Co-funded by the
Erasmus+ Programme
of the European Union

Resource, Data and Application Management for Cloud Federations and Multi-Clouds

VAMIS XHAGJIKA

Doctoral Thesis in Distributed Computing
Universitat Politècnica de Catalunya
Departament d'Arquitectura de Computadors,
Barcelona, Spain 2017
and
Doctoral Thesis in Information and Communication Technology
KTH Royal Institute of Technology
School of Information and Communication Technology
Stockholm, Sweden 2017

KTH School of Information and
Communication Technology
TRITA-ICT 2017:08 SE-164 40 Kista
ISBN 978-91-7729-363-7 SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av doktorsexamen i Informations-och kommunikationsteknik måndagen den 29 maj 2017 klockan 12.00 i Sal C, Electrum, Kungl Tekniska högskolan, Kistagången 16, Kista.

Tryck: Universitetsservice US AB

To my parents Llukan and Kushilda

Abstract

Distributed Real-Time Media Processing refers to classes of highly distributed, delay no-tolerant applications that account for the majority of the data traffic generated in the world today. Real-Time audio/video conferencing and live content streaming are of particular research interests as technology forecasts predict video traffic surpassing every other type of data traffic in the world in the near future. Live streaming refers to applications in which audio/video streams from a source need to be delivered to a set of geo-distributed destinations while maintaining low latency of stream delivery (as an example live event coverage). Real-time conferencing platforms are application platforms that implement many-to-many audio/video real-time communications. Both of these categories exhibit high sensitivity to both network state (latency, jitter, packet loss, bit rate) as well as stream processing backend load profiles (latency and jitter introduced as Cloud processing of media packets). This thesis addresses enhancing real-time media processing both at the network level parameters as well as Cloud optimisations.

In this context we investigated whether network resources could be controlled at the service level in order to increase network efficiency and performance, as well as quantifying the impact of the shared Network resource to Service quality. Shared network resources can negatively impact cloud service performance and thus by optimising or trading network resources can enhance Cloud service performance. This potential performance degradation is due to unregulated shared Network infrastructure (bandwidth resource allocation not aware of performance and Service Level Objectives (SLO). By mediating Network bandwidth through predictive control, we enable better usage of the available network resources and fewer SLO violations, achieving an increased system stability by at least a factor of 2.

The Access Networks (AN) (edge, core network) of ISPs, Carriers and Community Networks have no general purpose cloud infrastructure, while Internet Resource Providers provide on demand Cloud resources. We find an opportunity for the unification of the resources inside an AN and outside in order to provide a unified cloud offer through Cloud Federation and provide service mobility toward the users in order to optimise locality. This research work provides a novel hybrid Network and Federated Cloud architecture which provides an extended network infrastructure with a large scale Cloud deployment, directly incorporating it into the Network infrastructure. The novel Multi-Cloud architecture enables services to trade between locality to user or run-time performance thus optimising for latency toward optimal resource allocation for real-time applications.

To optimise latency in live streaming applications a novel self-managed Multi-Cloud overlay algorithm was proposed based on gradient topology in which each cloud for a stream broadcasting application optimises client proximity to the source. The application model is separated in a two layer design, the multi-cloud delivery back-end and the stream clients. The self-regulated gradient back-end minimises traffic load by creating a minimum spanning tree through the clouds that is used for the stream routing. The proposed algorithm has very fast convergence rate on large scale Cloud deployments, and is not effected by Cloud resource churn as well as providing increased resiliency of the live stream.

In this work we provide media quality analysis and enhancement of real-time Cloud stream forwarders, as well as allocation strategies to enhance service level performance of Web Real-Time Communication platforms. Media quality patterns are strongly influenced by Cloud processing performance, thus by tweaking this aspect we can control media quality. In particular we empirically demonstrate that as session sizes increase simulcast outperforms single-layer encoding. Additionally we introduce a stream allocation algorithm to minimise load spikes on Cloud stream forwarders and compare the behaviour of various stream allocation policies. With only minimal information and single server session allocation requirement, the minimal load allocation policy performs quite better than other Rotational, or Static Threshold based algorithms.

Keywords: *Multi-Cloud, Geo-Distribution, Live Streaming, Locality optimisation, Multicast, Multi-Cloud Overlay, Cloud Federation, Web Real-time Communications, WebRTC-RTCWEB, Media Quality*

Resumen

El procesamiento de medios en tiempo real distribuido se refiere a clases de aplicaciones altamente distribuidas, no tolerantes al retardo, que representan la mayoría del tráfico de datos generado en el mundo actual. Las conferencias de audio y video en tiempo real y la transmisión de contenido en vivo tienen especial interés en investigación, ya que la prospectiva tecnológica estima que el tráfico de video supere a cualquier otro tipo de tráfico de datos en el futuro cercano. La transmisión en vivo se refiere a aplicaciones en las que flujos de audio/vídeo de una fuente se han de entregar a un conjunto de destinos en lugares geográficos diferentes mientras se mantiene baja la latencia de entrega del flujo (como por ejemplo la cobertura de eventos en vivo). Las plataformas de conferencia en tiempo real son plataformas de aplicación que implementan comunicaciones de audio/video en tiempo real entre muchos participantes. Ambas categorías presentan una alta sensibilidad tanto al estado de la red (latencia, jitter, pérdida de paquetes, velocidad de bits) como a los perfiles de carga de la infraestructura de procesamiento de flujo (latencia y jitter introducidos durante el procesamiento en la nube de paquetes de datos multimedia). Esta tesis trata de mejorar el procesamiento de datos multimedia en tiempo real tanto en los parámetros de nivel de red como en las optimizaciones en la nube.

En este contexto, investigamos si los recursos de la red se podían controlar a nivel de servicio para aumentar la eficiencia y el rendimiento de la red, así como cuantificar el impacto del recurso compartido de la red en la calidad del servicio. Los recursos de red compartidos afectan el rendimiento del servicio en la nube y, por lo tanto, optimizando o intercambiando recursos de red pueden mejorar el rendimiento del servicio en la nube. Esta posible degradación del rendimiento se debe a la infraestructura de red compartida no regulada (la asignación de recursos de ancho de banda no es consciente de los objetivos del acuerdo de nivel de servicio (SLO) y de comportamiento). Gestionando el ancho de banda de la red a través de control predictivo, permitimos un mejor uso de los recursos de red disponibles y menores violaciones de SLO, logrando una mayor estabilidad del sistema por al menos un factor de 2.

Las redes de acceso (AN) (extremo, red principal) de los ISP, transportistas y redes comunitarias no tienen una infraestructura de nube de propósito general, mientras que los proveedores de recursos de Internet proporcionan bajo demanda recursos de la nube. Encontramos una oportunidad para la unificación de los recursos dentro de un AN y fuera con el fin de proporcionar una oferta de nube unificada a través de una federación de nubes y proporcionar movilidad del servicio hacia los usuarios para optimizar la localidad. Este trabajo de investigación proporciona una nueva arquitectura de red híbrida y nube federada que proporciona una infraestructura de red extendida con un despliegue en nube a gran escala, incorporándolo directamente a la infraestructura de red. La nueva arquitectura multi-nube permite a los servicios llegar a un compromiso entre localidad respecto al usuario o el rendimiento en tiempo de ejecución optimizando así para latencia para conseguir la asignación óptima de recursos de aplicaciones en tiempo real.

Para optimizar la latencia en las aplicaciones de transmisión en vivo se propuso un nuevo algoritmo de superposición de multi-nube autogestionado basado en una topología de gradiente en la que cada nube de una aplicación de transmisión de flujos optimiza la proximidad del cliente a la fuente. El modelo de aplicación se separa en un diseño de dos capas, el back-end de entrega multi-nube y los clientes de flujo. El backend de gradiente autorregulado minimiza la carga de tráfico creando un árbol de expansión mínimo a través de las nubes que se utiliza para el enrutamiento de cada flujo. El algoritmo propuesto tiene una tasa de convergencia muy rápida en los despliegues de nube a gran escala, y no resulta afectado por la rotación de recursos de la nube, así como proporciona una mayor estabilidad de la transmisión en vivo.

En este trabajo ofrecemos un análisis de calidad de los medios de comunicación y mejoras de los emisores de flujo en la nube en tiempo real, así como estrategias de asignación para mejorar el rendimiento de nivel de servicio de las plataformas de comunicación de Web en tiempo real. Los patrones de calidad de los medios están fuertemente influenciados por el rendimiento del procesamiento en la nube, y por lo tanto, al ajustar este aspecto, podemos controlar la calidad de los medios. En particular, demostramos empíricamente que a medida que los tamaños de sesión aumentan, la difusión simultánea supera la codificación de capa única. Además, introducimos un algoritmo de asignación de flujo para minimizar los picos de carga en los retransmisores de flujos en la nube y comparamos el comportamiento de varias políticas de asignación de flujos. Con la mínima información y el requisito de asignación de sesión de un único servidor, la política de asignación de carga mínima se comporta bastante mejor que otros algoritmos basados en un umbral rotativo o estático.

