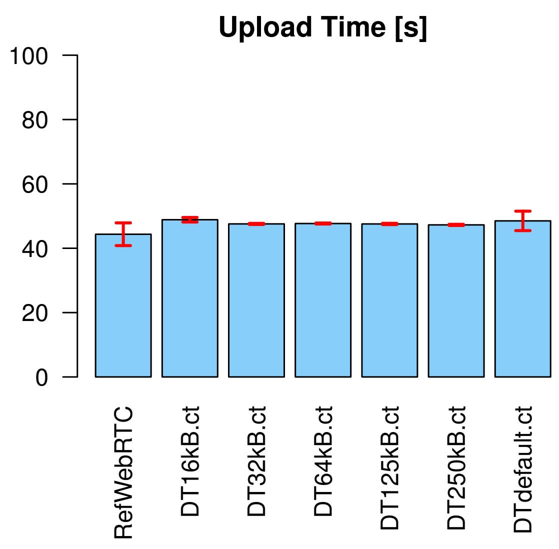


Figure 7.5 – Impact of droptail configuration on concurrent traffic upload time - Mean and standard deviation of all test cases per configuration

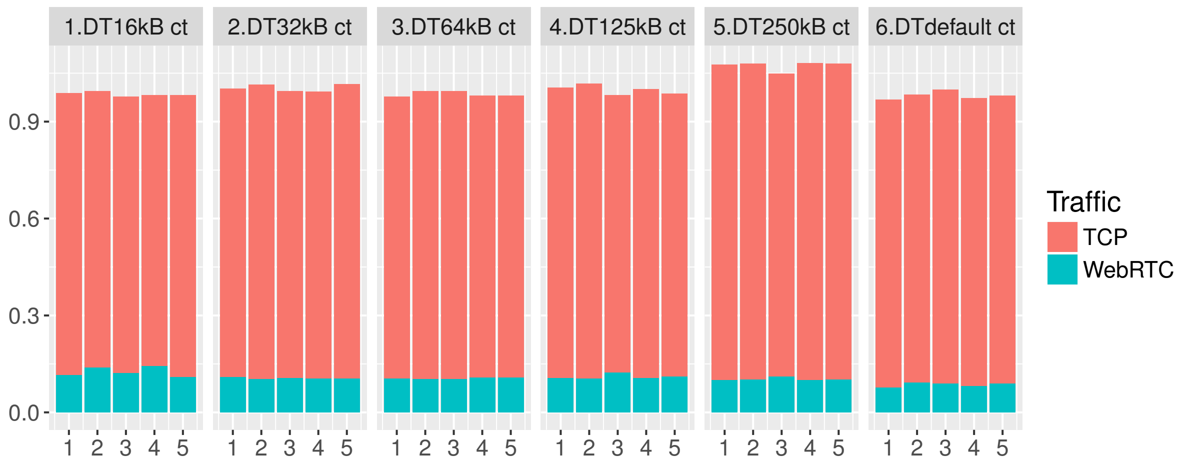


Figure 7.6 – Impact of droptail configuration on channel utilization of all test cases per configuration

Summary of the impact of the droptail configuration on WebRTC traffic

The evaluated test cases prove that current practices in buffer configurations are not adapted to the WebRTC traffic and privilege TCP traffic. WebRTC is starved in the presence of concurrent traffic. The implemented mechanisms, e.g. GCC, are not able to compensate poor network conditions and in most cases take time to recover from the negative impact of congestion.

Furthermore, it is important to choose an appropriate buffer size. Too large buffers can introduce excessive delays. Shorter buffers introduce less delay and in case of starvation, they enable faster WebRTC traffic recovery time. On the other hand, when buffers are too short, GCC and its delay-based controller are less sensitive and can lead to the introduction of packet loss. Hence, a balance when choosing buffer size needs to be achieved.

7.2.3 Evaluation of the target delay based configuration

In this section we present the evaluation of the target delay based configuration, i.e. a configuration using the AQM mechanisms. The summary of the test cases used for the evaluation of the target delay based configuration is given in the table 7.9.

The target delay test cases are compared with the reference measurements, i.e. RefWebRTC and RefUpload. They are also compared with measurements performed on a droptail configuration, test case DT125kB, with a buffer size of 125kB, test case DT125kB com and test case DT125kB ct, that represents a general trend of wide spread droptail approach, as discussed in the section 5.1.1.

Test case name	**CPE config**	**Concurrent Traffic**
RefWebRTC (reference)	Droptail default	none
DT125kB com (reference)	Droptail 125kB	none
DT125kB ct (reference)	Droptail 125kB	TCP file upload
FQ Codel com	FQ Codel	none
PIE com	PIE	none
FQ Codel ct	FQ Codel	TCP file upload
PIE ct	PIE	TCP file upload

Table 7.9 – Test matrix for ADSL uplink target delay based configuration

Impact of target delay based configuration on WebRTC traffic without any concurrent traffic

Figures 7.7 and 7.8 represent a set of schemas allowing to evaluate the impact of target delay based configurations on the WebRTC communication without any concurrent traffic.

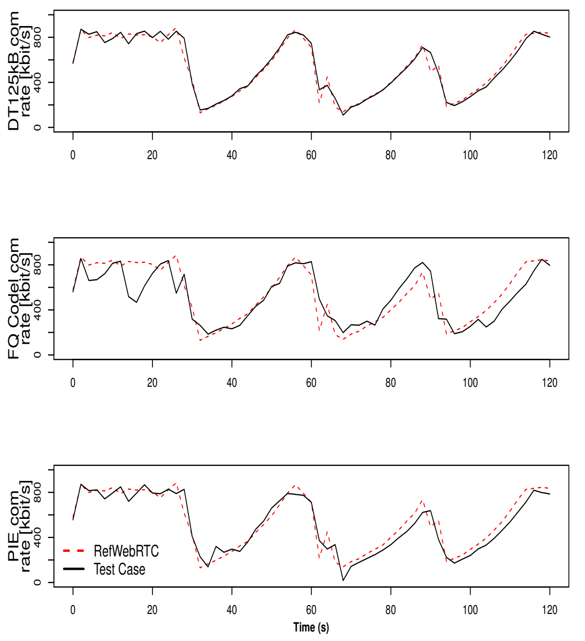
Figure 7.7a shows video bitrate, and figure 7.7b shows the frame rate and frame width. For better readability, only one test case per configuration is given. Each test case is superposed with the reference RefWebRTC.

Figure 7.8 shows the measurements of RTT and packet loss per configuration, but this time for all test cases, including the reference RefWebRTC.

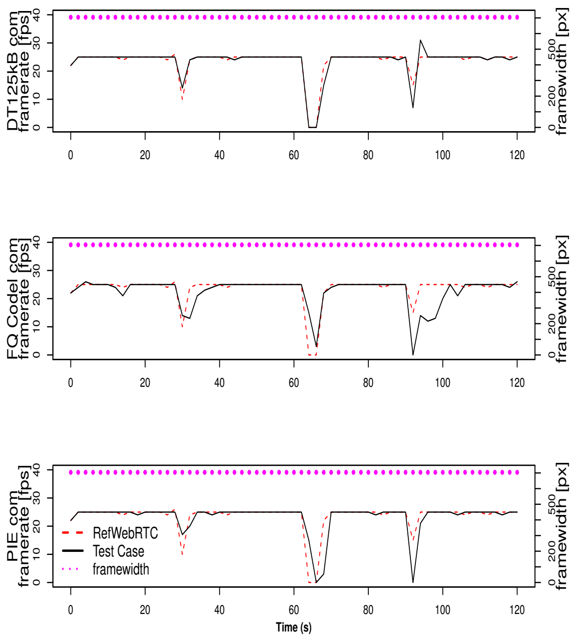
Additionally, on all figures a droptail configuration, test case DT125kB ct, is presented for comparison.

Regarding the bitrate, figure 7.7a, even though FQ Codel and PIE showed slightly worse results than DT125kB com, all configurations gave results close to RefWebRTC.

The same applies to the frame rate and frame width (figure 7.7b) and RTT (figure 7.8a).

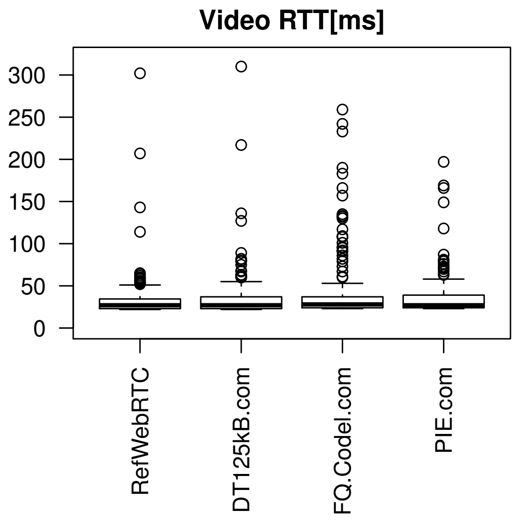


(a) Video bitrate for one test per configuration superposed with RefWebRTC.

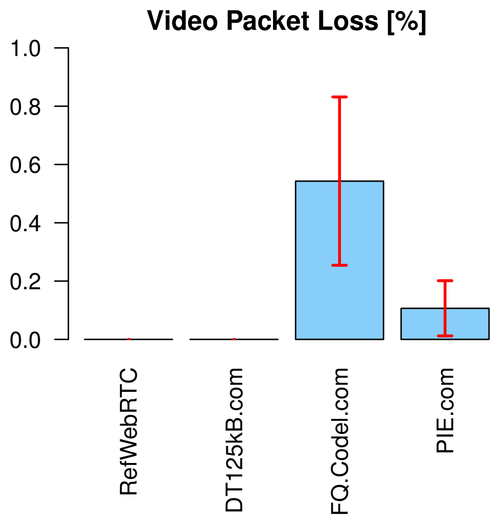


(b) Video frame rate (left axis) and frame width (right axis) for one test per configuration superposed with RefWebRTC.

Figure 7.7 – Impact of target delay based configuration on WebRTC traffic (bitrate, frame rate and frame width) without any concurrent traffic



(a) Video RTT - Boxplot of all test cases per configuration



(b) Video packet loss - Mean and standard deviation of all test cases per configuration

Figure 7.8 – Impact of target delay based configuration on RTT and packet loss of WebRTC traffic without any concurrent traffic

FQ Codel and PIE caused more packet loss, as seen on the figure 7.8b, but it does not significantly impact the frame rate and frame width.

In our opinion, there are two possibilities. First is that even though there is a packet loss, the delay-based controller prevails over the loss-based one, so packet loss does not have much impact on the sending rate.

The second possibility is that, when AQM mechanisms are used, packets are lost not only when a given buffer is full, but also when it is only filling up. Indeed, in case of FQ Codel when Codel enters the dropping state, it starts by dropping one packet per RTT and slowly increases. Thus, packet loss can be more uniformly distributed, so less sudden. Hence, the loss-based controller does not react immediately and reacts only when larger amount of packets is lost at once.

Furthermore, the packet loss handling mechanisms, as explained in the section 2.3.3, are robust enough to handle this packet loss and avoid quality degradation.

Impact of target delay based configuration on WebRTC traffic with TCP concurrent traffic

Figures 7.9 and 7.10 allow us to evaluate the impact of target delay based configuration on the WebRTC communication in the presence of TCP concurrent traffic.

Figure 7.9a shows video bitrate, and figure 7.9b shows the frame rate and frame width for a chosen test case superposed with RefWebRTC.

Figure 7.10 shows the measurements of RTT and packet loss per configuration for all test cases, including the reference RefWebRTC.

Figure 7.11 represents the impact of target delay based configuration on the concurrent traffic, i.e. the upload time of a file over TCP.

Figure 7.12 shows channel utilization of two observed traffics, i.e. WebRTC and TCP.

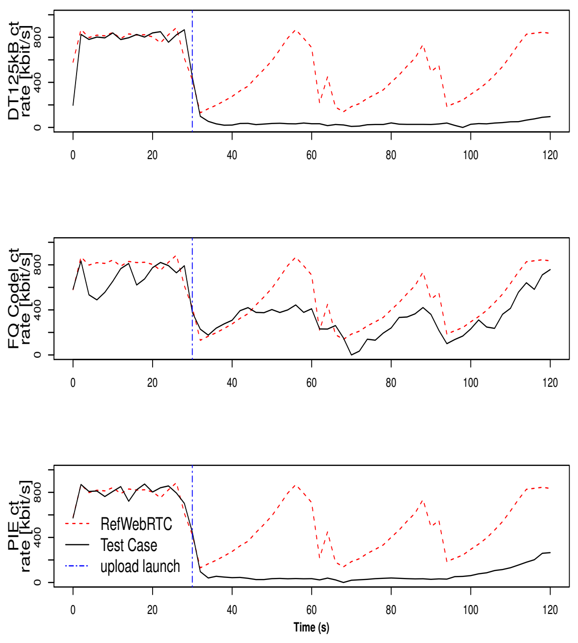
Additionally, on all figures a droptail configuration, test case DT125kB ct, is presented for comparison.

Regarding the bitrate, figure 7.9a, test case PIE is similar to test case DT125kB ct. For PIE WebRTC traffic is also starved, nevertheless it recovers faster than in the case of test case DT125kB ct. In the case of FQ Codel, the traffic is not starved and follows the trend of RefWebRTC, even though its bitrate values are lower than for RefWebRTC.

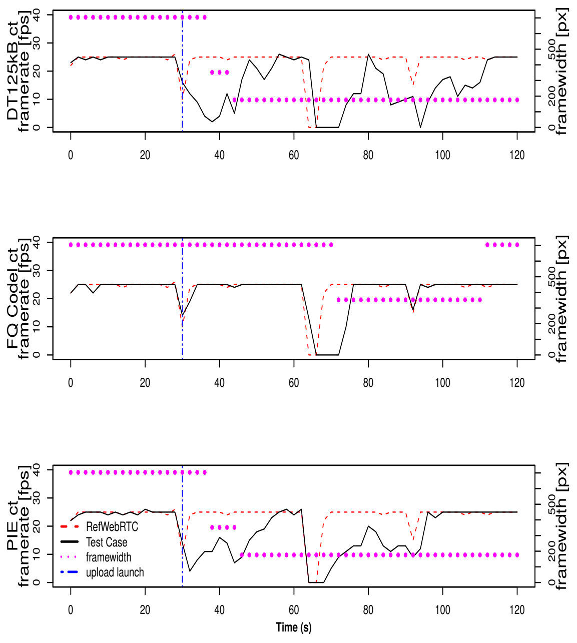
In figure 7.9b, it can be seen that PIE has a similar impact on the frame rate and frame width as observed in DT125kB ct, but again it recovers faster as frame rate is stabilized already at 100s. FQ Codel causes several freezes and a decrease of a frame width, but recovers quickly.

FQ Codel and PIE RTT is closer to RefWebRTC than in case of droptail configurations, as observed on the figure 7.10a. Test case DT125kB ct is not represented for better readability of the schema, since as it was seen on the previous figure 7.4a, it causes much higher RTT (even reaching 1s).

In this context it is interesting to observe the packet loss, figure 7.10b. FQ Codel causes the highest packet loss, over 1%, but still offers the best frame rate and frame width.

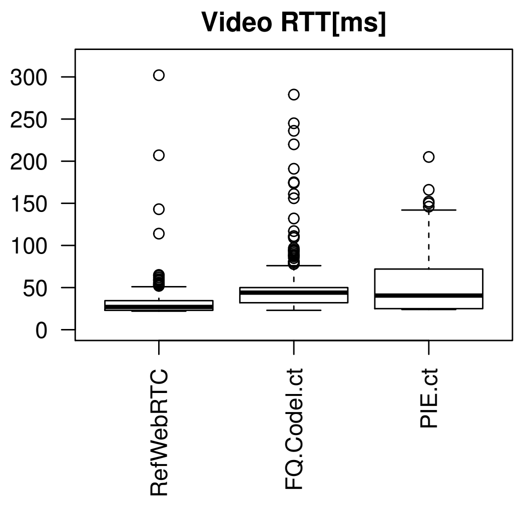


(a) Video bitrate for one test per configuration superposed with RefWebRTC.

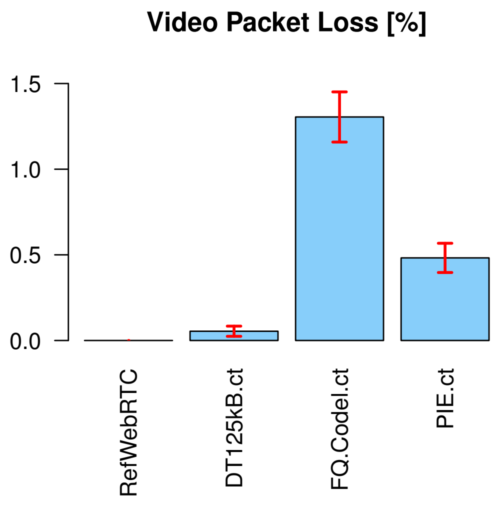


(b) Video frame rate (left axis) and frame width (right axis) for one test per configuration superposed with RefWebRTC.

Figure 7.9 – Impact of target delay based configuration on WebRTC traffic (bitrate, frame rate and frame width) with TCP concurrent traffic



(a) Video RTT - Boxplot of all test cases per configuration



(b) Video packet loss - Mean and standard deviation of all test cases per configuration

Figure 7.10 – Impact of target delay based configuration on RTT and packet loss of WebRTC traffic with TCP concurrent traffic

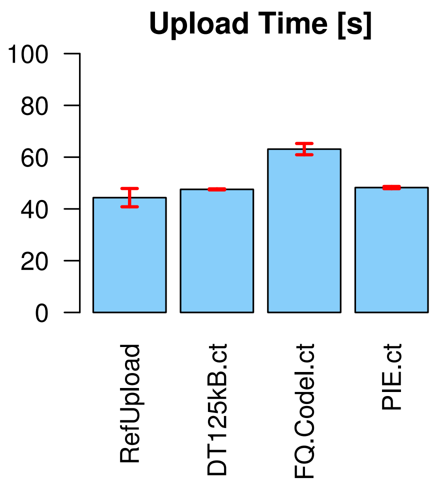


Figure 7.11 – Impact of target delay based configuration on concurrent traffic upload time

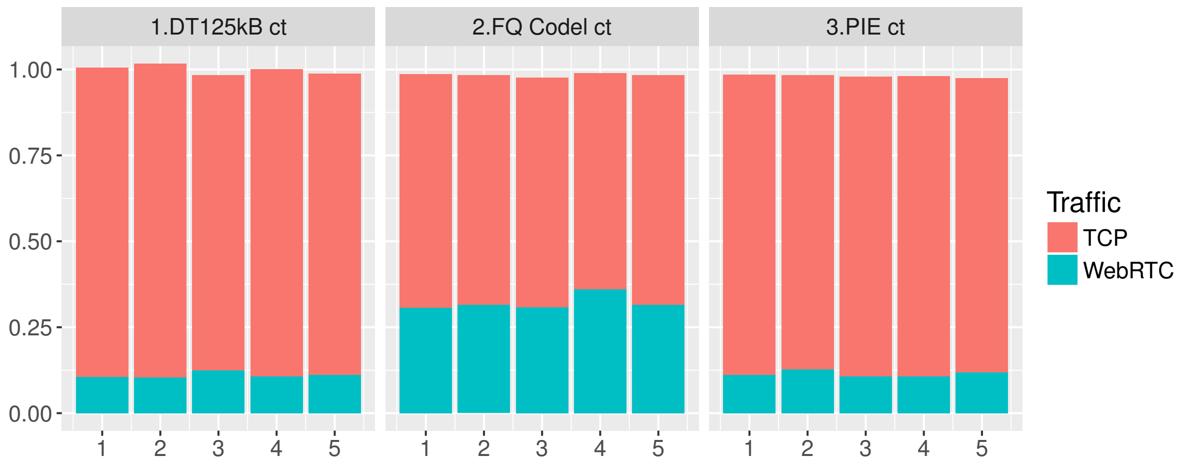


Figure 7.12 – Impact of target delay based configuration on channel utilization

The figure 7.11 shows that PIE and DT125kB have a similar impact on the concurrent TCP traffic, as the upload times were close, whereas FQ Codel increased the upload time.