Sammanfattning

Distribuerad realtidshantering av mediadata syftar på klasser av starkt distribuerade tillämpningar som inte tolererar fördröjningar, och som utgör majoriteten av datatrafiken som genereras i världen idag. Audio/video-konferenser i realtid och överföring av innehåll “live” är av speciellt intresse för forskningen eftersom teknikprognoser förutser att videotrafiken kommer att kraftigt dominera över all annan datatrafik i den nära framtiden. “Live streaming” syftar på tillämpningar i vilka audio/video strömmar från en källa och behöver distribueras till en mängd av geografisk distribuerade destinationer medan överföringen bibehåller låg latens i leveransen av det strömmade datat (som ett exempel kan nämnas “live”-täckning av händelser). Konferensplattformar för realtidsdata är tillämpningsplattformar som implementerar realtidskommunikation av audio/video-data av typen “många-till-många”. Båda dessa kategorier uppvisar hög känslighet för såväl nätverkets tillstånd (latens, jitter, paketförluster, bithastighet) och lastprofiler av ström bearbetning “back-end” (latens och jitter introducerat som Cloud-hantering av mediadatapaket). Denna avhandling adresserar förbättringar inom realtidshantering av mediainnehåll både med avseende på nätverksnivåns parametrar och optimeraringar för molninfrastrukturen.

I detta sammanhang har vi undersökt huruvida nätverksresurserna kan kontrolleras på servicenivån i syfte att öka nätverkets effektivitet och prestanda, och även att kvantifiera påverkan av den delade nätverksresursen på servicekvaliteten. Delade nätverksresurser påverkar molntjänstens prestanda och dessa kan genom en optimering eller handel med nätverksresurser förbättra molntjänstens prestanda. Denna potentiella prestandadegradering beror på en oreglerad delad nätverksinfrastruktur (allokeringen av bandbredd är inte medveten om prestanda och mål för servicenivån). Genom att mediera nätverkets bandbredd genom prediktiv kontroll, möjliggör vi ett bättre utnyttjande av de tillgängliga nätverksresurserna och en lägre grad av avvikelser mot SLO, vilket leder till en ökad stabilitet med åtminstone en faktor 2.

Accessnätverken (AN) (edge, kärnnätverk) hos ISP, bärare och lokala nätverk har ingen generell molninfrastruktur, medan s.k. “Internet Resource Providers” erbjuder resurser för molntjänster “on demand”. Vi ser en möjlighet till ensande av resurserna inuti ett AN och utanför i syfte att erbjuda ett samlat molntjänsterbjudande genom s.k. “Cloud Federation” och erbjuder tjänstemobilitet för användarna för att optimera lokaliteten. Denna forskningansats erbjuder ett nytt hybrid nätverk med Federated Cloud arkitektur vilken ger en utvidgad nätverksinfrastruktur med en storskalig användning av molntjänster, som direkt inkorporerar denna i nätverksinfrastrukturen. Den nyskapande “Multi-Cloud”-arkitekturen möjliggör för tjänster att balansera lokalitet för användaren mot run-time-prestanda och därigenom optimera för latens mot optimal resursallokering för realtidstillämpningar.

För att optimera latensen i “live streaming”-tillämpningar föreslås en nyskapande självstyrd “multi-Cloud-overlay”-algorithm baserad på gradienttopologi i vilken varje moln för en tillämpning inom “stream broadcasting” optimerar klientens närhet till källan. Tillämpningsmodellen separeras i en tvålagersdesign, “multi-cloud delivery back-end” och “stream clients”. Denna självreglerande gradientbaserade “back-end” minimerar trafiklasten genom att skapa ett minimalt spännande träd genom molnen som används för routing av strömmarna. Den föreslagna algoritmen har en mycket snabb konvergenshastighet vid större moln, och påverkas inte av “churn” hos molnresursen liksom att den erbjuder ökad motståndskraft hos “live”-strömmen.

I detta arbete erbjuder vi mediakvalitetsanalys och förstärkning av realtidsmolnets “forwarders”, liksom även allokeringsstrategier för att förstärka servicenivåprestanda hos “Web Real-Time Communication”-plattformar. Mediakvalitetsmönster påverkas kraftigt av molnets bearbetsningsprestanda, och således kan vi genom att påverka denna aspekt kontrollera mediakvaliteten. Specifikt demonstrerar vi empiriskt att efterhand som sessionsstorlekarna ökar, så utklassar simulcast enlagersinkodning. Dessutom introducerar vi en strömallokeringsalgoritm för att minimera “load spikes” hos “Cloud stream forwarders” och jämför beteendet hos olika strömallokeringspolicys. Med enbart minimal information och allokeringsbehoven hos en enskild serversession beter sig den minimala lastbalansallokeringspolicyn tydligt bättre än andra “rotational”- eller “static threshold”-baserade algoritmer.

Acknowledgments

This work was done and partially funded by the framework of the Erasmus Mundus Joint Doctorate in Distributed Computing (EMJD-DC) from the Education, Audiovisual and Culture Executive Agency (EACEA) of the European Commission under FPA 2012-0030, and Spanish government under TIN2013-47245-C2-1-R. A special mention is for Tokbox Inc. *a Telefónica company*, for funding the last part of this research work, and providing the testing environment and underlying technology to build on.

First I would like to extend my special thanks to my primary supervisors Leandro Navarro and Vladimir Vlassov, for their patient guidance, constant feedback, indispensable encouragement and unconditional support throughout this work.

My secondary supervisor Seif Haridi, for his constant presence and valuable insight. My advisors at Ericsson Dura, Magnus Molin and Simona Toma for introducing me to the complexity of Carrier Network architecture. My advisor at TokBox a Telefónica Company, Òscar Divorra Escoda for the continuous support and for the future collaboration.

Thankful for their insight and the impact they provided to my research.

Thomas Sjöland, Sandra Gustavsson Nylén for the support throughout my PhD studies at UPC and KTH.

To my family, the people that have seen me grow and have always been there: Saimir, Rrema, Bukurie, Flutura, Bardha, Neta.

For the people who were there to push me past bad moments and to savor the good ones, my brothers Alden and Ergys, my dearest friends Miki and Keti.

Special thanks to Sebastian, Rafaela and Sebastian for making Spain feel like my second home.

At last to my dear Aurora: for the help, the support, and unconditional love! Glad to have you in my life!

Vamis Xhagjika

Contents

List of Figures

Chapter 1

Introduction

1.1. Research Questions

The finality of this work is to develop empirical methods to enhance real-time stream processing applications with a focus on real-time media processing applications. More specifically live streaming and real-time conferencing by means of Multi-Clouds and Cloud Federation. In order to achieve the desired enhancements in performance metrics of real-time applications, we have investigated various interesting research problems related to both Cloud and Network architectures as well as real-time services. We briefly introduce some of the research questions that were tackled in this doctoral work and the methodology used to derive and evaluate such results.

Quality of service can degrade when multiple Cloud Services or multiple Resource Providers share the same network infrastructure, this is due to the Network architecture not being aware of application level objectives. In particular real-time applications have very strict performance objectives due to their real-time nature and even the slightest service level objective violation would impact service quality. Can we enhance service quality by controlling the shared network infrastructure parameters? The answer to such question is a positive one. By controlling network bandwidth at the edges of the service we can enhanced application performance for services sharing the same Network infrastructure. This leads to optimized service performance, and at the same time lowers service level objective violations. We conducted an empirical study on the performance of a distributed object store (OpenStack Swift), and exploited patterns in network bandwidth allocation to enhance service stability. Based on the data that was gathered from our Cloud test-bed, we provide a predictive network bandwidth manager that arbitrates network bandwidth between the nodes of distributed services. The proposed network bandwidth manager was evaluated on the Cloud test-bed and showed a gain in system stability of at least 2 times higher than the uncontrolled system.

Performance of real-time media stream processing applications is much impacted by network conditions between the clients and the Cloud service. Locality to the user is of utmost importance as it ensures better connectivity and enhanced locality leads to lower latency overheads. To enhance the quality and performance of such applications, the

services need to be highly dynamic and allocated close to the end user. Can existing large scale network deployment be modified to enhance service locality? Can Cloud services transparently be allocated in close proximity to Cloud application users? To provide answers to such questions we provide a unified Cloud architecture that enables a new Cloud model where Infrastructure providers (such as Telecom, ISPs, and Community Clouds) can become a Resource Provider for cloud services with a unified interface. This in terms improve services by enabling them to move closer to the users and lower service latency. Structural analysis was conducted between existing Cloud and Network Infrastructure Providers in order to create a novel hybrid Cloud/Network model to enhance service locality and provide a hierarchical Cloud architecture. We provide an architectural blueprint for a novel hybrid and hierarchical Cloud-Network model which enables large scale deployments, up to one hop away from the user.

This hybrid Cloud-Network architecture provides an interesting case toward optimizing applications that are impacted by latency. Such applications can use enhanced locality in order to minimize stream latency thus improve service quality. Can we improve live streaming applications, by using such novel Cloud model? Can we support large scale broadcasts, with clients in a multitude of geo-locations? Toward such providing such enhancements, in this work we provide a self-organized Inter-Cloud overlay algorithm to allocate and route live stream sessions though a large scale Cloud architecture. The proposed solution builds a gradient topology to optimize distribution path length of the Inter-Cloud streams and provides various benefits to known approaches. We conduct an study through simulation in which we verify that the overlay converges very fast and is not impacted by large scale. In fact we prove that the convergence of the overlay is not impacted by the scale of growth of neither the clouds located at the edges of the Network nor the clouds on the Internet outside of the Access Network.

Another case of real-time services that are hosted and run on Cloud environments are real-time live conferencing applications. The backends of real-time conferencing systems should handle a high number of communication streams while maintaining minimum processing latency. This processing latency would be propagated to the media stream and violate the real-time constraints. What is the impact of Cloud processing and stream routing on the media quality? Can we enhance media quality by lowering backend maximum server load? We conducted a study to discover patterns between media quality and Cloud service load, and whether we can improve quality of service by exploiting such patterns. The study (an empirical approach), was conducted by sampling passive client measurements of a Web Real-Time Communication (WebRTC) platform. We discover embedded patterns between load and media quality as well predictive patterns of backend loads. Furthermore we constructed a Cloud simulator in order to experiment on different stream allocation policies to minimize the impact of Cloud service load to media quality. The proposed stream allocation algorithm provides improved service quality by lowering the number of SLO violations.

1.2. General Introduction

The present work on this doctoral thesis is based on the notion of multi-cloud federations, architectural challenges and applications. According to Informa Telecom & Media annual report of 2012 the total number of deployed macro and micro cells for Telecom (Carrier) Providers worldwide ranges in the order of 6 million (6 069 224) microcells and 5 925 974 of macro cells deployed. The locations of deployment for such cells provide only network related services of connectivity but may very well be extended to accommodate a massively distributed cloud environment. This key points is where the cloud can be deployed to provide user proximity of up to one hop away from the End Users. Fig. 1.1 from Ericsson Mobility Report 2016 [1], provides evidence of enormous growth in mobile traffic as compared to other sources of traffic by a factor of 12x, thus surpassing every other source.

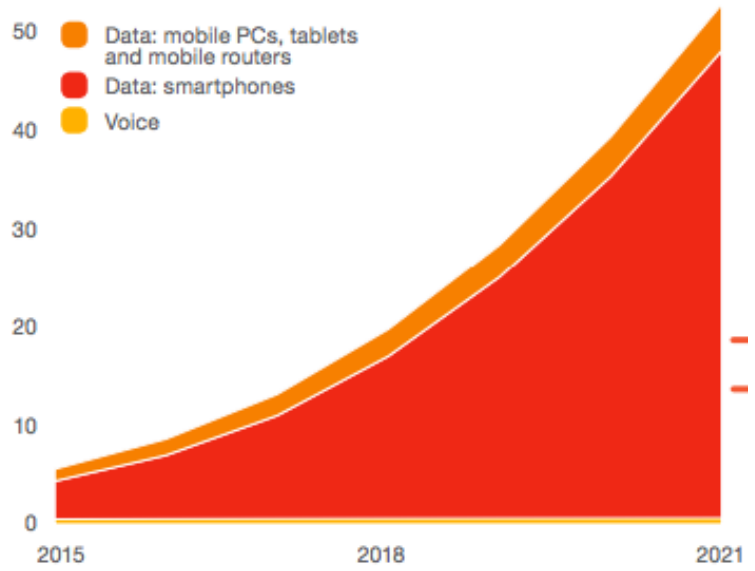

Figure 1.1 – Global Mobile traffic projection 2 015-2 021 [1]

Statistics presented in this report, provide a clear opportunity to provide cloud services close to end users, and that are optimized both for mobile clients as well as for different Access Network topologies. At present the challenges in multi-data center clouds are focused on autonomous resource trading, interoperability and various different architectural aspects like vendor-lockin, etc. Previous work in cloud federations is mostly based on such challenges namely on federation architectures and resource trading, thus for the doctoral thesis work we selected a bit of an orthogonal direction. This thesis considers the benefits of merged in a symbiotic relationship Multi-Cloud Architectures and the underlying Access Network architecture to provide performance enhancements in terms of latency and stream quality for live and real-time streaming applications.

Optimization of such real-time conferencing/streaming applications conforms nicely with a distributed environment having a highly decentralized, massive scale, cloud architecture that can provide performance and cost benefits. In general real-time streaming applications are relatively state-less applications but are subject to not having great tolerance to stream delays. Such streaming applications are impacted by network quality, user proximity, as well as the cloud backend performance. The fact that a relatively small application state is needed leads to small footprint in application environment, which can be relocated without high overhead. In this work I exploit the highly distributed nature of today's Access Networks in order to provide a novel Cloud and Network hybrid environment that provides application developers trade-off opportunities between locality and performance in order to optimize streaming applications.

1.2.1. Federated and Multi Cloud

Applications that reach a large scale both in terms of user workload and geo-distribution have a need to scale beyond the bounds of a single data center in order to optimize Service Level Objectives (SLO), costs, or proximity/latency toward the end users. When an application spans multiple management domains such architecture is considered to be a Inter-Cloud deployment, for a more concrete definition of the Cloud model taxonomy we refer to [9]. These environments spanning multiple Cloud domains enable applications to augment geo-distribution and provide better coverage to applications that have a highly diverse and distributed usage pattern. In the case of live streaming/conferencing applications, the end users using the Cloud service are sparse allover the world and interact with each other through audio/video or augmented reality means. In such a scenario of having a highly distributed application, resources need to be carefully distributed in order to optimize cost and user experience (audio/video stream latency, bit rate). A centralized solution would not cover all the application needs as users with closer network proximity to the data center would perceive better quality than users of further distance.

Inter-Cloud deployment provides not only benefits in terms of locality to user, resource availability but as well as in terms of cost optimization. By having multiple providers applications can be made to dynamically scale in or out and thus manage run-time costs. In the contest of streaming applications not only user locality is of utmost importance but also network usage. Such applications are mostly of the type of CPU/Network eager applications and the operational costs are contributed in general from CPU/Network utilization. If the applications would not be distributed between different cloud providers the single networked backbone of the data center would be a performance bottleneck, as well as the main cost of running the service.

In this thesis we will explore a novel hybrid architecture that merges the concept of a Inter-Cloud into the provider networks such that networked applications can be isolated to a certain degree of locality and thus lowering network traffic on the network backbone as well as providing shortest routes to Cloud-based stream forwarders. The real-time nature of streaming applications demands for better Inter-Cloud routing of application streams, as well as in-Cloud load management of resources so that the cloud forwarder processes

each stream in the shortest possible amount of time. The novel hybrid Cloud-Network architecture provides applications with the ability to trade-off between performance and user proximity as close as one hop away.

1.2.2. Network Architectures

Network architecture and the relation with Cloud architectures sharing the same network resources is a key actor in this work. The kind of networks which were accounted for are ISP and Carrier network. ISPs and Carriers share a lot of similarity in their network construct and also provide a sole architectural entity over which we build a highly distributed Cloud environment that permits to integrate Cloud and Network in one. Highly distributed Clouds can be used to optimize applications for user locality and also at the same time optimize backend network usage. The key benefits of a networked infrastructure such as ISP or Carrier providers are: high geo-distribution of compute resources, large client footprint, and large scale.

By integrating a Cloud architecture that spans in all of the points of presence of these networks, the resulting massive-scale Cloud shows the same benefits as the underlying network. Not only the cloud can benefit by such large scale distribution and presence, but also applications gain much more dynamic behaviour in terms of resource distribution. In a general high level view of a ISP and Carrier architecture we see the following components which in our novel architecture design are enhanced with cloud enabled hardware/software:

1. **Internet Facing Gateway** Are point of interconnection between different providers or the open Internet.
2. **Core Network** Is the main routing and accounting facility for a provider.
3. **Points of Presence** Are highly geo-distributed points of presence that are used for client inter-connectivity.

By augmenting this key network points with cloud enabled environments we can have a massive-scale cloud computing platform which presents augmented user locality as well as resource availability. We analyse network performance not just in the context of the presented novel architecture but as well as in the view of commonly shared resource by multiple cloud services. By mediating network resources we can derive the impact they have on cloud applications as well as the symmetrical case in which Cloud choices discriminate network performance of specific networked applications. This property is even more evident in the case of real-time media streaming applications which are very sensitive to both Network and Cloud architectural choices.

1.2.3. Real-Time Streaming Applications

Real-time communication applications are of a specific type of highly distributed and delay non-tolerant nature, these applications provide media communication between clients in real-time. The delay non-tolerant aspect of such applications puts stringent performance objectives, and do not permit the deployment of caches or other buffering

mechanisms for enhanced media quality. In this work we consider both a generalized view of live streaming as well as a more concrete implementation of real-time communications by examining stream allocation policies for Web Real-Time Communication technology stack. Network performance and stream processing time (for activities of stream forwarding or stream transcoding) have a big impact on the quality of media of such architectures thus novel algorithms and stream allocation policies should be introduced. In this work we analyze a WebRTC implementation and examine media quality in relation with load parameter, which is defined in terms of streams per server or streams per session.

WebRTC as a technology stack is well standardized [10] and implemented by the majority of modern browsers (Google, Mozilla, Edge etc...). This series of protocols and technology stacks have seen a wide adoption in live conferencing as well as live streaming services. A real-time communications enabled architecture is highly distributed by design and can be implemented as a P2P solution but as well as cloud based. In the scenario of a cloud backend backed WebRTC implementation cloud servers are used as media relays with software level forwarding and with/out media transcoding. Cloud based stream forwarding implementations provide a big improvement over P2P in terms of upload bandwidth but introduce Inter-Cloud distribution and application level routing concerns.