Regarding the channel utilization, figure 7.12, again FQ Codel is more fair to WebRTC flow than other configurations.

In our opinion, FQ Codel gives a good performance, because it provides more resources to the WebRTC traffic and avoids its starvation in the presence of a TCP flow.

PIE performs worse, it does not provide enough resources to WebRTC flows and consequently does not avoid WebRTC flow starvation. Nevertheless, it provides better recovery time for WebRTC traffic than test case DT125kB ct. In our opinion, the explanation of this behaviour is similar to the one we gave when analysing the droptail configuration. In PIE the buffer is shorter, so it stabilizes faster and needs less time to start increasing the sending rate.

Summary of the impact of target delay based configuration on WebRTC traffic

The evaluated test cases prove that FQ Codel is able to provide better resource sharing between different types of flows. FQ Codel cooperates well with the GCC algorithm since, even though it causes more packet loss, it ensures low delays, so it provides a better quality than other configurations tested in this context. Furthermore, quality degradation caused by certain packet loss can be limited by using appropriate mechanisms, as it was explained in the section 2.3.3. Thus, in our opinion, in this context it is more important to ensure low delays even though it can introduce a certain packet loss.

On the other hand, PIE performs worse, but still recovers faster than droptail with a large buffer, DT125kB.

7.2.4 Evaluation of rate and queue length based classful configuration

In this section we present the evaluation of the rate and queue length based classful configuration, i.e. a classful configuration. The summary of the test cases used for evaluation of the rate and queue length based classful configuration is given in the table 7.10.

The classful test cases are compared with the reference measurements, i.e. RefWebRTC and RefUpload. They are also compared with measurements performed on a droptail configuration, DT125kB, with a buffer size of 125kB, test case DT125kB com and test case DT125kB ct, that represents a general trend of wide spread droptail approach.

For test cases without any concurrent traffic, only one configuration is tested, since only one class is used, i.e. the WebRTC class with SFQ qdisc. Hence, for this test case any HTB configuration can be used, since all of them contain the same type of WebRTC class.

Test case name	CPE config	Concurrent Traffic
RefWebRTC (reference)	Droptail default	none
DT125kB com (reference)	Droptail 125kB	none
DT125kB ct (reference)	Droptail 125kB	TCP file upload
HTB-SFQ com	any HTB config with WebRTC->sfq TC qdisc	none
H204 ff-def ct	HTB204 fifo def	TCP file upload
H654 ff-def ct	HTB654 fifo def	TCP file upload
H204 ff-125 ct	HTB204 fifo 125kB	TCP file upload
H654 ff-125 ct	HTB654 fifo 125kB	TCP file upload
H204 ared ct	HTB204 ared	TCP file upload
H654 ared ct	HTB654 ared	TCP file upload

Table 7.10 – Test matrix ADSL uplink rate and queue length based classful configuration

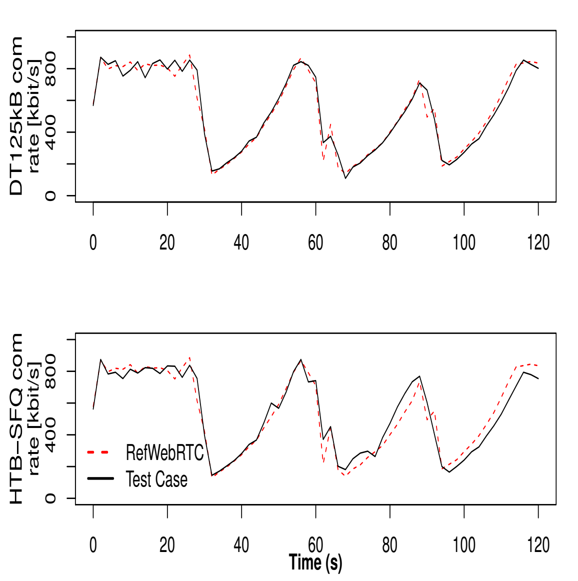
Impact of rate and queue length based classful configuration on WebRTC traffic without any concurrent traffic

Figures 7.13 and 7.14 allow us to evaluate the impact of rate and queue length based classful configuration on the WebRTC communication without any concurrent traffic.

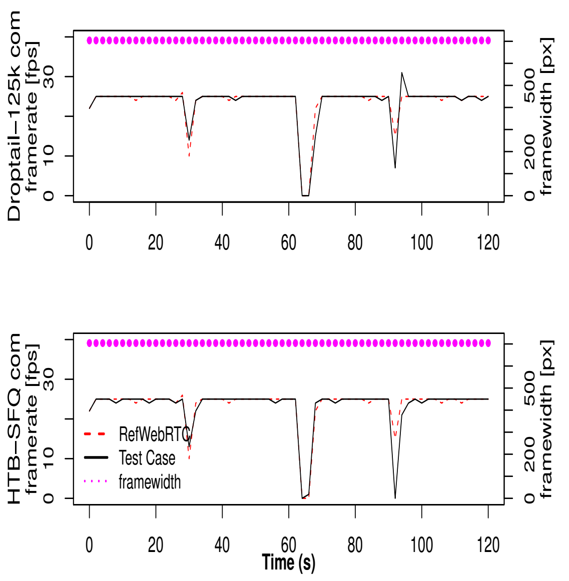
Figure 7.13a shows video bitrate, and figure 7.13b shows the frame rate and frame width. For better readability, only one test case per configuration is given. Each test case is superposed with the reference RefWebRTC.

Figure 7.14 shows the measurements of RTT and packet loss per configuration, but this time for all test cases, including the reference RefWebRTC.

Additionally, on all figures a droptail configuration, test case DT125kB ct, is presented for comparison.



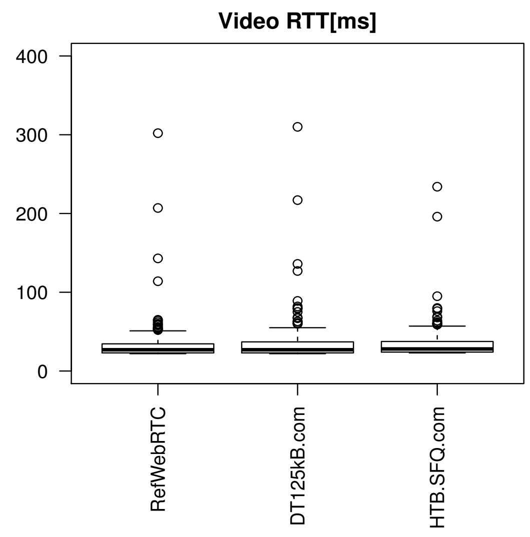
(a) Video bitrate for one test per configuration superposed with RefWebRTC.



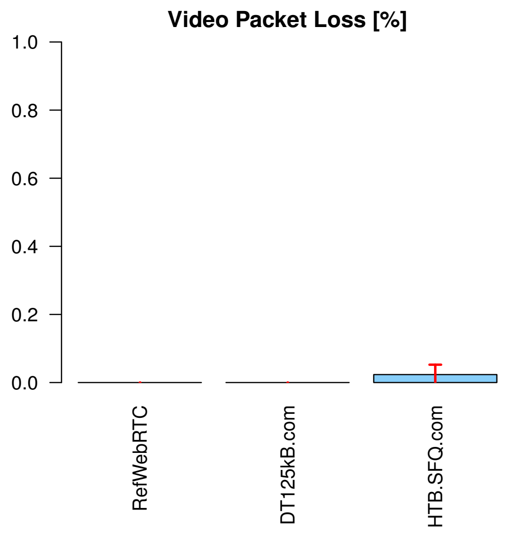
(b) Video frame rate (left axis) and frame width (right axis) for one test per configuration superposed with RefWebRTC.

Figure 7.13 – Impact of rate and queue length based classful configuration on WebRTC traffic (bitrate, frame rate and frame width) without any concurrent traffic

From figures 7.13a and 7.13b, we can see that the HTB-SFQ com test case results are close to the reference performance, regarding the bitrate, frame rate and frame width. It causes slightly more packet loss, as seen on the figure 7.14b, but it does not impact the overall performance.



(a) Video RTT - Boxplot of all test cases per configuration



(b) Video packet loss - Mean and standard deviation of all test cases per configuration

Figure 7.14 – Impact of rate and queue length based classful configuration on RTT and packet loss of WebRTC traffic without any concurrent traffic

Impact of rate and queue length based classful configuration on WebRTC traffic with TCP concurrent traffic

Figures 7.15 and 7.16 represent a set of schemas allowing to evaluate the impact of rate and queue length based classful configuration on the WebRTC communication in the presence of TCP concurrent traffic.

Figure 7.15a shows video bitrate, and figure 7.15b shows the frame rate and frame width for a chosen test case superposed with RefWebRTC.

Figure 7.16 shows the measurements of RTT and packet loss per configuration for all test cases, including the reference RefWebRTC.

Figure 7.17 represents the impact of rate and queue length based classful configuration on the concurrent traffic, i.e. the upload time of a file over TCP.

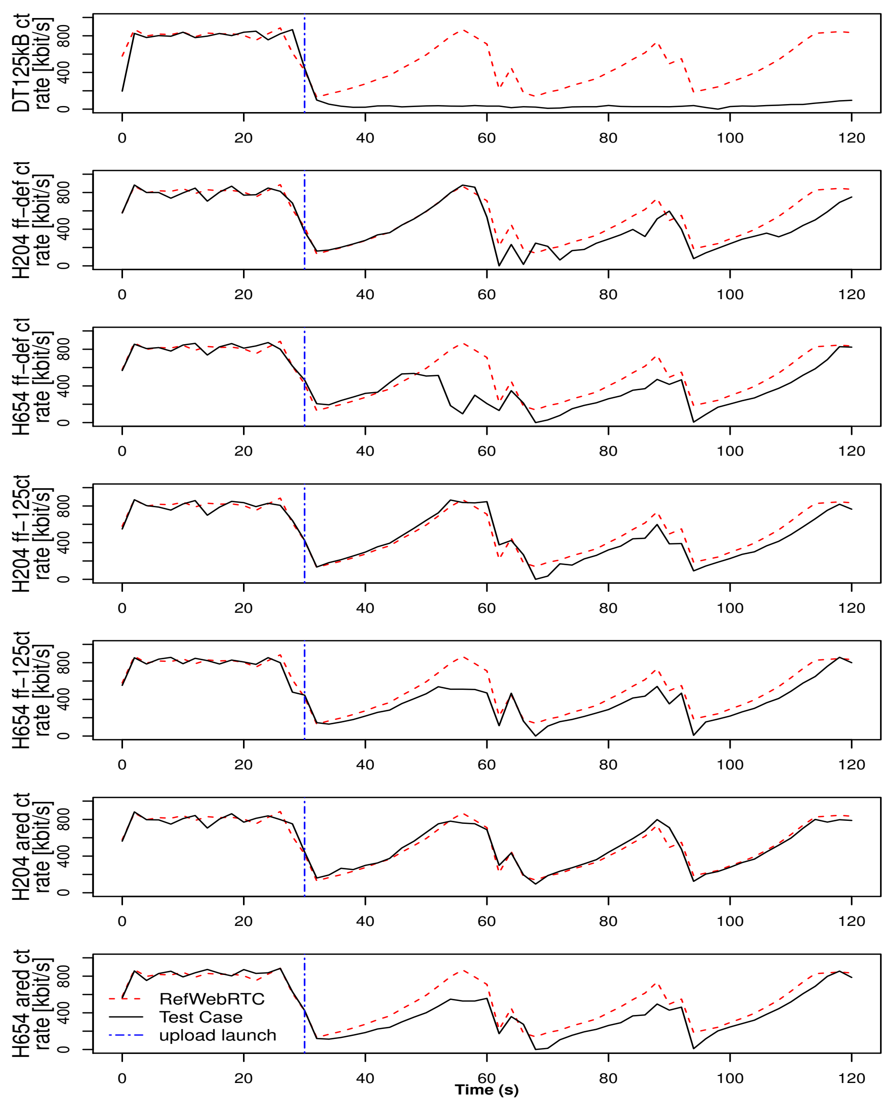
Figure 7.18 shows channel utilization of two observed traffics, i.e. WebRTC and TCP.

Additionally, on all figures a droptail configuration, test case DT125kB ct, is presented for comparison.

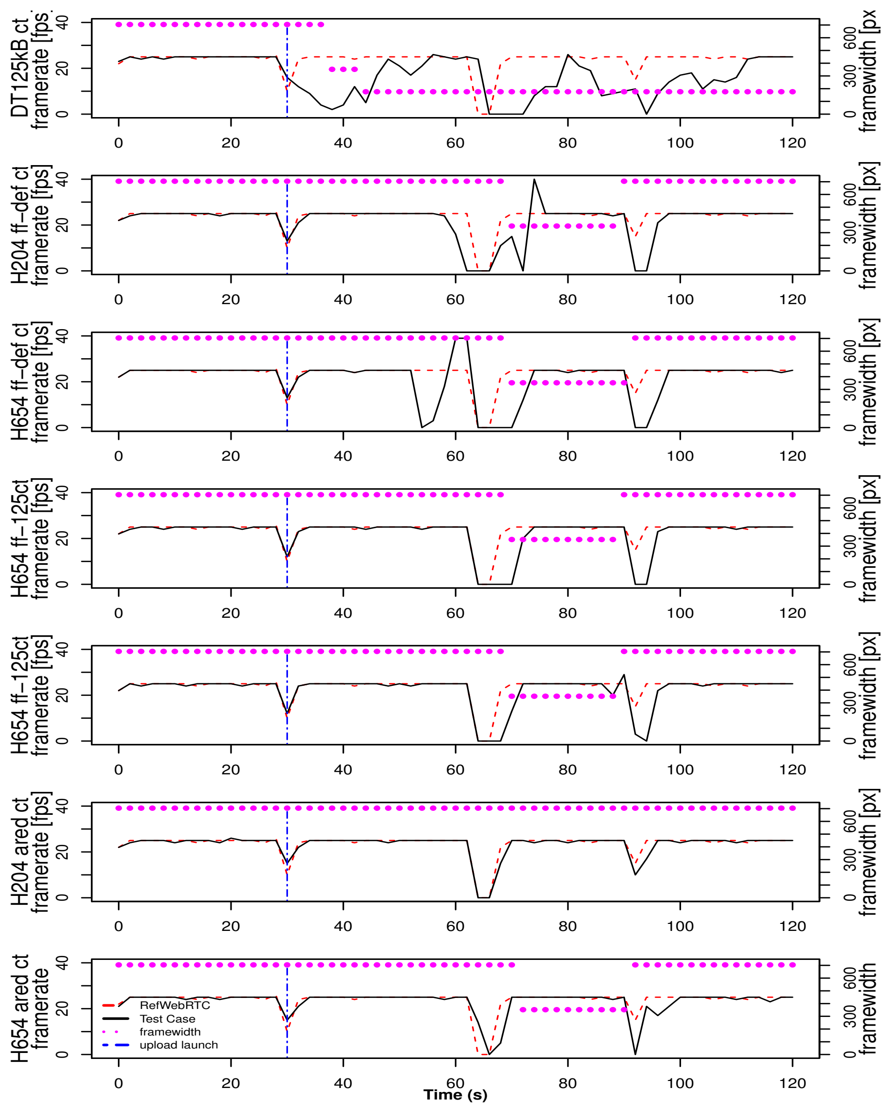
Classful configurations significantly improve WebRTC traffic bitrate, as seen on the figure 7.15a. Especially, for HTB204 configurations, test cases: HTB204 ff-def ct, HTB204 ff-125 ct and HTB204 ared ct, the obtained bitrate is very close to the reference RefWebRTC. The WebRTC traffic is not starved as it happens for DT125kB ct test case.

As observed on the figure 7.15b, classful configurations have also improved the frame rate, some freezes can be observed (frame rate close to zero), but overall performance is close to the reference RefWebRTC. In most cases, the frame width decreases for a short time, but recovers very fast.

Additionally, test case HTB204 ared ct obtained performance as good as RefWebRTC.

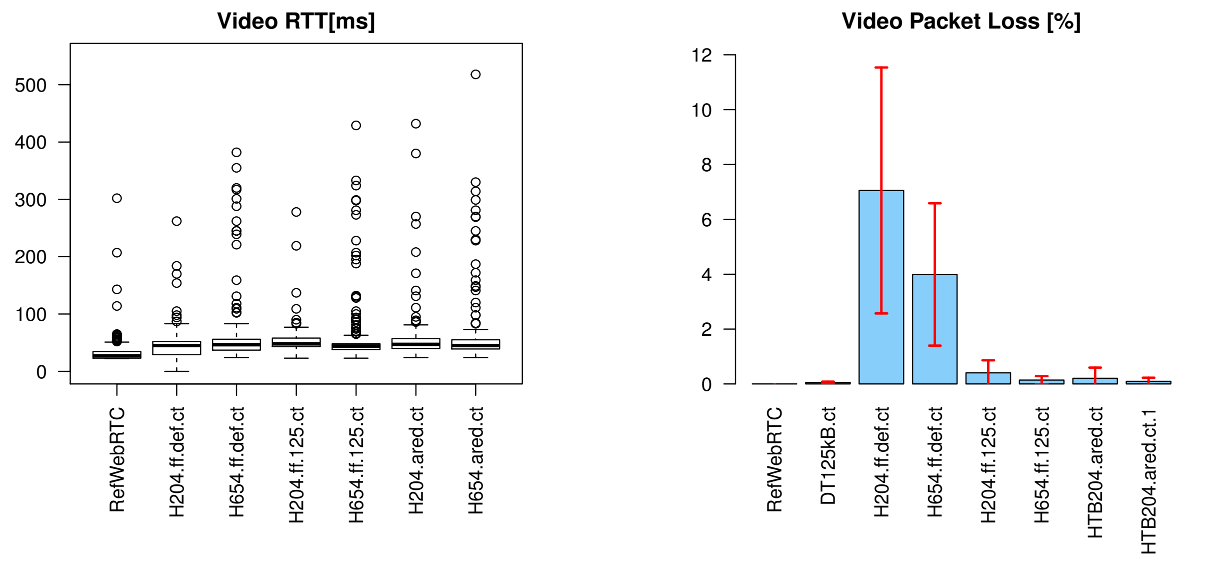


(a) Video bitrate for one test per configuration superposed with RefWebRTC.