Live streaming applications can benefit by highly distributed cloud architectures, as the small overhead permits to have a better resource allocation in order to move the services closer to the users and thus improve media quality due to better network usage and decreased latency. The presented novel hybrid Cloud-Network architecture part of this work, provides the possibility for the application implementers to not only move resources closer to a specific user, but as well in geographical key locations that provide the best network performance for all participants in a live audio/video conference.

1.3. Research Goals

We present in this section the research goals that were the focus of the work towards this PhD thesis. For each goal we have presented results in the form of conference papers that tackle a specific problem from the goal pool. We rephrase here in a short and contained way the research objectives previously defined in this chapter.

Regulation of shared network infrastructure: Multiple cloud services and resources providers transparently sharing the same network infrastructure results in potential performance degradation due to unregulated shared network infrastructure (bandwidth resource allocation not aware of performance and service level agreement objectives). This work contributes a machine learning control algorithm that is used by broker agents to arbitrate bandwidth between end nodes.

Common network resource interface: In the current state of Access Networks (ISPs, Telecom Providers and Community Clouds) the cloud is fragmented with allocatable resources outside of the AN. The Access Networks (edge, core network) have no general purpose cloud infrastructure, while Internet Cloud Providers (Resource Providers) provide on demand cloud resources. Opportunity for unification of resources inside an AN and

outside in order to provide a unified cloud offer through cloud federation. For this objective there is a need of cloud infrastructure in the AN with a unified interface toward Resource Providers such as the network operator becomes a box/resource provider (with locality in box provision and open to deployment from any service providers). The hybrid Cloud-Network architecture introduced in this work provides a multi-tier multi-cloud federation in which services can be moved anywhere from the Internet to the Edges of a AN in order to enhance locality or further towards public Clouds to enhance resource performance.

Performance Optimization for application-level multi-cast streaming: Such architecture introduces a clear case for optimization of resource selection according to performance objectives of locality aware services. The unified cloud infrastructure of Resource Providers enables service providers to trade-off over locality and performance, through box/resource allocation. To make the case for such services, we see an opportunity in providing a latency based optimization to one-to-many multi-casting for any stream based service in need of such abstraction. The work contributes a distributed algorithm, building a bounded depth multi-cast tree for software-level multi-casting. Fig. 1.2. shows why such contribution impacts present state of Network Providers where video traffic is projected to 70% of the whole data traffic by 2021.

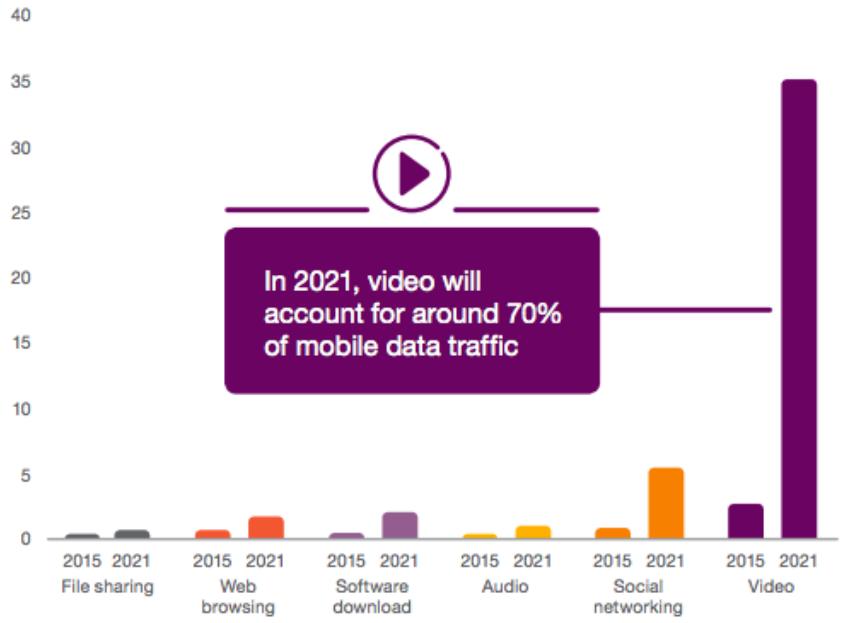

Figure 1.2 – Global expected Video Traffic 2 015-2 021 [1]

A novel resource allocation and discovery model is needed in order to provide better resource selection opportunities to service providers for service optimization. The model should incorporate detailed service and network information, like user volume for a given area, network congestion and capacity for a given part of the network, availability of cloud

resources for such area and in turn such usage of the model would provide better optimization of service performance metrics like reduced cost, reduced backbone traffic, reduced user latency etc... The matter of providing a more refined model for resource discovery, resource allocation and re-allocation in the proposed architecture will be the object of further inquiry towards other service optimizations. Usage of the proposed architecture and modifications of the model based on service needs could enable further optimizations to existing and novel services to be deployed.

Toward this goal we have examined in the form of real-time media quality parameters and load impact for WebRTC cloud based solutions. The main parameter of focus is bit-rate as it is a direct indicator of stream quality. Different stream quality patterns came out of our analysis that connect both quality and server loads. The study was not conducted on the load patterns and scale of the streams allocated to a particular server but as well on the impact of session size towards media quality. A minimal load algorithm was proposed for the stream allocation in a data center and through an empirical study was demonstrated to be sufficiently good at maintaining low operational max streams per machine.

We have demonstrated though big data analysis of stream parameters monitored passively at the client, that as scale grows single layer video encoding does not perform as well as simulcast. From the processed data we see a difference as high as 2x between the average bit rate of sessions and the minimal bit rate of the same session between simulcast and single layer encoding. This results makes simulcast the preferred way to deliver video in a multiparty conference with more than 3 connected streams.

1.4. Research Methodology

This research work follows an empiric approach to tackling problems in Inter-Cloud deployments of real-time applications such as live streaming and live conferencing and optimization that lead to lower management overhead/latency as well as better resource allocation algorithms. The research work includes as well big data analysis in exploring possible performance and quality issues on distributed real-time communication scenarios.

In exploring the state-of-art Inter-Cloud environments such as Openstack[2, 11], Community Clouds[12, 13, 14, 15], network architectures such as 3GPP based Mobile Carriers[16], ISP network deployments and real-time communication backends, namely TokBox real-time communication infrastructure in the web. I proposed through structural analysis a novel cloud architecture that embeds Cloud within the Network infrastructure and that provides additional benefits to application developers such as the possibility to trade-off between locality and performance, as well as deploying cloud applications as close as 1 hop to the targeted users. Once having identified such hybrid infrastructure design I moved into real-time stream processing application scenarios that could benefit from such architecture and optimized various aspects of such applications.

The selected applications were real-time stream processing applications, which have low state overhead, while having stringent real-time constrains and no-delay tolerant nature. These applications can benefit greatly by being deployed on vast scale and in close user proximity as that lowers the latency that such applications exhibit toward the user.

Challenges I came across in examining such deployments were at first a best effort management of the underlying network shared by multiple cloud services. We proposed a network bandwidth manager that arbitrates bandwidth between different cloud services at the endpoints and thus can trade-off among them. The proposed broker lowers the number of service quality violations and thus increases network efficiency. We used a real deployment of a Cloud object store service to conduct measurements of both application performance and network usage for different patterns of workloads. We trained a regression model as a control mechanism for our network bandwidth broker and used real experiments to validate the approach.

Other challenges that were encountered during such analysis was the complexity in maintaining and managing a highly distributed Cloud deployment, to tackle such problem a Sel-Managed Inter-Cloud Overlay was proposed for live streaming applications so that the clouds participating in a stream broadcast self-arrange in a gradient topology with metric distance to source. To verify such design, we provided simulation results that verify overlay convergence, and low setup time. For this part of the analysis we implemented a discrete event simulator for a Multi-Cloud environment, and used the simulator to run multiple experiments. The simulation environment was written in Python and does not use active threads to simulate Cloud entities. By using environment properties of the programming language the entities are not active and as such CPU utilization does not impact the quality of the results. The simulated results are congruent with results observed in the state-of-the-art.

In exploring real-time communication applications and the impact of user-load on media quality, we analyzed user traces from a live testing framework. By running big data analysis frameworks and analysis software written by us to analyse time series data for streams and sessions, we provide empirical evidence that different video coding techniques and adaptive streaming technologies behave differently as session size grows. Additionally we provide insight into temporal patterns in user load as well as an algorithm for stream allocation. In order to compare the performance of this algorithm with the state-of-art, we developed a Cloud simulator based on CloudSim[17] framework, over which we can simulate different scenarios based on real user traces. Having such powerful simulation tools it was possible to compare the load profiles of the different algorithms in order to provide evidence supporting the provided algorithm.

1.5. Contributions

The research conducted in the context of this doctorate work was mainly focused on cloud federations of a massive scale with the finality the enhancement of live streaming applications and live conferencing systems. In this aspect we can separate the work into two main categories, namely: architectural design for large scale federated cloud architectures and live streaming delay no-tolerant application optimizations. As such this section will at first introduce the architectural design contributions, and then continue with introducing the distributed real-time communication applications and various related enhancements. Live real-time conferencing and streaming applications, are relatively stateless applications but

with extreme demand on latency objectives. Traditional techniques used to hide latency such as buffering, or caching is not viable as both schemes imply that the stream at the receiver will be lagged but continuous.

When studying current state-of-the-art deployments of Multi-Clouds and possible network architectures toward a more decentralized Cloud, we realized a gap in knowledge or of a standardized architecture for a massively scaled geo-distributed Cloud with enhanced service latency. To improve over this gap in knowledge, we contribute a highly distributed Multi-Cloud novel architectural blueprint, that enables the distribution of Cloud applications in a standard and transparent way at crucial points of the network infrastructure. This architecture presents a stable cloud federation in which services can choose to trade between locality or performance in order to either diminish latency to the user or augment resource needs towards higher performance needs. By geo-distributing the cloud from the Internet, to the core network, and down to the points of presence of the network (as far as 1 hop away from the user) we lower requirements for data centers in terms of maintenance, energy, cooling etc... Such Cloud also is more resilient as each Micro-Cloud is a autonomous entity, which can run independently of the federation authority, thus loss of connectivity between the Micro-Clouds does not break local running services. As the hybrid Cloud-Network architecture presents a Cloud which is embed-ed in the network infrastructure, another perceived benefit is bandwidth usage of services, which if routed intelligently between the micro clouds can optimize bandwidth needs of the services. The novel Cloud architecture enables service placement at any part of both Resource Providers and Network Providers thus enabling services to move closer to the user in order to dynamically improve service latency. By improving Cloud service latency we can have stronger media quality levels for distributed real-time stream processing applications.

In *BwMan*, the Network Bandwidth Manager, we present a algorithmic approach in which Network Resources can be arbitrated in order to stabilize Cloud service performance. Considering real-time media stream processing applications, bit rate (another form of bandwidth) is of utmost importance and directly impacts media quality. BwMan uses a predictive MAPE control cycle and arbitrates bandwidth by tweaking endpoint bandwidth capabilities. By using such self-management techniques we can prioritize user-centric traffic (media streams) over system-centric traffic (accounting and data replication) in order to maintain stream performance while slowing down on non-functional data consumption. To verify our approach we apply the bandwidth managing techniques to a distributed data store, treating different user-centric, system-centric streams as different services sharing the same network infrastructure. It permits us to trade between user performance and system maintenance needs in order to favor one or the other. Even though the machine learning model is not a on-line model, it performs well for the intended workloads. The control system of *BwMan* permits a increase in system stability, at the level of 2x increase in system stability as compared to the uncontrolled scenario. This paper permitted me to see a much deeper correlation between the shared network resource and Cloud services. This duality of Cloud and network can be exploited toward lowering latency for streaming applications and conferencing systems. The following work builds on this notion to provide optimizations

to such applications.

In order to best optimize latency of Cloud services, a highly distributed environment providing resources in proximity of the users is needed. The extreme large scale of the distributed environment presents challenges in optimizing and managing a highly distributed Cloud infrastructure, not only for the geo-distribution scale and network fluctuations but also for the scale of the cloud management layer involved. A centralized solution would not be the best solution as message overhead for management would lower service time and admission times for the services running of such cloud, as well as pose a central point of failure and potential bottleneck. In case of live streaming and real-time communications we are dealing with a delay no-tolerant scenario, as such novel algorithms need to be implemented with lower management overhead. We introduce a self-managed Inter-Cloud overlay, that builds a gradient like topology, in which the source of the stream is the center of the gradient and the stream forwarding Clouds are arranged in orders of increasing proximity to the source. The novel Inter-Cloud overlay algorithm shows convergence rates of 8 - 14 communication cycles, and the same rate is shown as the Inter-Cloud overlay grows from 100 to 10 000 Clouds taking part in the federation. The algorithm approximates very closely the optimal path from source to destinations while keeping a very low convergence time. The experiments were conducted by having a varying number of Clouds that is far more inflated that what real scenarios are, this contributes to further validate the algorithm.

This work further focuses in the impact of load patterns in a distributed Cloud-based stream forwarder back-end in media stream quality for a WebRTC framework. Web real-time communication is a standard that enables platform-agnostic audio/video communications. The standard framework provides implementation guidelines for both P2P and Cloud-based relayed communications. The Cloud relayed traffic in our work is handled though Selective Forwarding Units (SFU) which implements software based multi-casting and are deployed in the same Cloud or multiple ones. Each SFU can handle a determined number of streams and as load increases for such distributed forwarders, audio and video quality are impacted. We provide an empirical study of different video encoding techniques (simulcast and single layer encoding) and compare them in terms of scale impact toward bit rate of streams (bit rate of a stream measures the quality for a given resolution). We demonstrate that, given large scale sessions, simulcast outperforms single layer encoding, in terms of average to minimal bit rate in a session, by as much as a factor of 2x. To minimize forwarding latency all streams that go to the same session need to be scheduled to the same SFU as such the algorithms that can be used are limited. We propose a minimal load server allocation policy with randomized minimal set, in order to lower probability of consequent load spikes impacting the same SFU. We implemented a Cloud simulator in order to simulate different state-of-the-art algorithms based on real test traces. The evaluation of the algorithms shows that the proposed algorithm outperforms the rest with a gain in max streams reached on server at levels 20%-50% less than the other reference algorithms.

1.6. Research Limitations

In this section we provide a view of research limitations that come with the provided approaches developed in this work toward better multi-cloud infrastructures and enhanced real-time media streaming applications.

The architecture design we provide in order to augment user locality and enable application deployment as close as 1 hop to the user provides a very promising highly distributed Cloud architecture. Nevertheless there are some shortcomings that need to be addressed in order for this Cloud architecture to be realised. *One such challenge is a generic resource allocation scheme, as the closer resources are deployed to the user the less available real resources we find.* This property of the system permits to trade-off between performance and locality, but if not well managed can lead to resource starvation. Thus a prioritization based deployment algorithm could be used based on auctioning or efficient time sharing for low power devices. A *second limitation of the proposed hybrid Cloud-Network architectural design is that according to the actual 3GPP standard for mobile carriers, the client and the core Network (main routing facility) are connected though direct encapsulated channel in a point-to-point fashion.* This would directly cutoff additional cloud based software to be deployed in the Base Station. This aspect can be overcome through the use of Software Defined Networks substituting the point-to-point link and giving traffic visibility to the Base Station. Additional mechanisms could be based on deep packet inspection to bypass the lower level routing of data to the Cloud enabled hardware in the Base Stations. A further limitation to be considered is that at the moment even though this cloud architecture presents the best deployment of cloud at such a scale, there exists no real deployment of such cloud infrastructure.

The network bandwidth manager **BwMan**, arbitrates bandwidth cloud services and more specifically in the presented work for Distributed Object Store Swift[18, 19]. The approach predicts load distribution in terms of bandwidth needs and arbitrates bandwidth allocation between service endpoints. This approach permits arbitrating existing bandwidth between endpoints but does not account for possible scaling mechanisms that in conjunction with data center SDNs could scale the bandwidth behind the actual limits set by the network. *The model of BwMan is fit to the data in a offline manner and as such could not handle unexpected situations that may arise outside of the flexibility bounds induced by choices in training the model.* The allocated bandwidth can be scaled only at the endpoints thus does account for variable topologies, and gives an averaged view of the system abstracting away the network topology.

As integral part of this research work, real-time applications were regarded as application scenarios in which to provide optimizations. A self-manged gradient based Inter-Cloud overlay, is provided to self manage massive scale deployment of live streaming architectures. The overlay provides routing capabilities between clouds building a minimal path distribution path spanning from the source of a live stream to the destinations. The algorithm performs well in managing large scale deployments, as of experimental results it can scale easily even in the order of hundred of thousands of Clouds. *One limitation of the approach is that it does not provide a algorithm for allocating new clouds, lowering the number of clouds in order to optimize cost, as well as*

the function that is calculated in a distributed manner takes only care of path length, thus not about cost. As cost is an important factor in deploying software solutions, the algorithm needs to be extended in order to not only optimize path length/latency but also cost of deploying the application.

At last this research work focuses on real-time communications in the context of WebRTC. The WebRTC standard describes media requirements and protocols in order to have a standard platform-agnostic communication framework. During our research work we discovered various temporal patterns of server/client load and audio/video quality objectives, based on such patterns a minimal load algorithm was introduced in order to fairly distribute load among servers and lower the impact of load spikes to media quality. *The proposed algorithm works as it is tuned to the load profile, but can't adapt to changes in workload profiles.* Since the load patterns may vary in the future due to unexpected uses of the system or new applications based on the studied platform. This shortcomings are inherent by the restrictions on the system, as once allocated a stream can't be switched to a different server and as such the first allocation discriminates the load. *Possible solutions could be by changing the system to permit conference sessions to span multiple servers.* This change would introduce more challenges on maintaining low latency and could be study of future work.

1.7. List of Publications

The content of this thesis is based on material previously published or submitted in peer reviewed conferences.