(b) Video frame rate (left axis) and frame width (right axis) for one test per configuration superposed with RefWebRTC.

Figure 7.15 – Impact of rate and queue length based classful configuration on WebRTC traffic (bitrate, frame rate and frame width) with TCP concurrent traffic



(a) Video RTT - Boxplot of all test cases per configuration

(b) Video packet loss - Mean and standard deviation of all test cases per configuration

Figure 7.16 – Impact of rate and queue length based classful configuration on RTT and packet loss of WebRTC traffic with TCP concurrent traffic

Rate and queue length based classful configurations slightly increase the RTT, as seen in figure 7.16a, but it still stays close to RefWebRTC. Test case DT125kB ct is not represented for better readability of the schema, since as it was seen on the figure 7.4a, it causes much higher delay.

For most of classful configurations, there is packet loss but it is still kept low, below 1% as seen on the figure 7.16b. However, test cases HTB204 ff-def ct and HTB654 ff-def ct caused higher packet loss. It is caused by the length of best-effort class queue. Since the queue is very long, the TCP sender does not react correctly to congestion and it continuously increases its congestion window. Hence, it causes a bursty traffic and for instance, if a packet is lost somewhere on the way, TCP sender takes time to realize it, because of the delays caused by excessively large buffer. When it gets a missing acknowledge, it saturates the link with all postponed packets. This burstiness is avoided when shorter queues for the best-effort class are used, since they improve TCP reactivity.

Nevertheless, even for packet loss close to 10%, WebRTC still provided a better performance (bitrate, frame rate and frame width) than in case of droptail configuration, test case DT125kB ct, that provided less packet loss, but much higher delay. In our opinion, the better performance is observed, because the WebRTC traffic is not starved, so even if at some point there is congestion, WebRTC recovers quickly.

Rate and queue length based classful configurations improve the WebRTC quality but they have also an impact on the TCP concurrent traffic as presented on the figure 7.17. The better WebRTC quality is achieved, the longer it takes to upload the TCP file.

Regarding channel utilization, as seen on the figure 7.18, rate and queue length based classful configurations are more adapted to real-time traffic. Hence, they ensure a better bandwidth share between TCP and WebRTC flows.

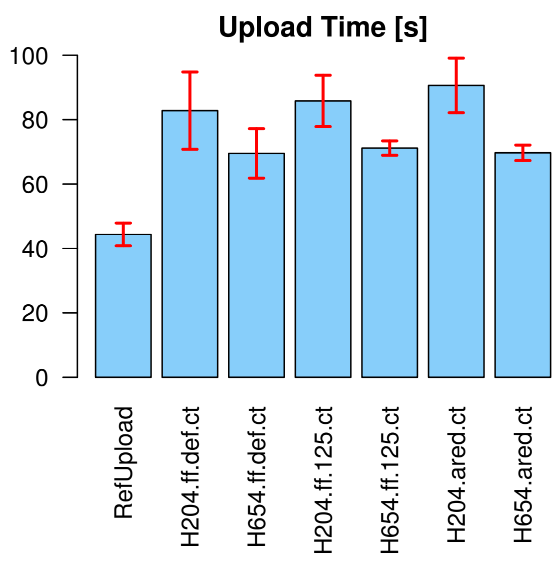


Figure 7.17 – Impact of rate and queue length based classful configuration on concurrent traffic upload time

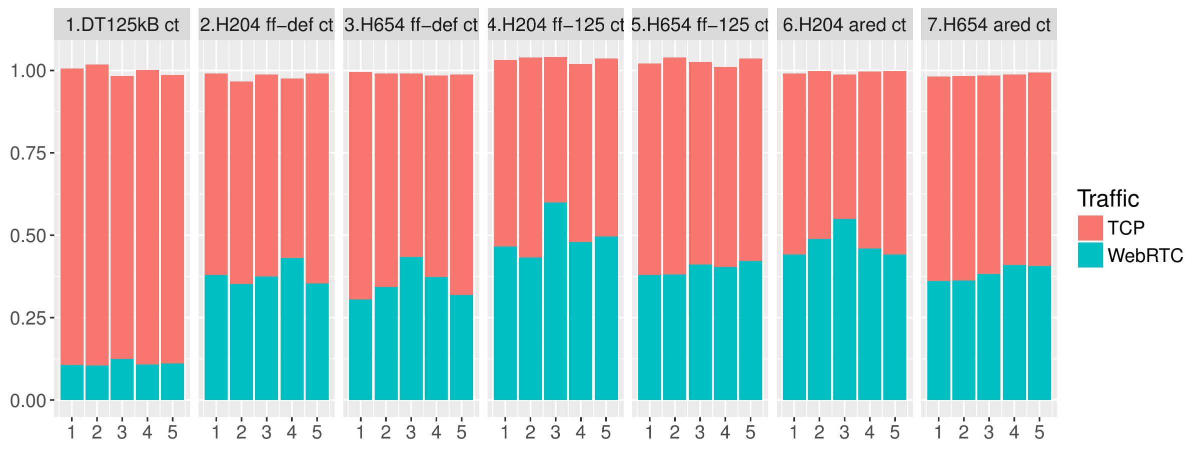


Figure 7.18 – Impact of rate and queue length based classful configuration on channel utilization

Summary of the impact of rate and queue length based classful configuration on WebRTC traffic

The evaluated test cases prove that classful configurations are able to provide better resource sharing between different types of flows. They provide a very good WebRTC quality, however they also impact the most the concurrent traffic.

It is important to point out, that classful configurations not only allow avoiding flow starvation, but enable obtaining WebRTC quality close to the one without any concurrent traffic. Additionally, HTB204 configurations performed better than HTB654. Hence, for WebRTC it is better to prioritize, while guarantying a minimum rate, than to guarantee a higher optimal rate, but with lower priority. Nevertheless, HTB654 provided an acceptable quality and caused less delay in packet upload.

In brief, classful configuration allows good separation between different traffic types. Nonetheless, it is important to provide an appropriate treatment to the WebRTC class, but also to correctly configure the best-effort class, e.g. excessive large buffer should be also avoided.

7.2.5 Comparison of the principal results of identified approaches

In order to present the overall results evaluation and assessment for the 1Mbit/s ADSL uplink, in this section, we compare the best performance obtained from two identified and tested approaches, i.e. target delay based and rate and queue length based classful approach. We compare them with a typical droptail configuration.

The summary of analysed test cases is given in the table 7.11. We focused on the impact on the WebRTC communication in the presence of concurrent TCP traffic.

Test case name	**CPE config**	**Concurrent Traffic**
RefWebRTC (reference)	Droptail default	none
DT125kB ct	Droptail 125kB	TCP file upload
FQ Codel ct	FQ Codel	TCP file upload
H204 ared ct	HTB204 ared	TCP file upload
H654 ared ct	HTB654 ared	TCP file upload

Table 7.11 – Selected configurations for comparison of principle results

Figure 7.19a shows video bitrate, and figure 7.19b shows the frame rate and frame width for selected test cases superposed with RefWebRTC.

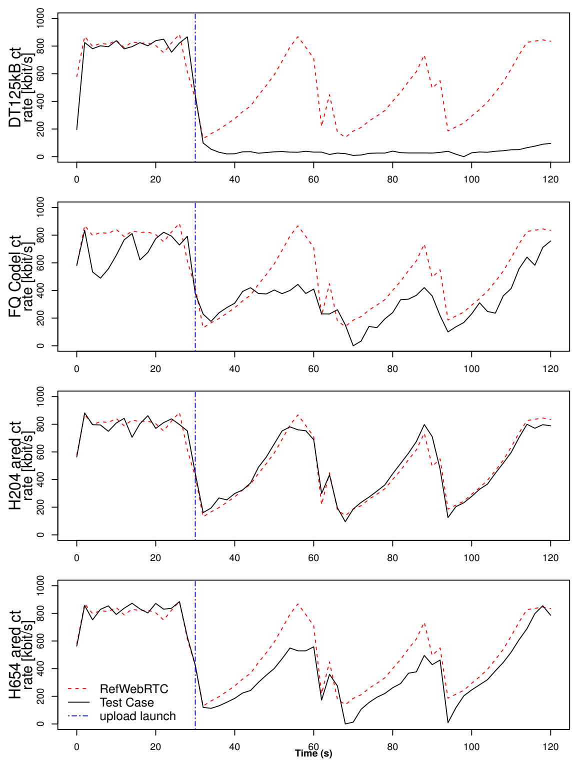
Figure 7.20 shows the measurements of RTT and packet loss per configuration for all test cases, including the reference RefWebRTC.

Figure 7.21 represents the impact of selected configurations on the concurrent traffic, i.e. the upload time of a file over TCP. The results are compared with the reference RefWebRTC.

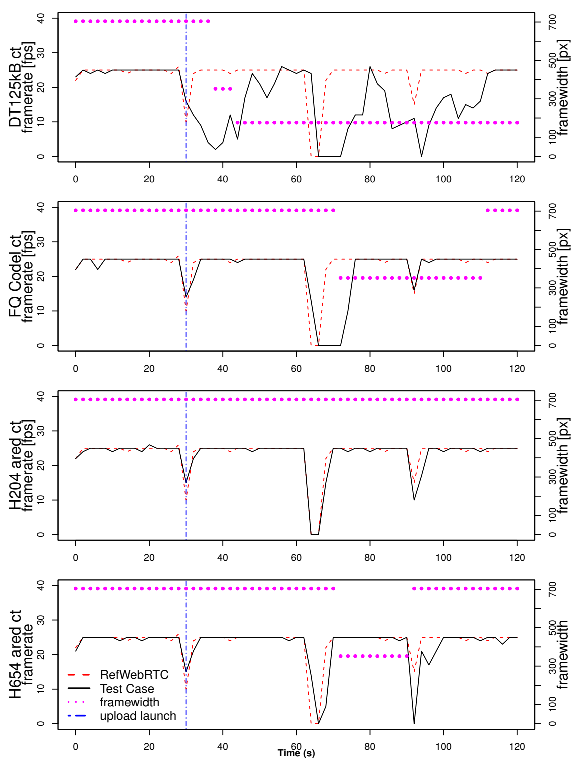
Figure 7.22 shows the impact of selected configurations channel utilization of two observed traffics, i.e. WebRTC and TCP.

As it can be observed on the figures 7.19a and 7.19b, droptail, test case DT125kB ct, performs significantly worse than other configurations. It also took time to recover from the negative congestion influence. FQ Codel, test case FQ Codel ct, and HTB654 with ared, H654 ared ct, provide acceptable quality and ensure faster recovery of WebRTC flows in case of congestion problems. HTB204 with ared, test case H204 ared ct, provides quality as good as the reference, RefWebRTC.

Based on the figure 7.20, in test case DT125kB ct, we obtained the lowest packet loss and the highest RTT, whereas in test case FQ Codel ct, the packet loss is higher but the RTT is kept low. Nevertheless, FQ Codel provided far better WebRTC performance, whereas droptail configuration caused WebRTC flow starvation and unacceptable quality.

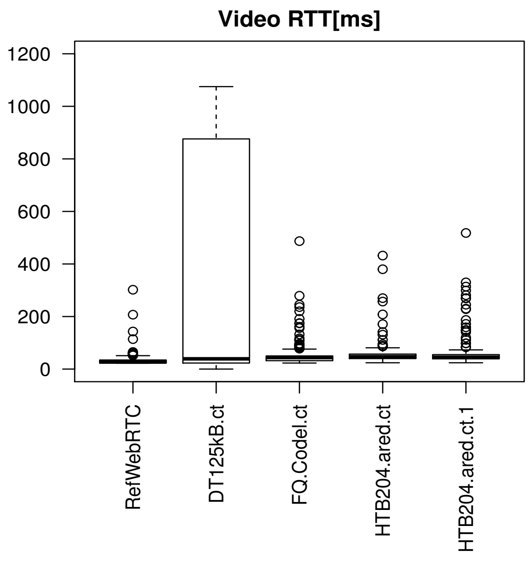


(a) Video bitrate for one test per configuration superposed with RefWebRTC.

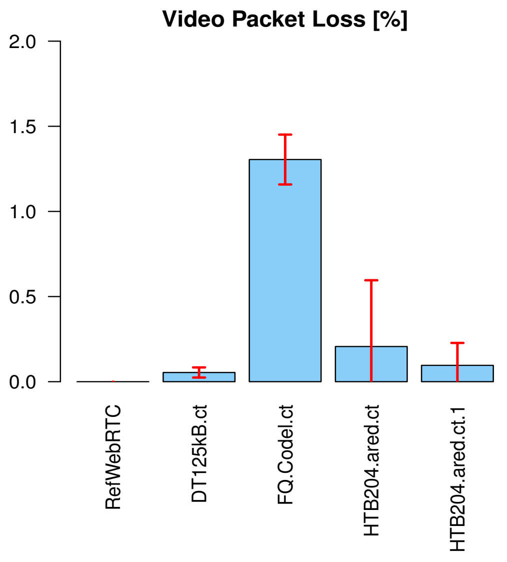


(b) Video frame rate (left axis) and frame width (right axis) for one test per configuration superposed with RefWebRTC.

Figure 7.19 – Selected results of different configurations and their impact on WebRTC video bitrate, frame rate and frame width in the presence of TCP concurrent traffic



(a) Video RTT - Boxplot of all test cases per configuration



(b) Video packet loss - Mean and standard deviation of all test cases per configuration

Figure 7.20 – Selected results of different configurations and their impact on WebRTC video RTT and packet loss in the presence of TCP concurrent traffic

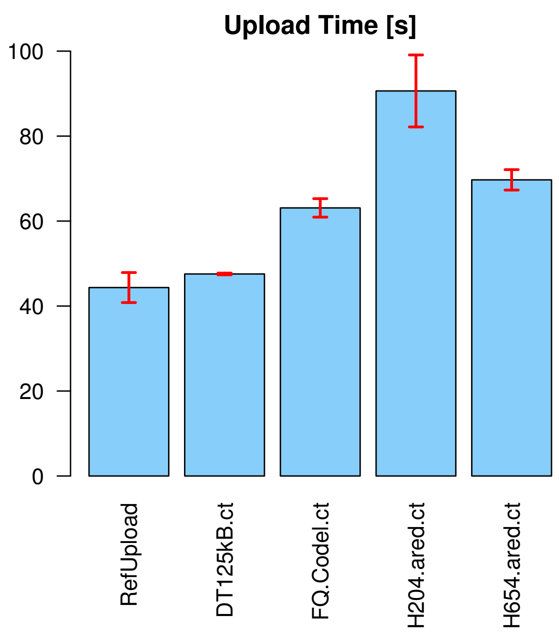


Figure 7.21 – Selected results of different configurations and their impact on concurrent traffic upload time

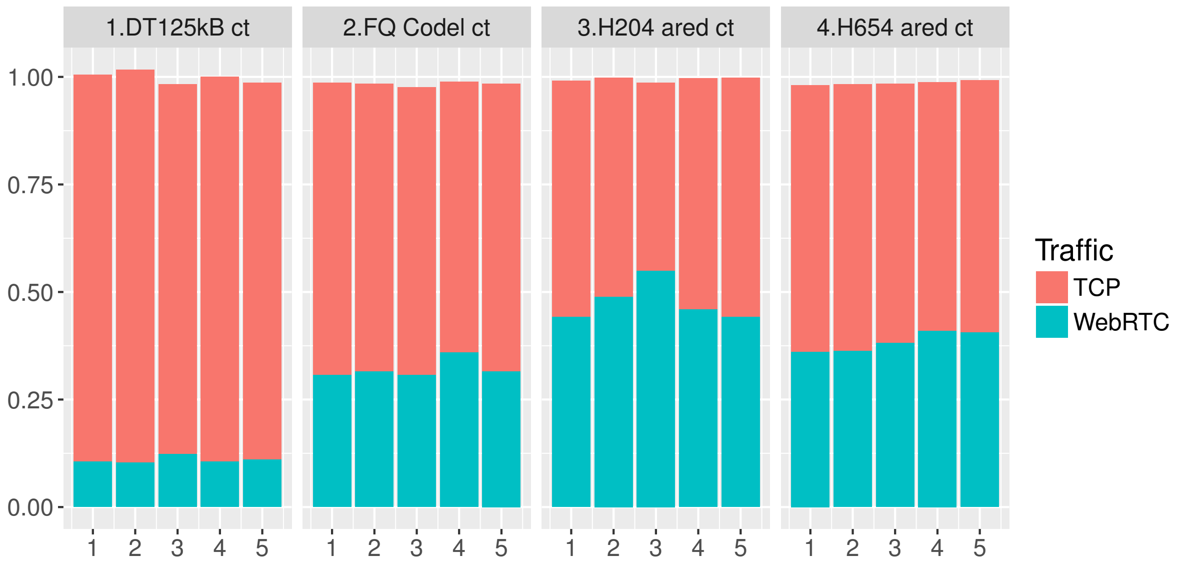


Figure 7.22 – Selected results of different configurations and their impact on channel utilization

In the evaluated configurations, GCC was especially vulnerable to excessive delays that caused flow starvation. However, if a more balanced share of bandwidth between TCP and WebRTC flows was ensured, GCC performed well. Even if network fluctuations caused punctually some packet loss or higher delay, GCC could recover fast.

In our opinion, the WebRTC traffic in droptail configuration is vulnerable, even with GCC assistance. Hence, WebRTC can benefit from network configuration that ensures more optimal resources sharing and is complementary to the endpoint mechanisms.

Figure 7.21 shows that all the configurations that improved WebRTC quality, increased the upload time of a TCP file, representing the concurrent traffic. In case of test case H204 ared ct, this upload time was the longest.

The droptail approach is more adapted to TCP flows, so it privileges high throughput while ensuring no packet loss. On the contrary, target delay based and rate and queue length based classful approaches are more adapted to real-time traffic. They provide fairer channel utilization and avoid WebRTC flow starvation.

The subjective opinion about the perceived quality has confirmed the above results. Droptail quality was not acceptable, because of lack of fluidity of the videos flow, along with many freezes and a very small video widow size. HTB204 with ared offered the best quality and it was as good as the reference one. FQ Codel and HTB654 with ared performed well. They caused some freezes and the video frame size decreased for certain time, which negatively impacted the perceived quality. Nevertheless, in both cases the video went back to normal in a short time and these quality fluctuations were less disturbing, because the overall video flow remained fluid.

7.3 Results assessment for 200Mbit/s fiber uplink

7.3.1 Test description

We run some additional tests for the fiber access networks. As explained in the section 6.4.3, in fiber networks it is more difficult to generate congestion, so we used iperf instead of simple TCP file upload.

With iperf we were able to generate several TCP flows. Hence, we evaluated the WebRTC communication in the presence of 1, 5 and 100 TCP flows. We did not need to run any test case with only WebRTC connection, since the link is oversized and WebRTC traffic can easily get the required bandwidth, since the video flow demands about 2Mb/s, which is a small fraction of available capacity.