Chapter. 4 presents BwMan the bandwidth manager for cloud services sharing the same network infrastructure and is based on:

1. BwMan: Bandwidth Manager for Elastic Services in the Cloud -Ying Liu, **Vamis Xhagjika**, Vladimir Vlassov and Ahmad Al-Shishtawy, In *12th IEEE International Symposium on Parallel and Distributed Processing with Applications* (ISPA), 2014 [Rank:B Conference]

Chapter. 5 provides a novel Multi-Cloud architecture for Carrier and ISP providers and is based on:

2. Structured Cloud federation for Carrier and ISP infrastructure - **Vamis Xhagjika**, Vladimir Vlassov, Magnus Molin, Simona Toma, In *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference* [Acceptance Rate: 28%]

Chapter. 6 is based on:

3. Enhancing Real-Time Applications by means of Multi-Tier Cloud Federations - **Vamis Xhagjika**, Leandro Navarro, Vladimir Vlassov, In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)* [Rank:C Conference, Acceptance Rate: 17.8%]

Chapter. 7 is based on:

4. Media Quality-Centric Stream Allocation and Related Patterns for a WebRTC Cloud Architecture - **Vamis Xhagjika**, Òscar Divorra Escoda, Leandro Navarro, Vladimir Vlassov [UNDER REVIEW]

5. Load and Video Performance Patterns of a Cloud Based WebRTC Architecture - **Vamis Xhagjika**, Òscar Divorra Escoda, Leandro Navarro, Vladimir Vlassov, Poster Paper In IEEE/ACM CCGrid 2017 [Rank:A Conference]

1.8. Research Ethics

This research work in no case has included human or alive organisms experimentation, all the results are exempted from discussions over most of ethic issues. All the proposed solutions in terms of Cloud architecture adhere to usage contracts that the clients of the architecture would sign with the providers and as such be protected by separate legal agreements. In no case the live streaming and real-time audio/video communications frameworks are to be used for streaming illegal content, and it is sole responsibility of the user of the platform if such assumption is violated.

The analytical part of this thesis which uses real user traces for simulating load balancing algorithms in the Cloud, uses data from a test data center but only at the stream level, with no user information. The only data used to identify the streams is a stream identification ID. These IDS are anatomized by removing any meaningful data substituting them with random integers so that no match can be traced back to any user. The arrival times of the streams are as well normalized to the first inter-arrival time set to 0, so no actual identifiable information is left on the data of the streams. This anonymity process and the lack of user information, limits the number of algorithms that can be applied to stream allocation algorithms in the sense of using load predictors to drive decision, but the most important prerequisite for this work was to keep away any user data in the process, and drive the decisions just based on system information that are unrelated to a particular user.

1.9. Structure of Document

The remaining part of this thesis is organized as follows Chapter. 2 provides an introductions and background over the main concepts used in this thesis. This chapter introduces various background information regarding Network Architectures, Multi-Cloud, Cloud Federations, WebRTC Media Protocols and Cloud-based Relay of multimedia streams. In Chapter. 4 we introduce a cloud bandwidth manager that arbitrates cloud bandwidth between services sharing the same network infrastructure. Following, in Chapter. 5 we propose a massively scaled, hierarchical, hybrid Cloud-Network Federation architecture in order to optimize service locality and tradeoff performance for locality or the other way around. Building on these previous work Chapter. 6 provides a multi-cloud self-managed management overlay for live streaming services that can manage with low overhead tens of thousands of clouds. At last in Chapter. 7 we exploit bit rate quality patterns of a distributed WebRTC Cloud-based relayed framework in order to optimize user quality. Additionally we provide a stream allocation algorithm that without any user information, tries to fairly distributed streams across the Cloud-based SFUs.

Each one of the following Chapters [5 - 7] follows the same structure in which: a small resume is given, then each subject follows a paper structure of introducing research problem and solution with evaluation (when present), ending with concluding comments.

Chapter 2

Background

2.1. Introduction

In this chapter we describe the necessary technological background needed to understand the work described in this work. This material introduces the basic general context of the environment in which this work operates on. We introduce the general concepts of Cloud Computing and Inter-Clouds which are necessary to understand the work conducted on architectural design and structural analysis. Additionally we describe Cloud deployment models as well as different Network provider architectures, to further continue with a description of live streaming applications which are the main focus of this work. We end this background chapter with a description of techniques used in this thesis to develop novel algorithms and empirically study the systems of interests. These core concepts are the basics over which we build the work and presented in the other chapters. Thus with that we introduce the main environment concepts without dwelling in too deep and leave to Chapter. 3 a study of related work and a deeper loolook inside the problems that this thesis tackles.

2.2. Cloud Computing and Inter-Clouds

Cloud Computing abstracts computing resources to a utility, on-demand, based model. In this model computing resources are considered to be allocated on-demand with a infinite availability. Cloud computing presents both virtualisation of network and compute resources and cloud clients can allocate nearly infinite combination of compute resources and network topologies, to implement their distributed applications. Physical data center resources are sold on different pricing models and can be allocated either on a space-shared (with queueing) or time-shared (virtualisation). Cloud users can allocate and de-allocate resources though software interfaces and virtually consider computation as utility resource. Following this paragraph the definition of Cloud computing from the National Standards Institute:

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand

> network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.[20]

With the advent of Cloud technologies new problems have risen in dealing with the geo-ditribution of multiple resources in multiple data centers, as well as problems of scale, reallocation and performance. In this thesis we use such large scale distribution of cloud resources in order to enhance distributed applications. Some essential characteristics of Cloud computing as listed in [20] are :

1. *On-Demand/Self-Service* The ability to allocate resources on-demand and through self-service interfaces.
2. *Broad-Band Access Network* Guarantee high speed broad band network to all of the resources.
3. *Resource Pooling* A common pool of resources is transparently scheduled between multiple tenants.
4. *Rapid Scale* The ability of a Cloud to adapt to scale in the shortest possible time.
5. *Resource Metering* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction.

A Cloud provider is an entity that owns cloud resources and sells them at different pricing models. Resources can be sold through various pricing models of which some are a pay-as-you-go model based on resource usage, flat cost pre-allocation or market based. In the context of the Cloud a client can either be considered as the User who directly buys and uses Cloud resources in order to implement distributed applications, or the users of such applications (End User), in this work we will differentiate between these two user categories based on the context and will explicitly point out the user category subject of the analysis.

Software as a Service **(SaaS)**
Data as a Service **(DaaS)**
Platform as a Service **(PaaS)**
Infrastructure as a Service **(IaaS)**

Table 2.1 – Layered Cloud service model

The service model for the Cloud is a layered model in which each layer builds functionality based on the preceding layers. The general nomenclature is based on the level of abstraction of the provided service and it is of the form **X** As A Service (where **X** can be *Infrastructure*, *Platform*, *Software* or recently introduced *Data*). The composition

of the layered model is shown in Tab. 2.1 starting from the more concrete layer (Infrastructure) to the most abstract (Software). Thus in the service model we can find that Infrastructure becomes a service, Platforms is considered as a multi-tenant service, and Software itself is a commodity resource which is allocated to tenants as a service.

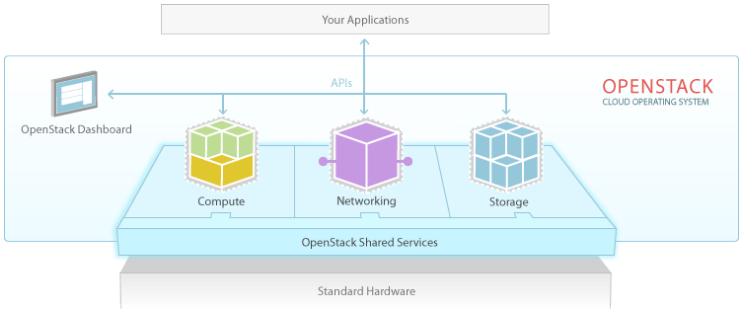

Figure 2.1 – Openstack Infrastructure as a Service [2]

To concretely visualize a Cloud IaaS and render the idea of commodity infrastructure let us look at Fig. 2.1. We observe that the base layer of the system is made of a pool of Standard Hardware that will handle the Cloud load. The Openstack [2] Infrastructure as a Service middle-ware abstracts these real hardware resources though virtualization in order to provide a multi-tenant pool of on-demand virtual resources. As mentioned in this section the resources are easily allocated through a software dashboard and devices can be allocated on-demand. The management of scheduling policies of the virtual resources on the real ones is made transparently to the user, by the Cloud provider. Continuing on this section we discuss both clouds and applications that span outside one data center.

2.2.1. Deployment Models

Clouds can be classified as Private (when Cloud infrastructure is deployed on premises for each organizations), Public (Cloud provider owns the virtual or hardware infrastructure), Hybrid (Cloud Infrastructure is deployed both on premises and as well as on public cloud deployments)[20, 21]. These different deployment models are justified by different application requirements from the cloud clients, the on premise cloud deployment or a private cloud deployment can be customized in order to get augmented security, enhanced latency profiles, and other requirements. On the other hand a private deployment would be more costly both in term of resource cost (electricity, cooling), availability and related cost, and maintenance. Public Cloud deployments provide benefits of economy of scale, but present much more exposures to security breaches, higher latencies than the on-premise deployments and generally offer a moderately customisable solution.

When an application (or the same Cloud virtual infrastructure) requires more capacity or locality than what can be provided by the resources of one data center or Cloud provider, its resources are distributed across multiple Cloud providers. In such cases the

Cloud environment is a *Inter-Cloud* and spans beyond the borders of the same data center. By moving the Cloud beyond data center boundaries, new challenges are introduced due to multiple administrative boundaries for each data center/Cloud, increase in network latency between data centers and multiple cloud metering. Applications that need enhanced locality to users such as live streaming or real-time communications and have relatively small application state can be greatly enhanced by moving the application closer to the end users.

Crucial to this work are Inter-Cloud deployments, made of scenarios in which scale or enhanced locality are of the utmost importance and as such can't be accommodated in the premises of one data center. Inter-Clouds can be either of type *Multi-Cloud* (where Cloud providers do not explicitly collaborate to provide a Cloud service, but the Cloud client is responsible for allocating resources on all the Clouds) or Cloud Federation (where multiple Cloud providers cooperate and provide a seamless Cloud Service offer). These two Inter-Cloud types provide different challenges both in terms of automatic management, scale, requirements and metering. In Fig.2.2 we observe a Inter-Cloud deployment in order to visualize the multi-tenant, multi-premise nature of such model and the difference between Cloud Federation and Multi-Cloud.

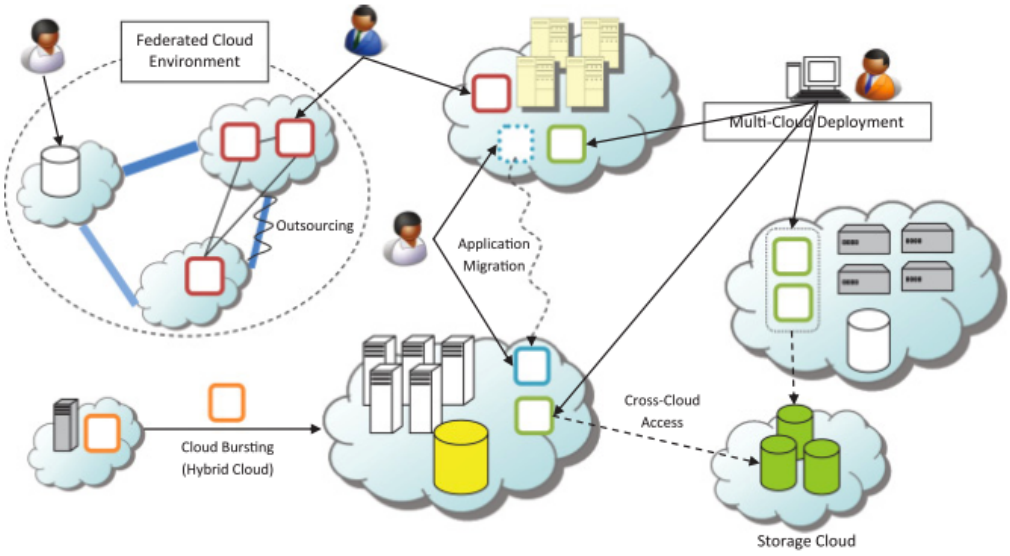

Figure 2.2 – Inter-Cloud Scenario [3]

2.3. Carrier Architecture

Carrier and ISP Providers generally offer as core business inter-connectivity both in the form of digital and analog communications. In this section we review the architecture of 3GPP[16] compliant network architecture. This architecture is the main focus of a

hybrid cloud architecture introduced in Chapter. 5 where network infrastructure and the Cloud are intertwined in order to provide enhanced locality to applications. The components of a traditional Carrier network are presented in Fig.2.3, where the *Core Network* generally a data center or data center grade hardware enabled facility is the main management, accounting and routing facility, while the signal reaches the end users though *Points of Presence*(PoP) located in user proximity, and each PoP in term is connected to the Core Network through broad band inter-connectivity or through data hubs called *Massive Multiple Input Multiple Ouput Network*. In general for the 3GPP case the PoP is the *Radio Base Station*, while for the ISP providers we generally have either WiFi access points or Fiber/Cable routers. The inter-connectivity between the PoPs and the Core Network is called the *Communication Backbone* and the bandwidth of it is shared between the PoPs. Depending on the number of PoPs allocated to each Backbone link, the aggregated bandwidth of End Clients that can connect to each PoP is greater than the capacity of the link. In this architecture routing facilities are designed to arbitrate the bandwidth in order not to saturate the Backbone which would lead to breaches of SLO and reduced user service quality. Generally for Carrier and ISP providers we have two types of traffics separated in their specific planes. The *Control Plane* is a secured communication channel, generally implemented on dedicated VPN or Physical connections with redundancy, is responsible for command traffic to manage and control the architecture. Whereas the *User Plane* are all the connectivity channels that provide connectivity to the End Users and is also the primary billable resource of a Carrier and ISP provider.

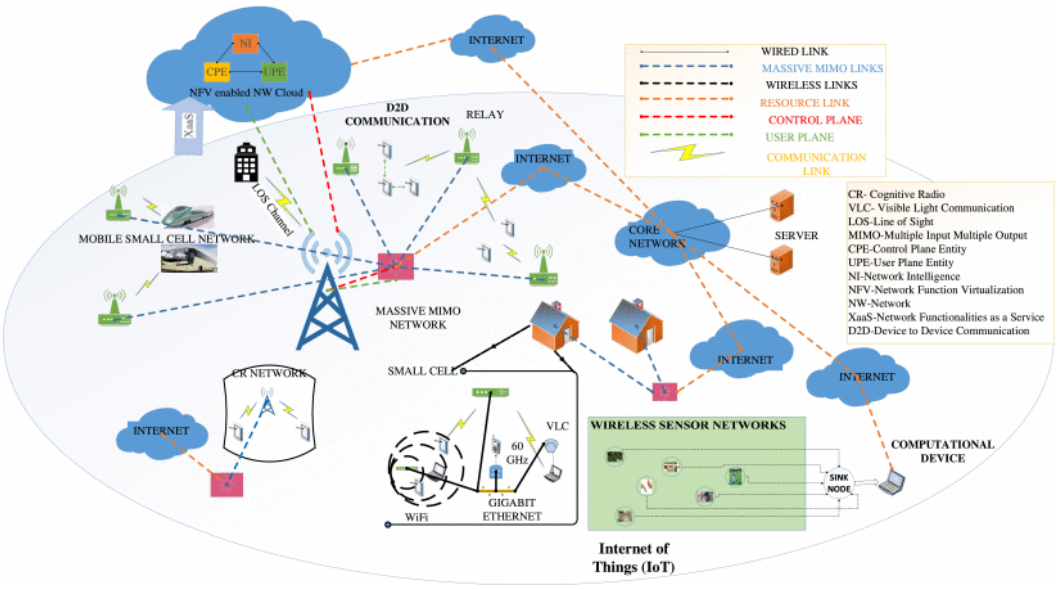

Figure 2.3 – 5G Network Architecture [4]

We observe in Fig. 2.3 the components of the next generation carrier networks 5G[4] and compare this properties with the 3GPP complaint Carrier architecture introduced

above. We see differences in what the PoPs are and as well on the Control Plane, but the overall architecture is not different than the 3GPP specification. The PoPs in the 5G architecture can be of different nature than just Radio Base Stations, we see an advance in access point technology and also non homogeneous means. PoPs can be *Small Cell*, *Mobile Small Cells*, *Wifi/WiMax*, *Wireless Sensors* etc... These new improvements to the existing Carrier network technology provide as well increase in Backbone bandwidth as well as extended geo-coverage through PoP diversity, thus augmenting the capacity and the reach of the network as well as the locality towards the End Users of the system. By enhancing such a highly distributed network architecture with Cloud enabled facilities we perceive higher scalability of services as well as better service performance in case that End User proximity and latency are crucial to service quality. In the work developed as part of this theses we develop a technological blueprint toward a hybrid Cloud-Network architecture that permits to exploit and integrate the benefits of both these architectures and complementing each other to bypass their respective limitations. In case of a purely Cloud environment locality to End Users is severely limited as the number of data centers in the world is far smaller and has far smaller reach than the Carrier and ISP providers, which reach virtually all Internet users. Whereas traditional Carrier and ISP networks always use the Core Network as main routing facilities and to reach cloud services users need to exit toward the Internet, thus loosing the benefits of geo-coverage provided by the network.

2.4. Mobile Cloud Computing

Mobile Cloud computing is a discipline in which thin clients offload computational resources to Cloud infrastructure, in order to extend the processing power of mobile devices or to gain additional benefits such as reduced power consumption. In [5] we observe that the access network is considered as static network and the cloud is reachable by the mobiles bu transversing the whole access network. Part of the system provided and displayed in Fig. 2.4 are the mobile clients and the Application service providers which implement the server side processing for the mobile applications. Another view of mobile cloud computing presented in [22] though Cloudlets, envisions distributing general computation units, in which mobile devices can offload computation, each computation unit is a Cloudlet.

In general mobile cloud computing is used to facilitate mobile device operations and related objectives of lower mobile battery consumption, extend computational power, lower overall energy overhead etc... Our extended Hybrid architecture provides a change in paradigm in which a uniform cloud architecture reaches up to one hop away from the user and thus instead of offloading computation to the Cloud, the services are the one to move closer to the End User of the applications in order to trade between locality and performance of the services.