We started by the evaluation of the widespread droptail configuration, i.e. a configuration based on the current IP network engineering. Later, we tested the target delay based queuing mechanisms and rate and queue length based classful queuing mechanisms. We compared the obtained results with the ones assessed for droptail configuration.

Each test case corresponds to a 120 second WebRTC communication. The TCP traffic is launched 30s after the WebRTC call is established and is run for 60s. Each test case is repeated 3 times for every CPE configuration. Since the tests were attended, running too many tests would be time consuming. In order to have enough data, we decided that for fiber configuration 3 tests were sufficient.

The measurements used for evaluation are given in the table 7.12. The details about collecting these statistics were given in the section 6.2.

Evaluation of different fiber configurations

In this section we present the evaluation of different fiber configurations on WebRTc traffic in the presence of TCP concurrent traffic. The summary of the test cases is given in the table 7.13.

Measurement name	Measurement source	Measurement type
Video bitrate	getStats	Application layer
Video frame rate	getStats	Application layer
Video frame width	getStats	Application layer
Video packet loss	getStats	Application layer
Video RTT	getStats	Application layer

Table 7.12 – Measurements used for fiber evaluation

Test case name	CPE config	Concurrent Traffic
DT TCP1	Droptail fiber	iperf - 1 TCP flow
DT TCP5	Droptail fiber	iperf - 5 TCP flows
DT TCP100	Droptail fiber	iperf - 100 TCP flows
HTB TCP100	HTB fiber	iperf - 100 TCP flows
FQ Codel TCP100	FQ Codel	iperf - 100 TCP flows
PIE TCP100	PIE	iperf - 100 TCP flows

Table 7.13 – Test matrix fiber uplink

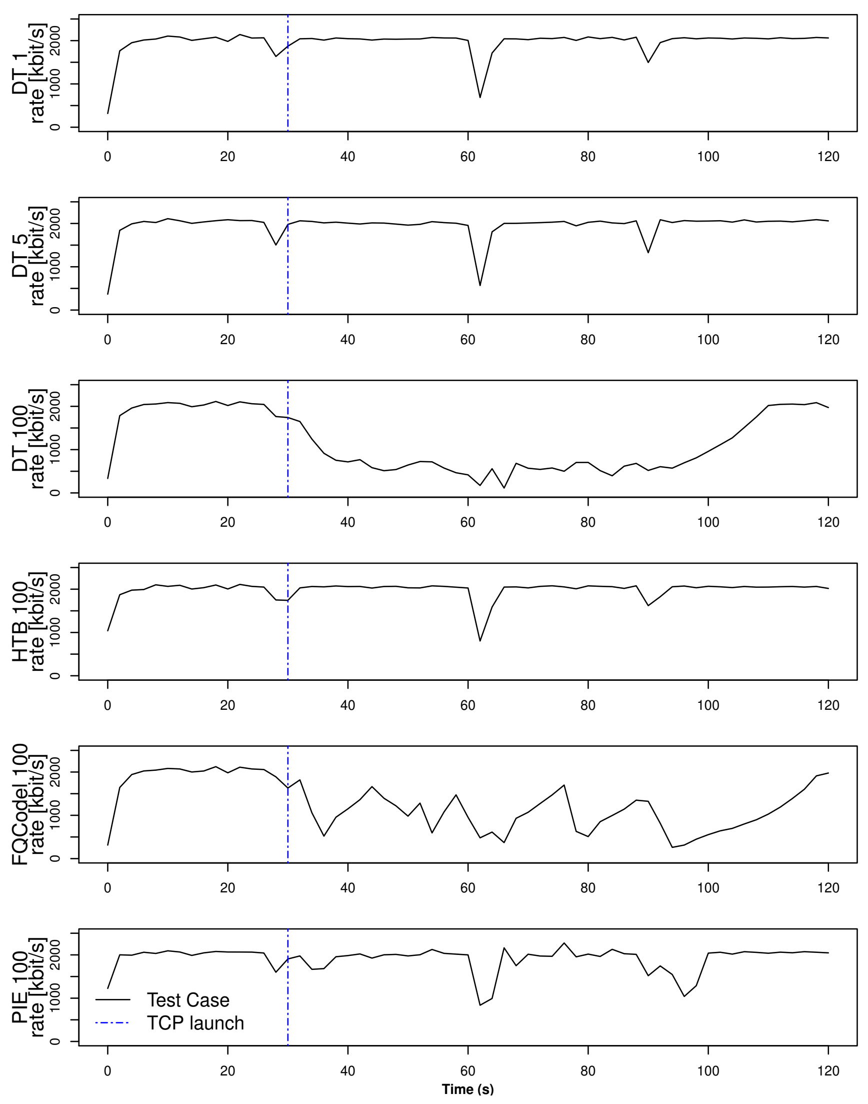
Figure 7.23a shows video bitrate, and figure 7.19b shows the frame rate and frame width for one test case per configuration.

Figure 7.24 shows the measurements of RTT and packet loss per configuration for all test cases.

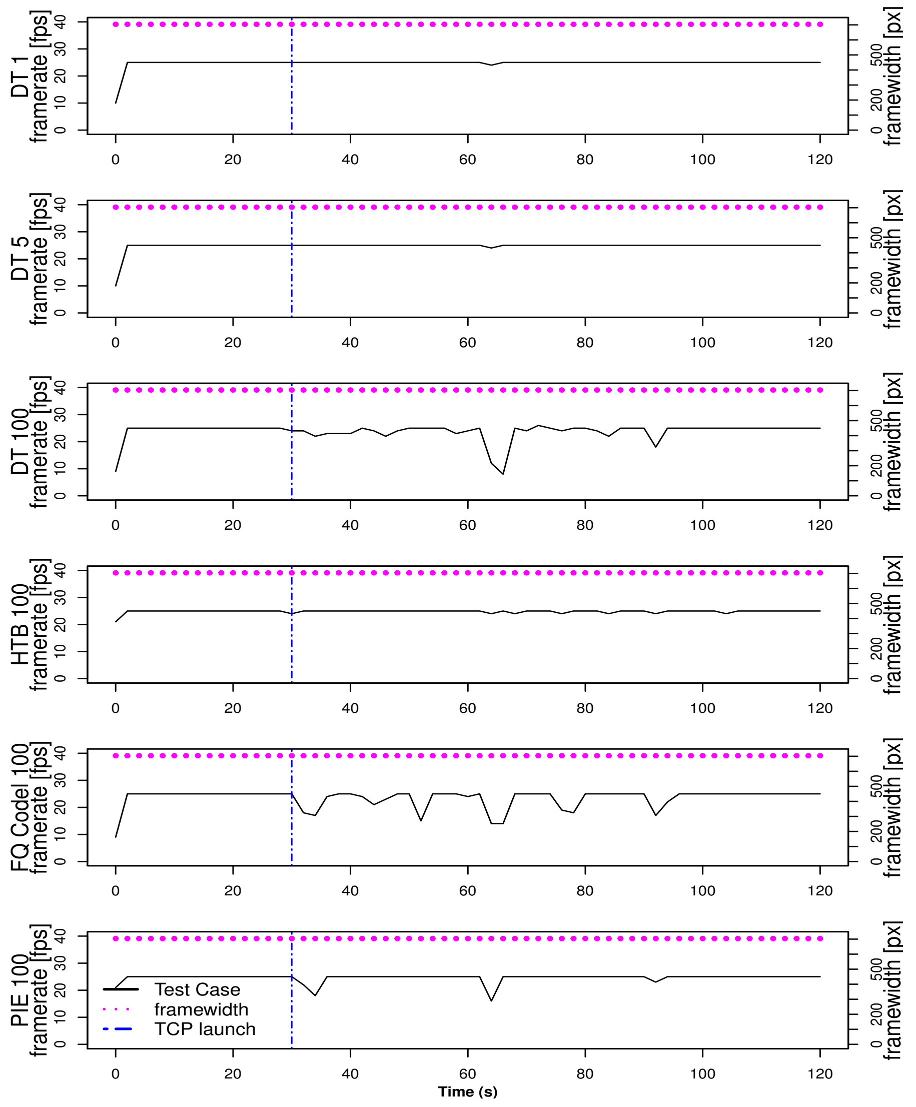
As it can be seen on the figure 7.23, it is difficult to congest the fiber uplink and perturb the WebRTC traffic. In the presence of 1 and 5 TCP flows, there is no impact on WebRTC communication. Given the limits of TCP, caused by its dependence on RTT and the sizes of congestion and receiver windows, 1 or 5 TCP flows do not fully fill the link. Since TCP does not utilize the full capacity of oversized link, it leaves enough capacity to a WebRTC flow. In fact, WebRTC needs up to 2Mbit/s for 720p video quality (**Grigorik 2013**). This is a small part of available 200Mbit/s.

Nonetheless, we managed to create the congestion with 100 TCP flows. Thus, on the figure 7.23a, it can be seen that the bitrate decreased for the duration of TCP concurrent traffic, test case DT 100. However, even though the bitrate decreased, the overall level of quality did not change, the frame rate was hardly impacted and the frame width value did not change.

On the figures 7.23a and 7.23b it can be observed that HTB configuration, test case HTB100 and AQM configurations, test cases FQ Codel 100 and PIE 100, improved the WebRTC bitrate in case of congestion. The HTB performed as well as the initial one.



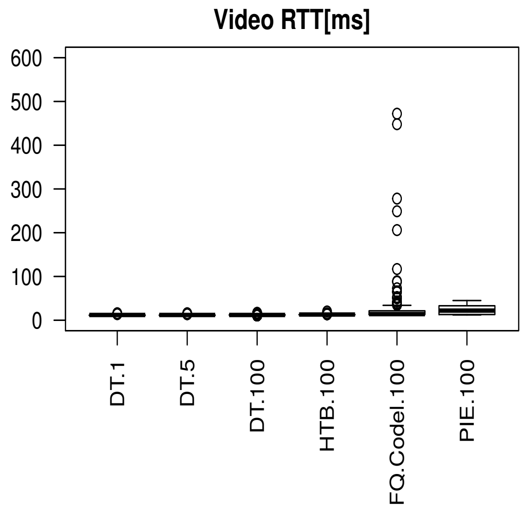
(a) Video bitrate for one test per configuration.



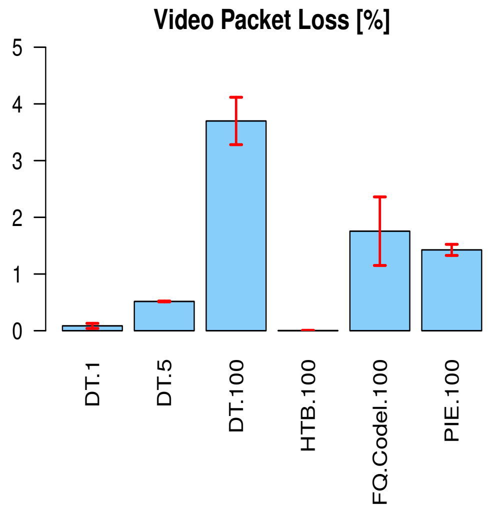
(b) Video frame rate (left axis) and frame width (right axis) for one test per configuration.

Figure 7.23 – Impact of different fiber configurations on WebRTC traffic in the presence of TCP concurrent traffic

‘



(a) Video RTT - Boxplot of all test cases per configuration



(b) Video packet loss - Mean and standard deviation of all test cases per configuration

Figure 7.24 – Selected results of different fiber configurations and their impact on WebRTC video RTT and packet loss in the presence of TCP concurrent traffic

FQ Codel provided smaller improvement, it was also the only configuration that caused the delays, as seen on the figure 7.24a. In our hypothesis, it is linked with the FQ Codel algorithm characteristics. FQ Codel requires a big amount of state information, including packet information (the time it spends in a buffer) and flow information (deficit per queue). It also holds a list of *old* and *new* queues. Since we do not use a typical router, but a Linux router with iptables, such an amount of state information increases the processing complexity. Therefore, it can cause additional treatment delay and consequently negatively impact the performance.

Regarding the figure 7.24b, HTB provided the lowest packet loss, whereas FQ Codel and PIE caused more packet loss, but still the obtained value was lower than 2%.

7.3.2 Summary of the impact of different fiber configurations on WebRTC traffic

Fiber link capacity oversizing guarantees good WebRTC performance even in the presence of concurrent flows. Hence, in general, there is no negative impact on WebRTC in case of congestion in fiber networks. Nevertheless, in case of heavy congestion, the WebRTC sending rate is decreased.

The identified mechanisms improve the WebRTC bitrate. HTB and PIE allow achieving initial quality. FQ Codel provided the smallest improvement, which to our opinion is caused by characteristics of FQ Codel algorithm.

Regarding the subjective opinion, the video quality was very good, regardless the test case.

7.4 Positioning of performed evaluation in relation to different studies

As mentioned before, WebRTC is currently under standardization, so the GCC algorithm is documented. Furthermore, there are some detailed studies focusing on its performance. In this section, we have decided to position our work in relation to the existing studies, so that a reader can have a full understaning of the research aspects.

Singh et al., in (**Singh et al. 2013**), present a study of the evaluation of GCC in the presence of different transport impairments, i.e. changing throughput (1 and 5Mb/s), packet loss (from 0 to 20%) and delay (from 0 to 500ms), along with various concurrent traffic, notably TCP. They have found that if there is packet loss, but no delay, GCC performs well even if packet loss reaches 10%. Moreover, GCC performs well with latencies up to 200ms. Furthermore, they found that WebRTC performs well when by itself and in the presence of certain concurrent traffic, notably similar RTP flows. However, it can also be starved in the presence of TCP traffic. In our study we could confirm this conclusion.

The study also tested WebRTC performance for changing queue lengths. They evaluated different buffer lengths (100ms, 1s and 10s) for two bottleneck rates (1Mbit/s and 5Mbit/s) and their impact on WebRTC without any concurrent traffic. For evaluation they used *Average Bandwidth Utilization* (ABU) that corresponds to our channel utilization. They have found that, regardless the queue length and bottleneck rates, the ABU was around 0.4, so they concluded that router queue lengths do not have any impact on GCC performance.

We cannot exactly adress these measurements and compare them with our results, as we have different test environment and input video. However, we have proven that changing buffer lengths has an impact on WebRTC quality, notably on bitrate variation. Hence, in our opinion channel utilization is not sufficient for evaluation of WebRTC performance and application layer statistics are essential for full quality evaluation.

De Cicco et al., in (**De Cicco et al. 2013a**) and in (**De Cicco et al. 2013b**), have performed detailed studies focusing on evaluation of the GCC algorithm.

First, in the study, (**De Cicco et al. 2013a**), they evaluate GCC in the presence of different concurrent traffic and variable bandwidth. They show that GCC, when by itself, can track available bandwidth and provide low queuing delays. However, the study also indicates that when sharing a bottleneck with TCP traffic, WebRTC is starved. Moreover, in the presence of another WebRTC traffic, also GCC controlled, the GCC behaviour is unpredictable.

Second, in the same study, (**De Cicco et al. 2013b**), the authors focus on the γ threshold that is a part of over-use detector, as explained in section 2.3.4. In the study, they evaluated WebRTC flows in case of different available bandwidth and in the presence of different concurrent traffic. They have proved that the γ threshold has a significant impact on the GCC sending rate dynamics. If the threshold is too large, the loss-based controller prevails over the delay-based controller, which leads to worse performance, i.e. greater delays and packet loss. In the case of a smaller threshold, GCC provides good channel utilization, but in case of concurrent flows, the delay-based controller prevails over the loss-based one and the WebRTC flows end up being starved. Thus, they have indicated that a dynamically changing threshold is needed.

This study was continued and has resulted in a proposal of mathematical model of

GCC, along with a design of adaptive γ threshold (**Carlucci et al. 2014**). This adaptive threshold by tracking the queuing delay variation can improve overall GCC performance.

Carlucci et al., in (**Carlucci et al. 2016a**), create a well detailed description of GCC and performed an analysis of GCC for different use cases, i.e. different available bandwidth and concurrent traffic. The study has shown that GCC effectively tracks available link capacity and in presented configurations provided fairness with TCP flows, i.e. it decreases but oscillates around 1Mbit/s. However, their measurements for long lived TCP flows were done for 100Mb/s link, so, given the limits of TCP, WebRTC could more easily oscillate around 1Mb/s, which is a small fraction of all available capacity, as it was also noticeable in our fiber access network evaluation.

When we were launching our PoC and evaluation measurements, the existing studies on GCC evaluation were using only one queuing mechanisms, i.e. droptail. There were studies focusing on different queuing mechanisms, but for different congestion control algorithms, i.e. NADA that remain simulation (**Zhu et al. 2015**) and SCReAM that is limited to OpenWebRTC implementations (**Swain 2015**).

However, we have found that in parallel to our work, there was a study done by Carlucci et al. and published recently, (**Carlucci et al. 2016c**), but after we have completed our evaluation.

Their study focuses on interplay between GCC and different queuing mechanisms, notably Codel, PIE, SFQ and FQ Codel. They analyse two scenarios: 1) isolated WebRTC flows for link capacity of 1 and 2 Mbit/s and 2) WebRTC flow in the presence of varying number of TCP flows and link capacity between 2 and 100Mbit/s, so that WebRTC rate was oscillating around 1Mbit/s.

In case of isolated WebRTC flow, the authors of the study have indicated that AQM solutions are not beneficial to GCC and degrade WebRTC performance, because of the introduction of packet loss.

In case of WebRTC, in the presence of concurrent TCP flows, the authors acknowledge that AQM solutions reduce queuing delay that is much lower than droptail. However, again they criticized them for introducing packet loss. They also indicate that FQ Codel and SFQ provide better performance than PIE and Codel. The study indicates that SFQ performed the best since it ensured delays as low as other queuing mechanisms, but by keeping lower packet loss.

This conclusion is based on the overall channel utilization, delay and packet loss used as metrics. The evolution of RTT and rate is shown only for the oversized link 100Mbit/s, so WebRTC is less impacted by congestion and the results are close to the ones we obtained for the fiber configuration. Without the whole set of measurements for different bandwidth and concurrent traffic, we cannot fully comment on these results and compare them with our assessment.

Nevertheless, in our opinion without metrics closely linked with perceived user quality it is difficult to judge the impact of a queuing mechanisms. With metrics like frame rate and frame width enriched by subjective opinion, we have shown that even if a mechanisms introduces packet loss, the impact on quality is often negligible. This was the case of FQ Codel that, even though introduced some packet loss, it kept good quality.

To conclude, there exist different studies on evaluation of WebRTC mechanisms. Thus there is an increased interest in queuing disciplines more adapted to real-time traffic, notably AQM solutions.

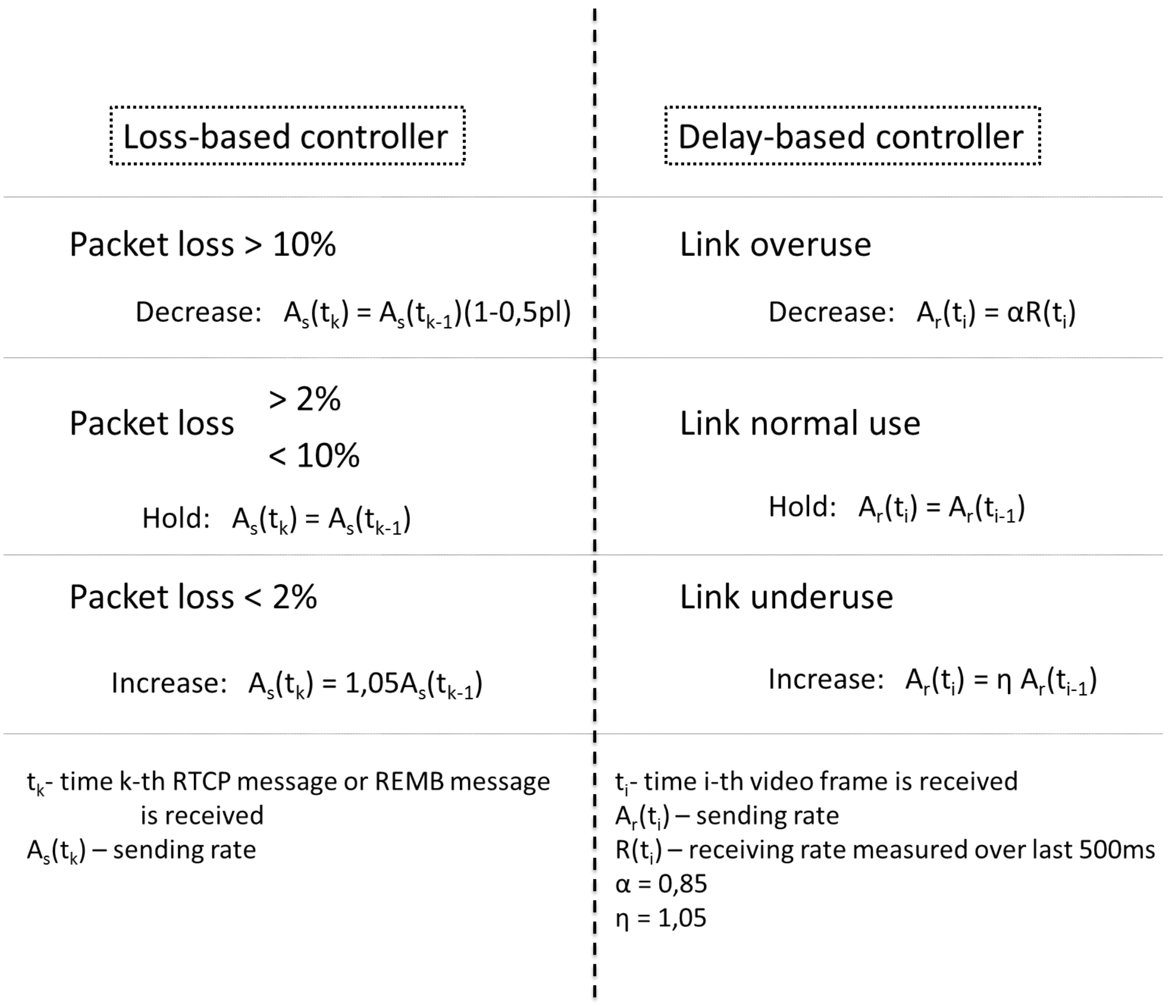
The mentioned studies used rate, packet loss, RTT and channel utilization as metrics for evaluation of WebRTC performance. However, we find that these metrics are not enough to get the full picture and that it is important to focus on metrics that are more linked with user perceived quality, e.g. frame rate and frame width.

7.5 Suggestions for GCC improvements

7.5.1 Suggestions of modifications to GCC algorithm

Thoughout this chapter, we have evaluated different network configurations in order to analyse their impact on the behaviour of WebRTC with a focus on GCC. Based on that assessment, we can suggest certain GCC improvements.

The GCC was explained in section 2.3.4. However, for better comprehension of proposed modifications, we present the summary of the equations used to calculate the sending rate by delay-based and loss-based controllers in figure 7.25.



Loss-based controller	Delay-based controller
Packet loss > 10% Decrease: $A_s(t_k) = A_s(t_{k-1})(1-0,5pl)$	Link overuse Decrease: $A_r(t_i) = \alpha R(t_i)$
Packet loss > 2% < 10% Hold: $A_s(t_k) = A_s(t_{k-1})$	Link normal use Hold: $A_r(t_i) = A_r(t_{i-1})$
Packet loss < 2% Increase: $A_s(t_k) = 1,05A_s(t_{k-1})$	Link underuse Increase: $A_r(t_i) = \eta\, A_r(t_{i-1})$
t_k- time k-th RTCP message or REMB message is received $A_s(t_k)$ – sending rate	t_i- time i-th video frame is received $A_r(t_i)$ – sending rate $R(t_i)$ – receiving rate measured over last 500ms $\alpha = 0,85$ $\eta = 1,05$

Figure 7.25 – Summary of the equations used to calculate sending rate by delay-based and loss-based controllers

While testing the droptail configuration, we have noticed that for very short buffers, notably 16kB, GCC had some performance issues, even without the presence of any concurrent traffic. It was discussed and our hypothesis was given in the section 7.2.2.

Simply put, if the buffer is too small, it does not introduce many changes into queuing delay. As a result, the delay-based controller does not effectively detect when a queue is building up. So when a queue builds up, the GCC does not react, which can lead to more packet loss.

In our opinion, if the GCC had information from networks about the configured buffer size, it could adapt its behaviour. For instance, loss-based controller could be more sensitive. It could start reacting to smaller packet losses and consequently decrease the sending rate earlier.

We could see that in case of certain configurations, WebRTC took a lot of time to recover after the congestion, i.e. when it was starved by a TCP flows. It was notably the case for droptail configuration for buffers over 32kB, as shown in the section 7.2.2.

As we explained in the same section, in our opinion it was caused by the reaction time of the delay-based controller. In brief, when the congestion is over and the queuing delay starts decreasing, the GCC does not increase its sending rate immediately. It goes to the *hold* state and waits for the buffers to stabilize. This takes more time in case of longer buffers.

Let us review the delay-based controller's behaviour, as explained in section 2.3.4. In essence, when the bottleneck queue starts to build-up, the link is considered as *overused* and the sending rate is decreased (*decrease* state). It causes buffer draining and consequently at a certain point the link becomes *underused* (*hold* state). The sending rate is kept constant, until the bottleneck buffer is emptied. At this point, the state of the link is considered *normal* and the sending rate is increased (*increase* state).

In our opinion, making the *hold* state shorter could shorten the recovery time. Moreover, instead of keeping the sending rate constant, when in *hold* state, it could be increased by a very small value, smaller than when in the *increase* state.

Another point is that GCC calculates the estimate of the one way delay gradient ($m(t_i)$) for a group of packets corresponding to a video frame. As explained in section 2.3.4, the estimate is obtained using measured one way delay variation ($d_m(t_i)$), calculated based on a time at which neighbouring video frames were sent and at which they were received, i.e.:

$d_m(t_i) = (t_i - t_{i-1}) - (T_i - T_{i-1})$

where:

t_i - the time of receiving the last packet of the i-th video frame,

T_i - the time of sending the first packet of the i-th video frame.

However, when WebRTC video traffic is starved, the video frame rate is very low. Hence, $m(t_i)$ is not calculated frequently enough.

Thus, in our opinion, it would be beneficial to calculate the estimate more often and make it less dependent from the video frame rate.

The coefficients used to increase or decrease the sending rate given in the specifications are arbitrary, but it would be interesting to test different values adapted to underlying queuing mechanisms.

If we assume that the WebRTC traffic benefits from queuing mechanisms aiming at reducing its delay, it could be possible to increase the α coefficient and consequently increase the sending rate faster. As a result, WebRTC video traffic could gain faster an appropriate share of the link.

Furthermore, WebRTC could act more aggressively when classful traffic management provides a configuration favorable to WebRTC.

Regarding the loss-based controller, in the presence of queuing mechanisms that systematically cause small packet loss, e.g. FQ Codel, the controller could be less sensitive to this packet loss.

7.5.2 Perspectives of interaction between networks and congestion control mechanisms

In the RMCAT congestion control requirements draft, (**Jesup & Sarker 2014**), it is specified that a congestion control algorithm should not require any specific support from the network and that it should leverage measured information about the flows, i.e. packet arrival time, packet loss, etc.

Nevertheless, we think that congestion control mechanisms could benefit from knowledge about the underlying network configuration. For instance, a congestion control could use the information about queuing mechanisms implemented in network equipment and consequently could adapt its behaviour - for example act more aggressively if there is a configuration favorable to WebRTC. Hence, it could be beneficial to work on richer interaction between networks and congestion control mechanisms. It would be also interesting to study the benefits of using the ECN marking.

In addition, NSPs would benefit from endpoint information, notably coming from a congestion control mechanism. By analyzing a large amount of statistics from application layer, NSPs could learn and adapt the configuration of their networks. In the first place this information could be useful for CPE reconfiguration, e.g. adapting HTB rate share or improving AQM settings. Our reasoning along with the test environment is a good starting point for linking dependence of CPE configuration, network and application layer measurements and their impact on perceived quality. Hence, it would be interesting to study what information an endpoint could provide, while preserving security and privacy aspects.

Another aspect would be to study what information the networks could provide to endpoints and also how an application could indicate its needs to network equipment.

In this context it is interesting to have a look at work on *Substrate Protocol for User Datagrams* (SPUD) prototype. During the IETF 92 meeting in March 2015, it was pointed out that there is a need for explicit cooperation with middleboxes, i.e. network equipment in the network path, provided by a UDP-based encapsulation protocol. SPUD is a prototype of a mechanism allowing grouping UDP packets together in a tube and enabling explicit cooperation between the endpoints and devices in the network path (**Hildebrand & Trammell 2015**). More precisely, devices along the path can provide path declarations to the endpoints, e.g. rate limit or latency information and the endpoints can indicate their needs, e.g. low loss or low latency (**Kuehlewind & Trammell 2016**), (**Trammell & Kuehlewind 2016**). Even though SPUD was not meant to be ever developed in the real life but is rather a prototype protocol used to test ideas and launch discussions, (**Chirgwin 2015**), it can be a starting point for further implementations of congestion control and network interactions.

7.6 Recommendations for coupling strategies

In the chapter 4 we have presented three strategies of loose coupling between WebRTC endpoint mechanisms and network layers for improving quality. In this chapter, we have evaluated identified traffic management solutions adapted to WebRTC. Thus, based on the obtained results, we position the identified traffic management solutions, in the context of the proposed loose coupling strategies, i.e. NSP driven, OTT driven and User/Enterprise driven.

7.6.1 NSP driven

Neutral solutions were proposed in the NSP driven strategy and were explained in the section 4.3.1. This concept assumes improving quality in general for all subscribers and aim at enabling optimal treatment of different types of traffic.

Target delay based queuing mechanisms could be positioned as a neutral solution, especially FQ Codel which has given promising results. It provides a better coexistence of real-time and non real-time traffic, by offering a balance between high link utilization and low delay. At the same time, it requires a minimal configuration and remains transparent to users and OTTs.

Given these satisfactory results and minimal configuration requirement, we believe that it would be a valuable improvement to access networks. AQM mechanisms could be used in residential solutions, notably for ADSL access networks with limited resources, as an intermediary solution before the fiber implementation. Although they do not require any complex configuration, their parameters would need to be adjusted to given network requirements. Thus, there is a need of certain effort to be able to implement them in the existing network equipment.

7.6.2 OTT driven

The TURN-based architecture with brokering was proposed in the OTT driven strategy and was explained in section 4.3.2. It enables improving quality for a given OTT service, so requires a per application treatment.

Rate and queue length based classful queuing mechanisms could be considered in this context, thanks to higher granularity obtained with HTB classification based on DSCP marking. Hence, HTB can provide a differentiated treatment only to applications with an appropriate marking.

Regarding scalability, CPE configuration needs to be adapted to foreseen traffic characteristics in order to ensure an appropriate resource sharing. Nevertheless, some assumptions can be made to simplify the CPE configuration. For instance, we have found that better results are obtained when higher priority is given even, if the guaranteed WebRTC rate is lower, i.e. HTB204 configuration. Hence, even if optimal rate for real-time communication is unknown, the appropriate configuration can be achieved by configuring an assumed minimum rate and giving higher priority to the real-time traffic.

Rate and queue length based classful queuing mechanisms can be used to improve network resources sharing and to provide differentiated treatment to identified traffic.

Furthermore, a monetization of this approach can be also considered since it requires NSP assistance and resources.

7.6.3 User/Enterprise driven

Enterprise policies solution was proposed in the User/Enterprise driven strategy and was explained in the section 4.3.3. It enables improving quality for a given user or application, so it also requires a certain granularity of flow identification.

Thus, rate and queue length based classful queuing mechanisms can be used, since HTB allows filtering and classification of traffic based on DSCP marking.

Traffic characteristics, limited to a given enterprise, can be analysed notably for prediction purposes. Consequently, they can be used for an appropriate HTB configuration.

7.6.4 Net neutrality aspects

The *Body of European Regulators for Electronic Communications* (BEREC) has been working on the *Guidelines on the Implementation by National Regulators of European Net Neutrality Rules*, (**BEREC 2016**), i.e. a set of rules for equal treatment of traffic. According to this document, a provider of Internet access service should not restrict the connectivity to any endpoint and should treat all traffic equally, i.e. without any discrimination or restriction, unless there is a reasonable justification of a differentiated treatment.

It is important to point out that end point congestion control and thus the GCC mechanism, is not covered by the regulations, but any network-internal mechanism, which assists these end point mechanisms needs to be considered in the regulation.

At the same time, the document points out that the regulations should not prevent providers from implementing reasonable traffic managements measures needed for the efficient use of network resources and for achieving optimal overall transmission quality. However, these measures should remain agnostic to the applications and be transparent, non-discriminating and not based on commercial agreements. They should take into account technical QoS requirements, e.g. jitter, packet loss or bandwidth.

Thus, the document acknowledges the pertinence of objective differentiation between different categories of traffic and as an example, it gives a category that consists of real-time traffic applications requiring low delay. At the same time, it highlights that traffic with similar QoS requirements should obtain similar treatment. Hence, there should be no discrimination between real-time traffic with similar characteristics, but coming from different applications.

In this context, AQM mechanisms do not cause any restrictions, since even though they enable differentiated treatment, like FQ Codel, they are application agnostic. They are based on the objective technical requirements, notably a queuing delay, so provide the improvement regardless the real-time traffic type.

The OTT driven approach would not be in line with BEREC's recommendations (**BEREC 2016**), since it is not application agnostic. Furthermore, the document states that there should be no traffic discrimination based on commercial consideration, so it would be difficult to justify a typically monetized solution.

Also based on the the BEREC's recommendations (**BEREC 2016**), it can be understood that in the case of User/Enterprise driven approach, the net neutrality rules do not apply since they are considered as networks that are not publicly available, e.g. internal corporate networks. The same applies to the access to *Virtual Private Networks* (VPN).

7.6.5 Discussion

Target delay based traffic management fits well with the NSP driven strategy. All things considered, the NSP driven approach would bring a real value to existing networks. In our opinion, the neutral solution based on AQM mechanisms is feasible and with certain adjustments, could be implemented for residential solutions.

Rate and queue length based classful traffic management is well adapted to OTT and User/Enterprise driven strategies.

The OTT driven strategy, with TURN-based architecture and brokering solution, can cause some difficulties given the net neutrality regulations and needs a more detailed analysis of these aspects from the legal point of view. Nevertheless, the concept could be reused in different solutions. For instance, this solution could be applicable for non-monetized use cases or for offering *specialized services* that is according to BEREC, (**BEREC 2016**), services other than Internet services and optimized to a specific content. They can be offered if the network capacity is sufficient and classic Internet services are not degraded. Furthermore, it could be also considered for public safety and emergency solutions. Also, the brokering could be used for load balancing purposes, to offer the best TURN server for a given WebRTC user.

The presented solution is dependent on the successful development of scalable brokering service. So far, a brokering prototype has been implemented under the reThink project, but it has been tested only for one NSP.

User/Enterprise driven strategy, notably the enterprise policies solution, is not under net neutrality regulations and is not concerned by the scalability problem, since it is limited to a given access network, so under one NSP's responsibility. As a result, it can be used in real world conditions. Furthermore, it was chosen to be further implemented by Orange under the reThink project.

7.7 Summary

In this chapter we have evaluated the impact of different uplink configurations on WebRTC communication and consequently on GCC algorithm.

We have evaluated two types of access networks: ADSL and fiber. We started by evaluation of the widespread droptail configuration that is based on the current IP network engineering. Later, we tested two identified approaches: 1) the target delay based queuing mechanisms and 2) the rate and queue length based classful queuing mechanisms. These approaches were compared with the results obtained from the droptail configuration.

ADSL results overview

We have focused mainly on test cases concerning the ADSL networks as it corresponds to a large part of French subscribers.

We have found that in the presence of long-lived TCP traffic, droptail has a significantly worse quality than other configurations. WebRTC video flow not only gets starved in the presence of long-lived TCP concurrent traffic, but additionally it takes time to recover, i.e. when the concurrent traffic disappears, GCC does not increase the sending rate straight away. Hence, GCC is not always able to compensate poor network conditions.

Furthermore, in droptail configuration it is important to choose and appropriate buffer size, since too large buffers can introduce excessive delays, whereas too short buffers can cause packet loss.

Concerning the target delay based configurations, in case of PIE configuration, the WebRTC flow is starved in the presence of TCP traffic, but starts recovering faster than in case of droptail configuration.

We have also shown that FQ Codel is able to provide a better resource sharing between different types of flows and cooperate well with GCC algorithms. Moreover, FQ Codel improves the WebRTC quality and avoids starvation. Even though it causes some packet loss, it provides good performance.

Overall, a well-chosen AQM mechanisms, can provide a significant improvement to WebRTC quality.

Rate and queue length based classful configurations have significantly improved the WebRTC performance. Certain configurations based on HTB enable obtaining the WebRTC quality close to the one without any concurrent traffic.

Furthermore, classful configurations provide much higher granularity. AQM improve the quality of any real-time traffic, whereas classful solutions based on HTB with DSCP marking, enable applying per flow treatment.

Fiber results overview

Regarding the fiber networks, the amount og available bandwidth guarantees good WebRTC performance, even in the presence of concurrent flows. Nonetheless, we have managed to congestion the fiber link and we have found that HTB and PIE are able to provide the initial quality of WebRTC. The improvement provided by FQ Codel was not as good, which could be linked to its algorithm complexity.

Anyhow, for current traffic characteristics, there is no negative impact on WebRTC quality in case of congestion in fiber networks. Nevertheless, the implementation of fiber for access networks needs big investments and time. Also traffic characteristics may change in the future, e.g. WebRTC may demand more resources in the future while increasing the video quality and proposing different use cases or WebRTC P2P CDN for sharing content may increase traffic in the access networks (**Levent-Levi 2015**). Furthermore, the congestion problems may be shifted to the backhaul networks, e.g. Vu-Brugier, in his study (**Vu-Brugier 2009**) highlights that FttH deployments cause congestion risk to move from access networks to the backhaul, since oversized links allow users to send bursts of data of tens of Mbit/s.

Recommendations

In our opinion, appropriate target delay based and rate and queue length based classful approaches are well adapted to real-time traffic, notably WebRTC. They also cooperate well with the GCC mechanism. They ensure fairer resource sharing and consequently avoid WebRTC flow starvation.

Nevertheless, there is a need of further testing, for instance for different network types, for downstream traffic and for various concurrent traffic.

As shown above, there exist different studies on evaluation of WebRTC and the GCC mechanism. On the whole, there is an increased interest in queuing disciplines more adapted to real-time traffic.

However, metrics used in related works are limited to rate, packet loss, RTT and channel utilization. In fact, these metrics are not able to give the full picture of the WebRTC performance. Our PoC adds analysis of frame rate and frame width that are useful in judging the perceived quality, e.g. in determining the impact of packet loss. The results can be confirmed by the subjective opinion. Additionally, we were able to quantify the impact on the concurrent TCP traffic by measuring the upload time.

Given the above points, we have suggested several improvements to the GCC algorithm, with a focus on interaction between networks and congestion control mechanisms. Overall, we think that the GCC would benefit from information about network equipment configuration from underlying networks.

Nevertheless, a further study in this context would be needed. It would be especially beneficial, to access the GCC code in Google Chrome. This would allow more precise analysis of loss-based and delay-based controller behaviour in the presence of different network configurations.

In the final analysis, we have positioned the identified traffic management solutions in the context of defined coupling strategies.

Target delay based queuing mechanisms could be applied in NSP driven approach. On the other hand, rate and queue length based classful queuing mechanisms, could be applied for OTT and User/Enterprise driven approach.

Chapter 8

Conclusion

Contents

8.1 Main contributions

The aim of this thesis was to study QoS management solutions for WebRTC while ensuring loose coupling between OTT and NSP approaches.

First of all, we identified the differences between OTT and NSP approaches. After reviewing the state of the art, that covered various aspects of NSP and OTT mechanisms, as well as standardisation efforts, we proposed strategies for improving OTT real-time communication service quality with NSP assistance.

These resulted in the following contributions.

Loose coupling strategies

We provided loose coupling strategies enabling cooperation between OTTs and NSPs. We have proposed three coupling strategies: 1) NSP driven, 2) OTT driven and 3) User/Enterprise driven. Each of these strategies focuses on different interactions between various actors. To validate this concept, for each of the strategies, we propose an implementation example with an architecture proposal.

The advantage of our loose coupling strategies is that they leverage the existing mechanisms. Thus, they do not impose any important changes to the WebRTC mechanisms and they do not enforce adapting WebRTC to NSP ecosystem. On the contrary, they are in line with web technologies and WebRTC standardisation efforts, since they demand little or no change to the existing WebRTC applications.

Traffic management adapted to WebRTC

We have chosen to verify the relevance of the loose coupling strategies in the context of the traffic management. Consequently, we identified the traffic management solutions adapted to WebRTC.

We have proposed two traffic management directions: 1) approach aiming at assuring lower queuing delays regardless the traffic and 2) approach aiming at isolating the sensitive traffic.

For each of these approaches we selected an appropriate queuing mechanism or a combination of mechanisms, which we later evaluated.

PoC and evaluation methodology

We have set a PoC, test environment and results evaluation methodology. They enabled verifying the impact of various network configurations on WebRTC and its congestion control mechanism, GCC.

The advantage of our evaluation methodology is that it enables an analysis of WebRTC and GCC performance on different levels. This is achieved thanks to detailed analysis of evolution of various KPIs throughout the communication. We also introduced an analysis of KPIs that are strongly linked with quality, e.g. frame rate and frame width. Additionally, we decided to collect KPIs from different levels, i.e. we provided network and application layer measurements enriched by subjective opinion.

Evaluation of traffic management solutions

We have performed an evaluation of proposed traffic management solutions and compared them with current best practices of network configuration.

We have shown that the wide spread droptail configuration, used in the best-effort delivery, is not always adapted to the WebRTC traffic. We have also proven that GCC mechanism does not always perform well in the presence of poor network quality.

Furthermore, we have shown that the identified traffic management, adapted to WebRTC, is beneficial. In brief, it ensures that the bandwidth share is fairer to the WebRTC traffic and it improves the overall quality.

Positioning of our approach in relation to the existing studies

We have positioned the performed evaluation in relation to the existing studies.

We have proven the importance of analysis of metrics collected from different levels. We have shown that metrics linked with user perceived quality are more adapted to give a full picture of WebRTC performance.

Suggestions of improvements to GCC

We have not only identified issues in WebRTC and GCC performance, but we also provided hypothesis explaining these encountered issues.

This analysis resulted in a proposal of improvements to the GCC mechanism. We have equally suggested research opportunities in assuring interaction between network equipment and GCC.

Validation of loose coupling strategies

The evaluated traffic management approaches were positioned in the context of the identified coupling strategies. Hence, they allowed validation of feasibility of loose coupling concepts.

Furthermore, the proposed solutions will continue to be implemented in Orange Labs and in the scope of reThink European Project and in the long term they may contribute to the implementation of a real service.

Reseach perspectives

Last but not least, this thesis contributed to identification of interesting research directions for further studies.

8.2 Research perspectives

8.2.1 Optimized network resource sharing

The study done in the context of this thesis is a good starting point for research concerning optimized sharing of network resources between different traffic types demanding different queuing mechanisms. Two possible research paths are presented in this section.

Latency Loss Tradeoff

In the state of the art, section 3.2.2, we have mentioned a *Latency Loss Tradeoff* (LTT) draft (You et al. 2016) that focuses on separating IP packets in two classes of services: (Lo) low-loss service and (La) low-latency service. The aim of this draft is to provide low-delay to real-time applications, while keeping the best-effort service.

The authors of the draft divide applications into two types:

- Applications that prefer to trade delay for loss, marked as Lo.

  It concerns applications that can accept higher delay, but are vulnerable to packet loss since it triggers a decrease in the sending rate and as a result, negatively impact the throughput.
- Applications that prefer to trade loss for delay, marked as La.

  It concerns interactive applications that need a short delay, but are less sensitive to packet loss.

Each application can choose which treatment it prefers, since the service is not supposed to prioritize any of the two classes and there is no advantage from cheating.

The LLT draft does not specify any particular method for achieving the defined behaviour. However, our bandwidth and queue length based approach has things in common with this draft and could be further tested within the LLT context.

In our study, we used HTB to separate the traffic into two classes: one adapted to WebRTC and the other one with typical best-effort parameters. We have tested the interaction between real-time and non real-time classes for two HTB configurations: HTB204 and HTB654.

Performed evaluation has shown that an approach like LLT could be valid and that a real-time traffic could benefit from treatment adapted to its characteristics without blocking the best-effort traffic. Nevertheless, it would be important to make sure that one class is not more beneficial than the other one.

Our study can be a starting point for further implementations of LLT concepts. Furthermore, in this context the bandwidth and queue length based mechanism would be in line with net neutrality aspects, as given in the BEREC's document (**BEREC 2016**), since it is not based on commercial agreements but leaves a decision to the application itself.

To further validate the concept, more tests should be performed for more traffic characteristics and network configurations, including the downstream scenarios. This would be helpful in defining the resource sharing strategy between the Lo and La classes.

Low Latency, Low Loss, Scalable Throughput Internet Service

The draft, (**Briscoe et al. 2016**), gives a problem statement for a new service that could replace the widespread best-effort access. This new Internet service is called *Low Latency, Low Loss, Scalable Throughput* (L4S) and aims at answering to low latency requirements of all applications. Furthermore, this solution aims at improving the average queuing delay without sacrificing the link utilization or causing packet loss. It recognises the fact that queuing is the major component of delay, but it also identifies TCP congestion control as the main cause of increased queuing delay and aims at fixing it.

The L4S service consists mainly of using the scalable congestion control, e.g. used most widely *Data Center TCP* (DCTCP), presented in (**Kühlewind et al. 2014**) and (**Briscoe et al. 2016**), that was designed to provide low queuing latency, low loss while assuring scalable throughput. DCTCP is a combination of congestion control and AQM. DCTCP principle is to use *Explicit Congestion Notification* (ECN) and to signal congestion even though the queue has only started to build up. This is different from the behaviour we know from TCP, which signals congestion only when a given buffer is already full and packets start getting dropped.

In order to implement DCTCP on a global scale, certain precautions need to be taken. DCTCP cannot be used together with TCP without being starved. Furthermore, it is not possible to completely replace the classic TCP mechanisms currently used. DCTCP also requires changes done to network buffers, senders and receivers. These changes are needed because of different network congestion signals interpretation used by DCTCP, since the congestion signals are triggered more often even if the queue is just building up.

So far, DCTCP was only implemented in the private networks and to allow its implementation on a global scale, there is a need of assuring the coexistence of new L4S service with the classic Internet service. This can enable backward compatibility and slow shifting to the new L4S service, as explained in (**De Schepper et al. 2015**).

The simplified schema of the L4S solution is given on the figure 8.1

Two types of sender can be used: scalable and classic. The principle of scalable congestion control is to trigger congestion signals rather than losses, i.e. DCTCP is a good example. Nevertheless, the draft indicates that there will be a need of scalable variants of existing congestion controls also for transport protocols other than TCP.

Classifier is required to separate the traffic into different queues, i.e. L4S specific and classic service specific.

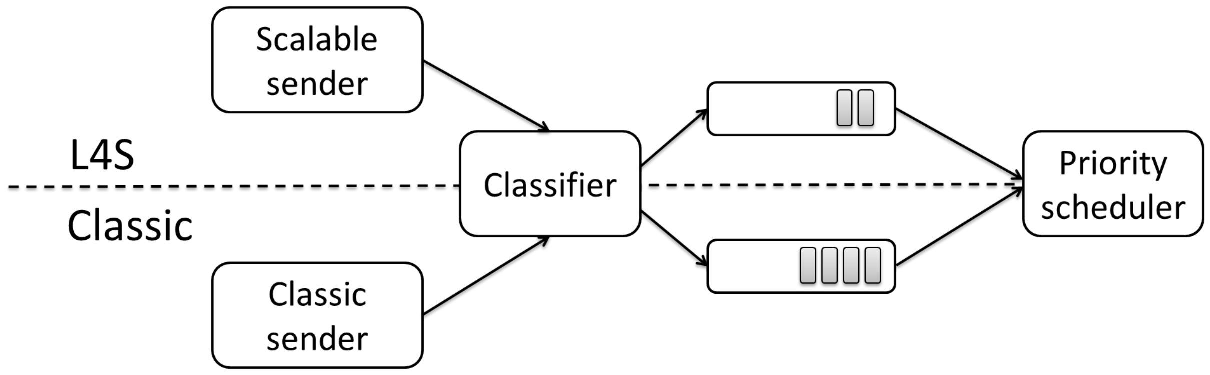


Figure 8.1 – Simplified schema of the L4S solution

Classic and L4S service needs to be isolated but at the same time they have to be able to adjust how they share resources, since in advance, it is difficult to predict the required capacity. For this purpose a draft suggests using a Dual Queue Coupled AQM solution, i.e. they use AQM across two queues. In order to ensure low latency of L4S queue, currently it gets a strict priority, so that classic queue can send only when the L4S queue is empty. At the same time, L4S queue is kept short so that classic queue is not completely blocked (**De Schepper et al. 2015**). The exact AQM mechanisms are not indicated in the draft (**Briscoe et al. 2016**), in order to encourage testing different ideas.

In our study we focused on differentiated treatment of real-time traffic type that was associated with a different class than the best-effort traffic. Furthermore, with bandwidth and queue length based mechanism, we tested different possibilities of fair resource sharing. Additionally, we have performed a detail study of AQM mechanisms and WebRTC congestion control, i.e. GCC.

These aspects can be used in order to pursue a study close to the L4S approach, either to ensure appropriate coexistence between L4S and classic services or to focus on real-time congestion control aspects. Thus in GCC a mechanisms analogical to DCTCP could be proposed and ECN concepts could be exploited.

8.2.2 Mobile cross layer aspects

As we have shown in the section 7.4, WebRTC congestion control are studied in the context of wireline networks. The existing studies are enriched by this thesis. However, it is equally necessary to focus on cellular and WiFi networks.

In this field there are two possible directions of studies:

- Over-the-top approach,
- Cross-layer approach.

The over-the-top approach, similar to principles of WebRTC congestion control, consists in taking actions based on measurements performed by the endpoints. There is a first analysis of impact of WiFi networks on GCC algorithms, notably in the presence of channel outages interpreted as congestions and negatively impacting the WebRTC communication (**Carlucci et al. 2016b**). However, more detailed studies for WiFi and cellular networks are needed.

The cross-layer approach allows interaction between a given application and the underlying radio resources. In this context, there are some ongoing studies.

CQIC

CQIC is a cross-layer congestion control for cellular networks that uses network estimates to provide optimal packet sending (**Lu et al. 2015**). This technology is not explicitly focusing on communication services, but rather on improving link utilization for CDNs.

CQIC creators highlight that cellular last hop causes a bottleneck. Its capacity is also difficult to predict since mobile data rates are very variable. Bandwidth oscillation together with delay caused by cellular links negatively impact TCP performances, so by degrading its throughput cause underutilization of cellular links.

CQIC is built with Google's *Quick UDP Internet Connection* (QUIC), i.e. transport protocol based on UDP, which has better performance than typical TCP thanks to providing faster connection establishment, along with better congestion control and loss recovery (**Chromium 2015**).

CQIC does not send probe traffic like TCP, but it leverages information provided by physical layer based on *Channel Quality Indicator* (CQI) extracted from radio layer traces. CQI come with CQI-to-rate mapping table that allows estimating the available bandwidth.

The receiving end device updates the estimation of available cellular bandwidth and sends it to the sending end device. Hence, the sender can adjust its sending rate.

The development of this solution would demand the implication of other actors, like device vendors and NSPs. So far, CQI are not directly visible to application layer and would require the assistance of chip makers and network operators to be deployed. Additionally, CQI-to-rate mapping changes between different implementations, so additional effort to systematize this mapping variation would be needed (**Lu et al. 2015**).

MTG

Mobile Throughput Guidance (MTG) (**Jain et al. 2015a**), is a mechanism designed to enable a cellular network to explicitly provide a near real-time information concerning the estimates of throughput to TCP servers. Thanks to this information about capacity at the radio link between the *Radio Access Network* (RAN) and the endpoint, the TCP server can adapt to changing conditions of underlying radio channel.

The implementation details are not given in the IETF drafts (**Jain et al. 2015a**), (**Jain et al. 2015b**), since they depend on the access network technology that can vary. Nevertheless, the mentioned drafts introduce a *Throughput Guidance Provider*. It is an element in the RAN that provides an appropriate throughput to a given TCP server based on an estimated link capacity and TCP flow information. As a result, the TCP server can use this information to optimize its behaviour and to improve the congestion control decisions.

So far MTG focuses only on TCP flows and on downlink. Is also requires a trustful relationship between a TCP server and a Throughput Guidance Provider.

5G network slicing

Network slicing, explained in (**Ericsson 2015**) and (**Elliott & Sharma 2016**), is a concept of logical networks aimed to be enabled within 5G systems. Indeed, 5G envisions new types of connectivity services that will be scalable and programmable.

The current *one-size-fits-all* network is not adapted to changing demands and various traffic characteristics. 5G will have to support a large number of use cases, requiring different network characteristics. Hence, 5G aims at assuring more flexible networks including logical slices able to meet specific demands. These logical network slices can be implemented on a common network infrastructure thanks to leveraging technologies like *Software-Defined Networking* (SDN) and *Network Function Virtualization* (NFV).

As a result, there can be a network slice dedicated to real-time communication allowing focusing at assuring low delays and at improving the overall end-to-end performance and quality.

Given the above points, there are mobile network aspects that need to be studied. Currently the studies focus on TCP traffic, however it would be interesting to apply these concepts in the WebRTC context.

8.3 Final thoughts

This thesis started as a general study on Telco APIs for OTT communication services. Throughout the period of three years, the subject has matured and focalized on specific research aspects. In effect, it resulted in a detailed study on loose coupling strategies enabling offering QoS management for WebRTC.

We have proven the pertinence of assuring loose coupling between NSP and OTT approaches. We have also shown that network assistance can be used in order to propose queuing mechanisms to OTT real-time communication services, notably by offering traffic management adapted to the WebRTC traffic. Moreover, we have proposed solutions that are beneficial to different actors, especially to the end users, as they improve the overall perceived quality.

The identified solutions and recommendations go beyond the studied aspects and can be a starting point in pursuing further work, notably on optimal network resource sharing, various interactions between OTT and NSPs, along with collaborative solutions.

Already, the study done in this thesis has been used in defining a new PhD subject: *A study and implementation of adaptive algorithms for real-time interactive applications in mobile access networks* that started in October 2016.

Bibliography

Alvestrand, H. (2016a). Overview: Real Time Protocols for Browser-based Applications, draft-ietf-rtcweb-overview-15. Retrieved from https://tools.ietf.org/html/draft-ietf-rtcweb-overview-15. Accessed : 2016-06-19.

Alvestrand, H. (2016b). RTCP message for Receiver Estimated Maximum Bitrate, raft-alvestrand-rmcat-remb-02. Retrieved from https://tools.ietf.org/html/draft-alvestrand-rmcat-remb-02. Accessed : 2016-07-06.

Alvestrand, H. (2016c). Transports for WebRTC, draft-ietf-rtcweb-transports-14. Retrieved from https://tools.ietf.org/html/draft-ietf-rtcweb-transports-14. Accessed : 2016-06-19.

Andreessen, M. (2011). The Wall Street Journal: Why Software Is Eating The World. Retrieved from http://www.wsj.com/articles/SB10001424053111903480904576512250915629460. Accessed : 2016-06-12.

Aouini, Z., Kortebi, A., & Ghamri-Doudane, Y. (2015). Traffic monitoring in home networks: enhancing diagnosis and performance tracking. In *14th International Wireless Communications and Mobile Computing Conference*, IWCMC 2015, Dubrovnik, Croatia, (pp. 545–550). IEEE.

APIExchange (publication date unknown). API Exchange for the GSMA by Apigee. Retrieved from http://www.gsma.com/personaldata/api-exchange. Accessed : 2016-06-19.

Arcep (2006). L'Autorité détermine les conditions relatives à l'interconnexion des réseaux et services de télécommunications ouverts au public. Retrieved from http://www.arcep.fr/index.php?id=6981. Accessed : 2016-10-25.

Arcep (2007). Droits et obligations des opérateurs et fournisseurs de services Guide juridique pour les opérateurs locaux et les collectivités. Retrieved from http://arcep.fr/uploads/tx_gspublication/guide-juridique-crip2007.pdf. Accessed : 2016-10-11.

Arcep (2015). Qualité du service fixe d'accès à internet, Version-test (bêta). Mesures de la qualité du service effectuées au 1er semestre 2015. Retrieved from http://www.arcep.fr/uploads/tx_gspublication/QoS_internet-rapport-S1_2015-nov2015.pdf. Accessed : 2016-09-28.

Arcep (2016). Observatoire des marchés des communications électroniques - Services fixes haut et très haut débit (suivi des ABONNEMENTS) - 2ème trimestre 2016 - résultats provisoires (publication le 8 septembre 2016). Retrieved from http://www.arcep.fr/index.php?id=13333. Accessed : 2016-09-13.

Arcep (publication date unknown). Observatoire/Qualité de service. Les résultats des mesures de la qualité de l'accès aux services fixes. Retrieved from http://www.arcep.fr/?id=10606. Accessed : 2016-10-25.

Baker, F. & Fairhurst, G. (2015). IETF Recommendations Regarding Active Queue Management. Retrieved from https://tools.ietf.org/html/rfc7567. Accessed : 2016-08-29.

Bankoski, J., Koleszar, J., Quillio, L., Salonen, J., Wilkins, P., & Xu, Y. (2011a). VP8 Data Format and Decoding Guide. Retrieved from https://tools.ietf.org/html/rfc6386. Accessed : 2016-06-27.

Bankoski, J., Wilkins, P., & Xu, Y. (2011b). Technical overview of VP8, an open source video codec for the web. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, ICME 2011, Barcelona, Spain, (pp. 1–6). IEEE.

Bauer, S., Clark, D. D., & Lehr, W. (2009). The evolution of internet congestion. In *Proceedings of the 39th Research Conference on Communication, Information and Internet Policy*, TPRC 2009, Arlington, VA, USA, (pp. 1–34). SSRN.

Baugher, M., McGrew, D., Naslund, M., Carrara, E., & Norrman, K. (2004). The Secure Real-time Transport Protocol (SRTP). Retrieved from https://www.ietf.org/rfc/rfc3711. Accessed : 2016-06-22.

Bécot, S., Bertin, E., Crom, J.-M., Frey, V., & Tuffin, S. (2015). Communication Services in the Web Era. How can Telco join the OTT hangout? In *Proceedings of the 18th International Conference on Intelligence in Next Generation Networks*, ICIN 2015, Paris, France, (pp. 208–215). IEEE.

BEREC (2016). BEREC Guidelines on the Implementation by National Regulators of European Net Neutrality Rules. Retrieved from http://berec.europa.eu/eng/document_register/subject_matter/berec/download/0/6160-berec-guidelines-on-the-implementation-b_0.pdf. Accessed : 2016-11-18.

Berezan, O., Yoo, M., & Christodoulidou, N. (2016). The impact of communication channels on communication style and information quality for hotel loyalty programs. In *Journal of Hospitality and Tourism Technology*, volume 7, (pp. 100–116). Emerald Group Publishing Limited.

Bergkvist, A., Burnett, D., Jennings, C., Narayanan, A., & Aboba, B. (2016). WebRTC 1.0: Real-time Communication Between Browsers, W3C Working Draft (13 September 2016). Retrieved from https://www.w3.org/TR/webrtc/. Accessed : 2016-06-19.

Bertin, E., Crespi, N., & L'Hostis, M. (2011). A few myths about telco and OTT models. In *Proceedings of the 15th International Conference on Intelligence in Next Generation Networks*, ICIN 2011, Berlin, Germany, (pp. 6–10). IEEE.

Bertin, E., Cubaud, S., Tuffin, S., Cazeaux, S., Crespi, N., & Beltran, V. (2013). WebRTC, the day after. In *Proceedings of the 17th International Conference on Intelligence in Next Generation Networks*, ICIN 2013, Venice, Italy, (pp. 46 – 52). IEEE.

Bouguen, Y., Hardouin, E., Maloberti, A., & Wolff, F.-X. (2012). *LTE et les réseaux 4G*. Paris, France: Editions Eyrolles.

Boyd, M. (2015). Programmable Web: Telco APIs Offer Huge Revenue if Carriers Can Handle the Disruption. Retrieved from http://www.programmableweb.com/news/telco-apis-offer-huge-revenue-if-carriers-can-handle-disruption/review/2015/09/21. Accessed : 2016-06-19.

Briscoe, B., Brunstrom, A., Petlund, A., Hayes, D., Ros, D., Tsang, J., Gjessing, S., Fairhurst, G., Griwodz, C., & Welzl, M. (2014). Reducing internet latency: A survey of techniques and their merits. In *IEEE Communications Surveys & Tutorials*, volume 18, (pp. 2149–2196). IEEE.

Briscoe, B., De Schepper, K., & Bagnulo Braun, M. (2016). Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Problem Statement, draft-briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem-02. Retrieved from https://tools.ietf.org/html/draft-briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem-02. Accessed : 2016-10-17.

Brodkin, J. (2014). Netflix packets being dropped every day because Verizon wants more money. Retrieved from http://arstechnica.com/information-technology/2014/02/netflix-packets-being-dropped-every-day-because-verizon-wants-more-money/. Accessed : 2016-07-27.

Brown, M. A. (2006). Traffic Control HOWTO, Version 1.0.2. Retrieved from http://tldp.org/HOWTO/Traffic-Control-HOWTO/. Accessed : 2016-08-30.

Burnett, D., Bergkvist, A., Jennings, C., Narayanan, A., & Aboba, B. (2016). Media Capture and Streams, W3C Candidate Recommendation (19 May 2016). Retrieved from https://www.w3.org/TR/mediacapture-streams/. Accessed : 2016-06-23.

Campedel, B. (2007). Quality of Service: The Basics. In *Management, Control and Evolution of IP Networks (ed G. Pujolle)*, (pp. 23–48)., London, UK. ISTE.

Carlucci, G., De Cicco, L., Holmer, S., & Mascolo, S. (2016a). Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, Klagenfurt, Austria, (pp.$\tilde{1}$3). ACM.

Carlucci, G., De Cicco, L., Holmer, S., & Mascolo, S. (2016b). Making Google Congestion Control robust over Wi-Fi networks using packet grouping. In *Proceedings of the 2016 Applied Networking Research Workshop*, ANRW '16, Berling, Germany, (pp. 74–80). ACM.

Carlucci, G., De Cicco, L., & Mascolo, S. (2014). Modelling and control for web real-time communication. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, CDC 2014, Los Angeles, CA, USA, (pp. 6824–6829). IEEE.

Carlucci, G., De Cicco, L., & Mascolo, S. (2016c). Controlling Queuing Delays for Real-Time Communication: The Interplay of E2E and AQM Algorithms. In *ACM SIGCOMM Computer Communication Review*. ACM.

Chappell, C. (2016). Mapping Open APIs With TM Forum. Retrieved from http://www.lightreading.com/mapping-open-apis-with-tm-forum/a/d-id/723770. Accessed : 2016-06-19.

Chirgwin, R. (2015). SPUD – The IETF's anti-snooping protocol that will never be used. Retrieved from http://www.theregister.co.uk/2015/07/30/understanding_spud_the_ietfs_burnafterreading_protocol/?page=1. Accessed : 2016-10-19.

Chong, H. M. & Matthews, H. S. (2004). Comparative analysis of traditional telephone and voice-over-Internet protocol (VoIP) systems. In *Proceedings of the IEEE International Symposium on Electronics and the Environment, 2004*, (pp. 106–111). IEEE.

Chromium (2015). Chromoim blog: A QUIC update on Google's experimental transport. Retrieved from http://blog.chromium.org/2015/04/a-quic-update-on-googles-experimental.html. Accessed : 2016-08-03.

Cisco (2012). Overview of Cisco Unified Communications Networking. Retrieved from http://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucm/srnd/9x/uc9x/ovnetwrk.html. Accessed : 2016-08-03.

Cisco (2016). White paper: Cisco VNI Forecast and Methodology, 2015-2020. Retrieved from http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. Accessed : 2016-11-21.

Clark, D. D., Bauer, S., Lehr, W., Claffy, K., Dhamdhere, A. D., Huffaker, B., & Luckie, M. (2014). Measurement and analysis of Internet interconnection and congestion. In *Proceedings of the 44th Research Conference on Communication, Information and Internet Policy*, TPRC 2009, Arlington, VA, USA, (pp. 1–16). SSRN.

Copeland, R., Bouabdallah, A., Javed, I., Paillet, E., Bécot, S., Janczukowicz, E., Corre, K., Crom, J.-M., Chainho, P., Beierle, F., Göndör, S., Luart, F., Al-Hezmi, A., Corici, A. A., Emmelmann, M., Pereira, R. L., Chaves, R., & Santos, N. (2015). Framework Architecture Definition. Retrieved from https://bscw.rethink-project.eu/pub/bscw.cgi/d29495-2/*/*/DOI-D2.1.html. Accessed : 2016-08-08.

Cox, C. (2014). *An introduction to LTE: LTE, LTE-advanced, SAE and 4G mobile communications, second edition*. Chichester, UK: John Wiley & Sons.

De Cicco, L., Carlucci, G., & Mascolo, S. (2013a). Experimental Investigation of the Google Congestion Control for Real-time Flows. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, FhMN '13, Hong Kong, China, (pp. 21–26). ACM.

De Cicco, L., Carlucci, G., & Mascolo, S. (2013b). Understanding the dynamic behaviour of the google congestion control for rtcweb. In *Proceedings of the 20th International Packet Video Workshop*, PV 2013, San Jose, CA, USA, (pp. 1–8). IEEE.

De Pessemier, T., Stevens, I., De Marez, L., Martens, L., & Joseph, W. (2015). Analysis of the quality of experience of a commercial voice-over-IP service. In *Multimedia Tools and Applications*, volume 74, (pp. 5873–5895). Springer.

De Schepper, K., Tsang, I., Bondarenko, O., & Briscoe, B. (2015). Data Centre to the Home: Ultra-Low Latency for All. Under submission, 2015.

Devera, M. (2002a). Hierachical token bucket theory. Retrieved from http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm. Accessed : 2016-08-30.

Devera, M. (2002b). HTB Linux queuing discipline manual - user guide. Retrieved from http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm. Accessed : 2016-08-30.

Ekstrom, H. (2009). QoS control in the 3GPP evolved packet system. In *IEEE Communications Magazine*, volume 47, (pp. 76–83). IEEE.

Elliott, J. & Sharma, S. (2016). Nokia, Dynamic End-to-end network slicing unlocks 5G possibilities. Retrieved from https://insight.nokia.com/dynamic-end-end-network-slicing-unlocks-5g-possibilities. Accessed : 2016-10-24.

Ericsson (2015). Ericsson White Paper: 5G Systems, Enabling Industry and society transformation. Retrieved from https://www.ericsson.com/res/docs/whitepapers/what-is-a-5g-system.pdf. Accessed : 2016-10-24.

Facebook (publication date unknown). Facebook for developers. Retrieved from https://developers.facebook.com/docs/apis-and-sdks. Accessed : 2016-06-19.

Floyd, S., Gummadi, R., Shenker, S., et al. (2001). Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. Technical report, ICSI.

Floyd, S. & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. In *IEEE/ACM Transactions on networking*, volume 1, (pp. 397–413). IEEE.

Gavois, S. (2014). Téléphonie mobile et restrictions imposées par les opérateurs, on fait le point. Retrieved from http://www.nextinpact.com/news/85186-telephonie-mobile-et-restrictions-imposees-par-operateurs-on-fait-point.htm. Accessed : 2016-06-10.

Gettys, J. (2011). Bufferbloat - Dark Buffers in the Internet. Retrieved from https://www.ietf.org/proceedings/80/slides/tsvarea-1.pdf. Accessed : 2016-09-28.

Gettys, J. & Nichols, K. (2011). Bufferbloat: Dark Buffers in the Internet. In *Magazine Queue - Virtualization*, volume 9, (pp. 40–54). ACM.

Google (publication date unknown). Google APIs. Retrieved from https://developers.google.com/apis-explorer/#p/. Accessed : 2016-06-19.

Gozdecki, J., Jajszczyk, A., & Stankiewicz, R. (2003). Quality of service terminology in IP networks. In *IEEE Communications Magazine*, volume 41, (pp. 153–159). IEEE.

Grigorescu, E., Kulatunga, C., & Fairhurst, G. (2013). Evaluation of the impact of packet drops due to AQM over capacity limited paths. In *Proceedings of the 21st IEEE International Conference on Network Protocols*, ICNP 2013, (pp. 1–6). IEEE.

Grigorik, I. (2013). *High Performance Browser Networking: What every web developer should know about networking and web performance*. Sebastopol, CA, USA: O'Reilly Media, Inc.

Ha, S., Rhee, I., & Xu, L. (2008). CUBIC: A New TCP-friendly High-speed TCP Variant. volume 42, (pp. 64–74). ACM.

Hart, C. (2015). webrtcH4cKS: Developing mobile WebRTC hybrid applications. Retrieved from https://webrtchacks.com/webrtc-hybrid-applications/. Accessed : 2016-11-21.

Hildebrand, J. & Trammell, B. (2015). Substrate Protocol for User Datagrams (SPUD) Prototype. draft-hildebrand-spud-prototype-03. Retrieved from https://tools.ietf.org/html/draft-hildebrand-spud-prototype-03. Accessed : 2016-10-19.

Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., & Dumazet, J. (2015). FlowQueue-Codel, draft-ietf-aqm-fq-codel-02. Retrieved from https://tools.ietf.org/html/draft-ietf-aqm-fq-codel. Accessed : 2016-08-28.

Holmer, S., Lundin, H., Carlucci, G., De Cicco, L., & Mascolo, S. (2015). A Google Congestion Control Algorithm for Real-Time Communication, draft-ietf-rmcat-gcc-01. Retrieved from https://tools.ietf.org/html/draft-ietf-rmcat-gcc-01. Accessed : 2016-07-01.

Holmer, S., Shemer, M., & Paniconi, M. (2013). Handling packet loss in WebRTC. In *Proceedings of the 20th IEEE International Conference on Image Processing*, ICIP 2013, Melbourne, Australia. IEEE.

Huawei (publication date unknown). Thriving in the new ecosystem: How Telecom Operators can secure a bigger share of the evolving value chain. Retrieved from http://www.huawei.com/ilink/en/solutions/arpu-up/HW_345562. Accessed : 2016-10-26.

Hubert, B. (publication date unknown). Linux Advanced Routing - Traffic Control HOWTO. Chapter 9. Queueing Disciplines for Bandwidth Management - 9.5. Classful Queueing Disciplines. Retrieved from http://lartc.org/howto/lartc.qdisc.classful.html. Accessed : 2016-09-29.

ITU-T (2001). G.1010: End-user multimedia QoS categories. In *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS, Quality of service and performance*, November, 2001. ITU-T.

ITU-T (2003). G.114: One-way transmission time. In *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS International telephone connections and circuits – General Recommendations on the transmission quality for an entire international telephone connection*, May, 2003. ITU-T.

ITU-T (2015). G.107: The E-model, a computational model for use in transmission planning. In *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS International telephone connections and circuits – Transmission planning and the E-model*, June, 2015. ITU-T.

Ivancic, D., Hadjina, N., & Basch, D. (2005). Analysis of precision of the HTB packet scheduler. In *Proceedings of the 18th International Conference on Applied Electromagnetics and Communications*, ICECom 2005, Dubrovnik, Croatia, (pp. 1–4). IEEE.

Jain, A., Terzis, A., Flinck, H., Sprecher, N., Arunachalam, S., & Smith, K. (2015a). Mobile Throughput Guidance Inband Signaling Protocol. draft-flinck-mobile-throughput-guidance-03.txt. Retrieved from https://tools.ietf.org/html/draft-flinck-mobile-throughput-guidance-03. Accessed : 2016-10-21.

Jain, A., Terzis, A., Sprecher, N., Arunachalam, S., Smith, K., & Klas, G. (2015b). Requirements and reference architecture for Mobile Throughput Guidance Exposure. draft-sprecher-mobile-tg-exposure-req-arch-01.txt. Retrieved from https://tools.ietf.org/html/draft-sprecher-mobile-tg-exposure-req-arch-01. Accessed : 2016-10-21.

Janczukowicz, E., Braud, A., Tuffin, S., Fromentoux, G., Bouabdallah, A., & Bonnin, J.-M. (2015). Specialized network services for WebRTC: TURN-based architecture proposal. In *Proceedings of the 1st Workshop on All-Web Real-Time Systems*, AWeS '15, Bordeaux, France, (pp. 1–6). ACM.

Janczukowicz, E., Tuffin, S., Braud, A., Bouabdallah, A., Fromentoux, G., & Bonnin, J.-M. (2014). Approaches for Offering QoS and Specialized Traffic Treatment for WebRTC. In *Advances in Communication Networking, 20th EUNICE/IFIP EG 6.2, 6.6 International Workshop*, EUNICE, Rennes, France, (pp. 59–69). Springer.

Jesup, R., Loreto, S., & Tuexen, M. (2015). WebRTC Data Channels, draft-ietf-rtcweb-data-channel-13. Retrieved from https://tools.ietf.org/html/draft-ietf-rtcweb-data-channel-13. Accessed : 2016-06-23.

Jesup, R. & Sarker, Z. (2014). Congestion Control Requirements for Interactive Real-Time Media, draft-ietf-rmcat-cc-requirements-09. Retrieved from https://tools.ietf.org/html/draft-ietf-rmcat-cc-requirements-09. Accessed : 2016-06-30.

Johansson, I. & Sarker, Z. (2016). Self-Clocked Rate Adaptation for Multimedia, draft-ietf-rmcat-scream-cc-05. Retrieved from https://tools.ietf.org/html/draft-ietf-rmcat-scream-cc-05. Accessed : 2016-07-01.

Jones, P., Dhekisan, S., Jennings, C., & Druta, D. (2016). DSCP Packet Markings for WebRTC QoS, draft-ietf-tsvwg-rtcweb-qos-18. Retrieved from https://tools.ietf.org/html/draft-ietf-tsvwg-rtcweb-qos-18. Accessed : 2016-08-31.

Khademi, N., Ros, D., & Welzl, M. (2013). The new AQM kids on the block: Much ado about nothing? In *Technical Report 434*. Universitetet i Oslo.

Khan, M. (publication date unknown). WebRTCPedia! the Encyclopedia! Retrieved from https://www.webrtc-experiment.com/webrtcpedia/. Accessed : 2016-08-16.

Kim, K., Niculescu, D., & Hong, S. (2009). Coexistence of VoIP and TCP in wireless multihop networks. In *IEEE Communications Magazine*, volume 47, (pp. 75–81). IEEE.

Kreibich, C., Weaver, N., Nechaev, B., & Paxson, V. (2010). Netalyzr: illuminating the edge network. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, Melbourne, Australia, (pp. 246–259). ACM.

Kuehlewind, M. & Trammell, B. (2016). Use Cases for a Substrate Protocol for User Datagrams (SPUD). draft-kuehlewind-spud-use-cases-01. Retrieved from https:

//tools.ietf.org/html/draft-kuehlewind-spud-use-cases-01. Accessed : 2016-10-19.

Kühlewind, M., Wagner, D. P., Espinosa, J. M. R., & Briscoe, B. (2014). Using data center TCP (DCTCP) in the internet. In *2014 IEEE Globecom Workshops*, GC Wkshps, Austin, TX, USA, (pp. 583–588). IEEE.

Kuhn, N., Lochin, E., & Mehani, O. (2014). Revisiting old friends: is CoDel really achieving what RED cannot? In *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity sharing workshop*, CSWS '14, Chicago, IL, USA, (pp. 3–8). ACM.

Kulatunga, C., Kuhn, N., Fairhurst, G., & Ros, D. (2015). Tackling Bufferbloat in capacity-limited networks. In *Proceedings of European Conference on Networks and Communications*, EuCNC 2015, Paris, France, (pp. 381–385). IEEE.

Lamouri, M. (2014). The Network Information API, W3C Working Group Note (10 April 2014). Retrieved from https://www.w3.org/TR/netinfo-api/. Accessed : 2016-08-04.

Leavitt, N. (2010). Network-usage changes push internet traffic to the edge. In *Computer*, volume 10, (pp. 13–15). IEEE.

Levent-Levi, T. (2013). Would You Use a Telco API? Retrieved from https://bloggeek.me/use-telco-api/. Accessed : 2016-06-19.

Levent-Levi, T. (2014). WebRTC JTBD: Reducing Barriers of Entry. Retrieved from https://bloggeek.me/webrtc-jtbd-barrier-of-entry/. Accessed : 2016-10-26.

Levent-Levi, T. (2015). WebRTC P2P CDN: Where are the Use Cases? Retrieved from https://bloggeek.me/webrtc-p2p-cdn-use-cases/. Accessed : 2016-09-11.

Lu, F., Du, H., Jain, A., Voelker, G. M., Snoeren, A. C., & Terzis, A. (2015). CQIC: Revisiting cross-layer congestion control for cellular networks. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, HotMobile '15, Santa Fe, NM, USA, (pp. 45–50). ACM.

Lundqvist, B. (2011). *QOS-Enabled Networks Tools & Foundations*. Chichester, UK: John Wiley & Sons.

Mahy, R., Matthews, P., & Rosenberg, J. (2010). Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). Retrieved from https://tools.ietf.org/html/rfc5766. Accessed : 2016-06-17.

Martinsen, P., Andersen, T., Salgueiro, G., & Petit-Huguenin, M. (2015). Traversal Using Relays around NAT (TURN) Bandwidth Probe, draft-martinsen-tram-turnbandwidthprobe-00. Retrieved from https://tools.ietf.org/html/draft-martinsen-tram-turnbandwidthprobe-00. Accessed : 2016-06-26.

Martinsen, P., Reddy, T., Wing, D., & Singh, V. (2016). Measurement of Round Trip Time and Fractional Loss Using STUN, draft-ietf-tram-stun-path-data-04. Retrieved from https://tools.ietf.org/html/draft-ietf-tram-stun-path-data-04. Accessed : 2016-06-26.

McKenney, P. E. (1990). Stochastic fairness queueing. In *Proceedings of the 9th Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration.*, INFOCOM'90, San Francisco, CA, USA, (pp. 733–740). IEEE.

Meddeb, A. (2010). Internet QoS: pieces of the puzzle. In *IEEE Communications Magazine*, volume 48, (pp. 86–94). IEEE.

Minerva, R. (2013). *Will the Telco survive to an ever changing world ? Technical considerations leading to disruptive scenarios.* Phd dissertation, Institut National des Télécommunications.

Nelson, E. & van den Dam, R. (2015). IBM Institute for Business Value. Telco 2015. Five telling years, four future scenarios. Retrieved from http://www-05.ibm.com/cz/gbs/study/pdf/GBE03304USEN.PDF. Accessed : 2016-10-25.

Newman, M. (2016). Telco APIs: operators are trying again. Retrieved from http://www.telecomasia.net/content/telco-apis-operators-are-trying-again. Accessed : 2016-06-19.

Nichols, K. & Jacobson, V. (2012). Controlling queue delay. In *Magazine Communications of the ACM*, volume 55, (pp. 42–50). ACM.

Nichols, K., Jacobson, V., McGregor, A., & Iyengar, J. (2016). Controlled Delay Active Queue Management, draft-ietf-aqm-codel-04. Retrieved from https://www.ietf.org/id/draft-ietf-aqm-codel-04.txt. Accessed : 2016-08-26.

O'Connell, J. (2013). An IT Perspective on Standards, Service Architectures and Platforms. In *Evolution of Telecommunication Services. The Convergence of Telecom and Internet: Technologies and Ecosystems*, (pp. 118–137). Springer.

Offremedia.com (2015). Médiamétrie quantifie et profile les foyers équipés "4 écrans". Retrieved from http://www.journaldunet.com/ebusiness/le-net/foyers-e-quipe-s-4-e-crans-0215.shtml. Accessed : 2016-07-27.

OMA (publication date unknown). OMA APIs Standardize Access to Unique Resources within Operator Networks. Retrieved from http://technical.openmobilealliance.org/Technical/api-information. Accessed : 2016-06-19.

Opus (publication date unknown). Opus Interactive Audio Codec. Retrieved from https://www.opus-codec.org/. Accessed : 2016-06-27.

Orange (publication date unknowna). Developer APIs. Retrieved from https://developer.orange.com/apis. Accessed : 2016-06-19.

Orange (publication date unknownb). L'internet mobile, à quoi ça sert ? Retrieved from http://reseaux.orange.fr/internet-mobile-3Go. Accessed : 2016-06-10.

Pan, R., Natarajan, P., Baker, F., & White, G. (2016). PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem, draft-ietf-aqm-pie-09. Retrieved from https://tools.ietf.org/html/draft-ietf-aqm-pie-09. Accessed : 2016-08-29.

Pan, R., Natarajan, P., Piglione, C., Prabhu, M. S., Subramanian, V., Baker, F., & VerSteeg, B. (2013). PIE: A lightweight control scheme to address the bufferbloat problem. In *Proceedings of the 14th International Conference on High Performance Switching and Routing*, HPSR, Taipei, Taiwan, (pp. 148–155). IEEE.

Pautasso, C. & Wilde, E. (2009). Why is the Web Loosely Coupled?: A Multi-faceted Metric for Service Design. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, Madrid, Spain, (pp. 911–920). ACM.

Perkins, C. & Singh, V. (2016). Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions, draft-ietf-avtcore-rtp-circuit-breakers-18. Retrieved from https://tools.ietf.org/html/draft-ietf-avtcore-rtp-circuit-breakers-18. Accessed : 2016-10-28.

Perkins, C., Westerlund, M., & Ott, J. (2016). Web Real-Time Communication (WebRTC): Media Transport and Use of RTP, draft-ietf-rtcweb-rtp-usage-26. Retrieved from https://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage-26. Accessed : 2016-06-22.

Pfeiffer, S. (2016). WebRTC predictions for 2016. Retrieved from http://gingertech.net/2014/03/19/apprtc-googles-webrtc-test-app-and-its-parameters/. Accessed : 2016-09-15.

Poese, I., Frank, B., Smaragdakis, G., Uhlig, S., Feldmann, A., & Maggs, B. (2012). Enabling content-aware traffic engineering. In *ACM SIGCOMM Computer Communication Review*, volume 42, (pp. 21–28). ACM.

Quayle, A. (2012). Please, No More Telco API Standards, Please.... Retrieved from http://alanquayle.com/2012/07/please-no-more-telco-api-stand/. Accessed : 2016-06-19.

Ramahandry, T., Bonneau, V., & Nakajima, S. (2012). Telco APIs: Service delivery platforms, an enabler of new business models. IDATE Research. ISBN 978-2-84822-322-3.

Ramakrishnan, K., Floyd, S., & Black, D. (2001). The Addition of Explicit Congestion Notification (ECN) to IP. Retrieved from https://tools.ietf.org/html/rfc3168. Accessed : 2016-09-28.

Reddy, T., Patil, P., Ravindranath, R., & Uberti, J. (2015). Session Traversal Utilities for NAT (STUN) Extension for Third-Party Authorization. Retrieved from https://tools.ietf.org/html/rfc7635. Accessed : 2016-08-31.

Rescorla, E. & Modadugu, N. (2012). Datagram Transport Layer Security Version 1.2. Retrieved from https://tools.ietf.org/html/rfc6347. Accessed : 2016-06-23.

Roach, A. (2016). WebRTC Video Processing and Codec Requirements. Retrieved from https://tools.ietf.org/html/rfc7742. Accessed : 2016-06-27.

Rosenberg, J. (2010). Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. Retrieved from https://tools.ietf.org/html/rfc5245. Accessed : 2016-06-17.

Rosenberg, J., Mahy, R., Matthews, P., & Wing, D. (2008). Session Traversal Utilities for NAT (STUN). Retrieved from https://tools.ietf.org/html/rfc5389. Accessed : 2016-06-17.

Schön, O., Zimmermann, P., & KVJ, S. (2011). Capgemini, Innovating the Telco Business Model Drivers and Emerging Trends Telecom & Media Insights. Issue 64. Retrieved from https://www.de.capgemini.com/resource-file-access/resource/pdf/Innovating_the_Telco_Business_Model_Drivers_and_Emerging_Trends.pdf. Accessed : 2016-10-26.

Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). RTP: A Transport Protocol for Real-Time Applications. Retrieved from https://www.ietf.org/rfc/rfc3550. Accessed : 2016-06-22.

Siemon, D. (2013). Queueing in the Linux Network Stack. Retrieved from https://www.coverfire.com/articles/queueing-in-the-linux-network-stack/. Accessed : 2016-09-14.

Sime, A. (2016). IoT and WebRTC – presentation at WebRTC Boston. Retrieved from https://webrtc.ventures/2016/03/iot-and-webrtc-presentation/. Accessed : 2016-11-21.

Singh, V., Lozano, A. A., & Ott, J. (2013). Performance analysis of receive-side real-time congestion control for WebRTC. In *Proceedings of the 20th International Packet Video Workshop*, PV 2013, San Jose, CA, USA, (pp. 1–8). IEEE.

Skype (2016). Overview of Skype for Business SDN Interface. Retrieved from https://msdn.microsoft.com/en-us/library/dn785191(v=office.16).aspx. Accessed : 2016-08-03.

Statovci-Halimi, B. & Franzl, G. (2013). QoS differentiation and Internet neutrality. In *Telecommunication Systems*, volume 52, (pp. 1605–1614). Springer.

Stewart, R. (2007). Stream Control Transmission Protocol. Retrieved from https://tools.ietf.org/html/rfc4960. Accessed : 2016-06-23.

Swain, R. (2015). Your OpenWebRTC calls are now congestion controlled. Retrieved from http://www.openwebrtc.org/blog/2015/11/19/your-openwebrtc-calls-are-now-congestion-controlled. Accessed : 2016-11-20.

Teitelbaum, B. & Shalunov, S. (2002). Why premium IP service has not deployed (and probably never will). In *Internet2 QoS Working Group Informational Document*.

Thomas, D. (2013). European telecoms revenue fall accelerates, Spending on faster next generation networks at risk. Retrieved from https://www.ft.com/content/51090f7e-538a-11e3-9250-00144feabdc0. Accessed : 2016-10-26.

TMForum (publication date unknown). tmforum open:apis. Retrieved from https://www.tmforum.org/open-apis/. Accessed : 2016-06-19.

Trammell, B. & Kuehlewind, M. (2016). Requirements for the design of a Substrate Protocol for User Datagrams (SPUD). draft-trammell-spud-req-04. Retrieved from https://datatracker.ietf.org/doc/draft-trammell-spud-req/. Accessed : 2016-10-19.

Ubuntu (publication date unknown). Ubuntu manuals: tc - show / manipulate traffic control settings. Retrieved from http://manpages.ubuntu.com/manpages/wily/man8/tc.8.html. Accessed : 2016-08-31.

Vakulenko, M. (2013). Vision Mobile Blog: Asymmetric business models and the true value of innovation. Retrieved from http://www.visionmobile.com/blog/2013/02/asymmetric-business-models-and-the-true-value-of-innovation/. Accessed : 2016-06-19.

Valin, J. & Bran, C. (2016). WebRTC Audio Codec and Processing Requirements. Retrieved from https://tools.ietf.org/html/rfc7874. Accessed : 2016-06-27.

Valin, J., Vos, K., & Terriberry, T. (2012). Definition of the Opus Audio Codec. Retrieved from https://tools.ietf.org/html/rfc6716. Accessed : 2016-06-27.

Vu-Brugier, G. (2009). Analysis of the impact of early fiber access deployment on residential internet traffic. In *Proceedings of the 21st International Teletraffic Congress*, ITC 21 2009, Paris, France, (pp. 1–8). IEEE.

W3C (2014a). Mailing List: API for network optimization. Retrieved from https://lists.w3.org/Archives/Public/public-web-mobile/2014Dec/0009.html. Accessed : 2016-08-04.

W3C (2014b). Mailing List: [W3C WebMob] Data Info API. Retrieved from https://lists.w3.org/Archives/Public/public-web-mobile/2014Oct/0004.html. Accessed : 2016-08-04.

W3C (2014c). Telcos at W3C (TPAC 2014 breakout session). Retrieved from https://www.w3.org/2014/10/29-telcos-minutes.html. Accessed : 2016-08-04.

W3C (2016). Identifiers for WebRTC's Statistics API, W3C Working Draft (27 May 2016). Retrieved from https://www.w3.org/TR/webrtc-stats/. Accessed : 2016-09-16.

Wallace, K. (2004). Cisco IP Telephony Flash Cards: Weighted Random Early Detection (WRED). Retrieved from http://www.ciscopress.com/articles/article.asp?p=352991&seqNum=7. Accessed : 2016-08-31.

Welch, G. & Bishop, G. (2006). An introduction to the kalman filter. Department of Computer Science, University of North Carolina. University of North Carolina at Chapel Hill, Department of Computer Science, unpublished manuscript.

Wing, D., Reddy, T., Williams, B., & Ravindranath, R. (2014). TURN extension to convey flow characteristics, draft-wing-tsvwg-turn-flowdata-01. Retrieved from https://tools.ietf.org/html/draft-wing-tsvwg-turn-flowdata-01. Accessed : 2016-07-01.

WirelessConnect (publication year unknown). Bandwidth Control. Retrieved from http://wirelessconnect.eu/articles/bandwidth%20_control. Accessed : 2016-08-30.

You, J., Welzl, M., Trammell, B., M., K., & Smith, K. (2016). Latency Loss Tradeoff PHB Group. Retrieved from https://tools.ietf.org/html/draft-you-tsvwg-latency-loss-tradeoff-00. Accessed : 2016-08-03.

Zhu, X., Pan, P., Ramalho, M., Mena, S., Jones, P., Fu, J., D'Aronco, S., & Ganzhorn, C. (2016). NADA: A Unified Congestion Control Scheme for Real-Time Media, draft-ietf-rmcat-nada-02. Retrieved from https://tools.ietf.org/html/draft-ietf-rmcat-nada-02. Accessed : 2016-07-01.

Zhu, X., Pan, R., Ramalho, M., Mena de la Cruz, S., Ganzhorn, C., Jones, P., & D'Aronco, S. (2015). NADA Interactions with AQMs. Retrieved from https://www.ietf.org/proceedings/92/slides/slides-92-rmcat-5.pdf. Accessed : 2016-03-17.

Résumé

Depuis plusieurs années, on observe une multiplication des services de communication en temps réel de type Over-The-Top (OTT). Ces solutions utilisent l'Internet « best-effort » et s'adaptent aux fluctuations du réseau. Néanmoins, il est discutable que l'approche OTT soit suffisante pour fournir une qualité de service de communication acceptable quelles que soient les conditions réseaux. Dès lors, est-il possible d'utiliser l'assistance réseau pour améliorer la qualité de service des solutions OTT ?

Pour traiter cette question, nous étudions tout d'abord les solutions OTT, et particulièrement la technologie WebRTC. Nous identifions trois stratégies de couplage lâche qui permettent de tirer parti des mécanismes réseaux pour améliorer la qualité de service des solutions OTT.

Nous vérifions la pertinence de ces stratégies dans le contexte de la gestion du trafic. On identifie deux approches de gestion du trafic adaptées à WebRTC : 1) qui assure des délais d'attente courts quel que soit le trafic ou 2) qui isole le trafic sensible.

On évalue ces solutions et leur impact sur WebRTC, pour les réseaux d'accès filaire (uplink, ADSL et fibre). Les résultats obtenus montrent que les pratiques actuelles de gestion du trafic ne sont pas adaptées au trafic WebRTC. De plus, les solutions proposées assurent plus d'équité entre le trafic WebRTC et TCP et elles permettent d'éviter que le trafic WebRTC soit désavantagé et elles améliorent la qualité de communication.

Enfin, ces solutions de la gestion du trafic sont positionnées dans le contexte des stratégies de couplage proposées. A partir de là, on fournit des recommandations pour améliorer la qualité WebRTC avec l'assistance du NSP.

Mots clés : WebRTC, Voix sur IP, Qualité de Service, Visioconférences, Contrôle de congestion, Analyse de réseau, Service de communication, Qualité d'expérience.

Abstract

The number of real-time Over-The-Top (OTT) communication services has increased in the recent years. OTT solutions use the best-effort Internet delivery and rely on mechanisms built into the endpoints to adapt to underlying network fluctuations. Nevertheless, it is questionable if this approach is enough to provide acceptable quality of communication regardless the network conditions. Therefore, can network assistance be used to improve the quality of OTT real-time communication services?

To address this question, we study OTT solutions with a focus on WebRTC. We identify three loose coupling strategies that leverage network mechanisms for improving OTT communication services quality.

We verify the pertinence of these coupling strategies in the context of traffic management. We identify two approaches of traffic management solutions adapted to WebRTC traffic: 1) aiming at assuring lower queuing delays regardless the traffic or 2) isolating the sensitive traffic.

We study the impact of identified traffic management solutions on WebRTC for wireline access networks (uplink, ADSL and fiber). The obtained results show that current Internet engineering practices are not well adapted to the WebRTC traffic, but are optimized for TCP traffic. Furthermore, the proposed solutions ensure more fairness between WebRTC and TCP flows and consequently enable avoiding WebRTC traffic starvation and improve the overall quality of the communication.

In the final analysis, the evaluated traffic management solutions are positioned in the context of identified coupling strategies. Based on this assessment, we provide recommendations of improving WebRTC quality with the assistance of NSP.

Keywords : WebRTC, VoIP, Quality of Service (QoS), Videoconference, Congestion control, Network analysis, Communication service, Quality of Experience (QoE).

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

N° d'ordre : 2017IMTA0010

IMT Atlantique Bretagne-Pays de la Loire - www.imt-atlantique.fr

Campus de Brest	Campus de Nantes	Campus de Rennes
Technopôle Brest-Iroise	4, rue Alfred Kastler - La Chantrerie	2, rue de la Châtaigneraie
CS 83818	CS 20722	CS 17607
29238 Brest Cedex 03	44307 Nantes Cedex 3	35576 Cesson Sévigné Cedex
T +33 (0)2 29 00 11 11	T +33 (0)2 51 85 81 00	T +33 (0)2 99 12 70 00
F +33 (0)2 29 00 10 00	F +33 (0)2 51 85 81 99	F +33 (0)2 99 12 70 08