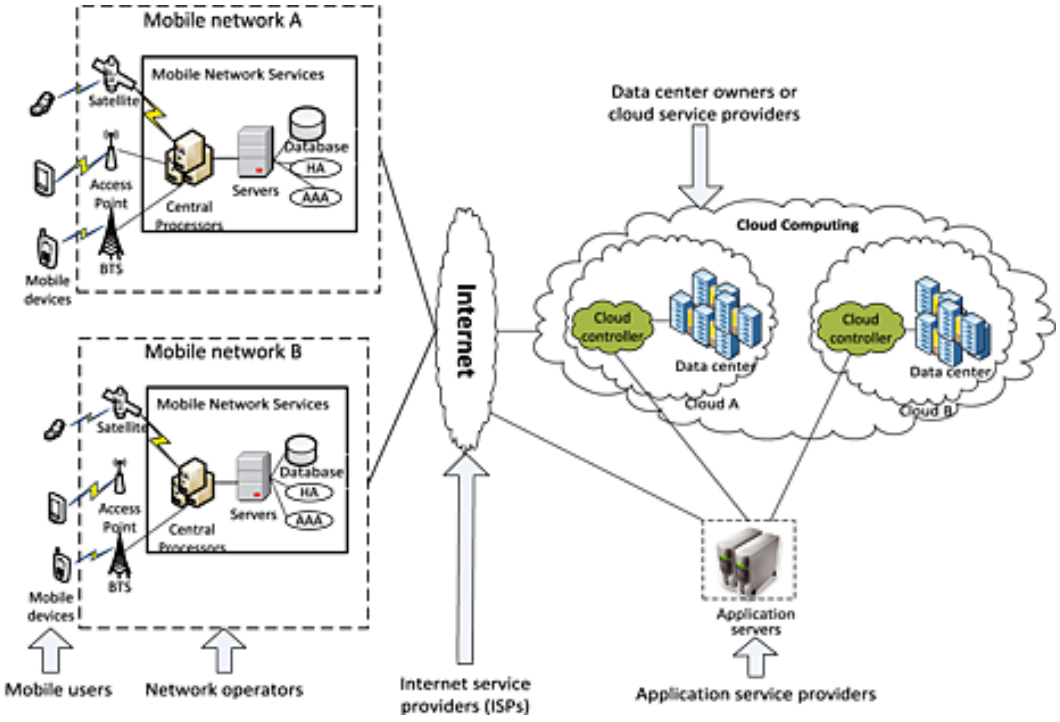

Figure 2.4 – Mobile Cloud Computing Architecture [5]

2.5. Live Streaming

With Streaming we refer to Audio and Video communications starting at a source (producer of the stream) and terminating at the *End Points* (End User destinations consuming the stream). In general the source captures Audio/Video from hardware devices and through digitalizing it sends it over the network to the intended destinations of the stream. When streaming is not live, means that the time of reproduction at the End Point is not related to the time of capture of the stream at the source. Whereas in the case of *Live Streaming* the stream is non-delay tolerant and need to be reproduced at the destination with minimal latency as compared to the time of capture. The live version of streaming makes it of utmost priority to handle both communication latency and jitter from source to destinations as well as mechanisms to handle packet loss and available bandwidth. Traditional caching techniques can't be used as that would introduce higher latencies in the system, and routing of the streams in the communication network needs to be optimized to lower latency and inter-arrival time.

In Fig. 2.5 we show a live encoders architecture, which consumes the stream from the source, transforms it to various output formats and streams it to the various clients consuming the stream. In this case the processing backend does not handle just stream routing but adds additional processing to convert the streams to different formats thus making it easier for different clients to consume it. Even though this approach gives higher

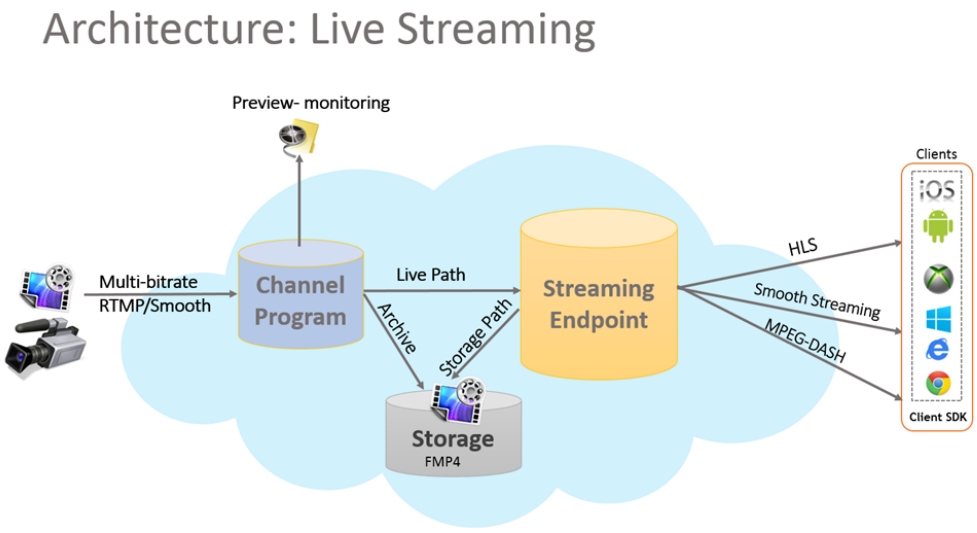

Figure 2.5 – Microsoft Azure Live Streaming Architecture [6]

client coverage, it impacts the overall stream quality as online transcoding is quite a heave processing task and may introduce fatal stream delays. Additionally other architectures can use Peer-to-Peer connections between clients instead of using a Cloud backend in order to route or convert the stream before sending it to the intended destinations. In this work we will discuss both architectures as well as optimized routing through cloud forwarders (in Chapter. 6) that are allocated to optimize such system parameters.

2.6. WebRTC Media parameters and Workloads

Web Real Time Communications WebRTC/RTCWEB is the leading standard for real-time communications in the web and standardized in [23]. The WebRTC/RTCWEB standard defines the protocols and data structures that enable media communications over the World Wide Web in a platform-agnostic way. All the latest versions of the major browsers at the time of writing support the standard (Chrome, Mozilla, Edge, Safari) and implement the WebRTC/RTCWEB software stack. The software stack implemented already by such browsers implements a layered design in which the browser already provide various functionalities needed to handle media communications. The WebRTC/RTCWEB stack implementations take care of low level media source access, video encoding at variable bit rates, audio encoding at variable bit rates, extended media functionalities such as noise cancellation/suppression, bit rate estimator. The encoding facilities provide various encoding protocol like **VP8**, **VP9**, **h.264** and **h.265** for video and OPUS for audio. When referring to the term **Media** as a general term, we refer to both audio and video, and specify in this work audio or video when context is needed. In

general we focus on video traffic, as the volume is far bigger than audio and thus impacts more the quality of service.

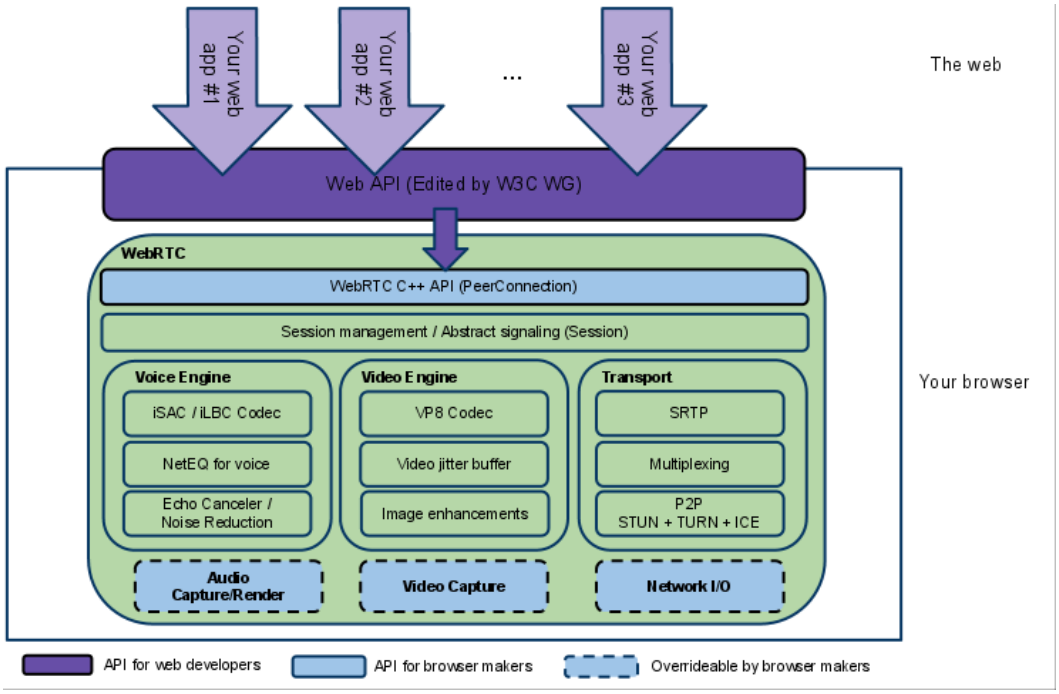

Figure 2.6 – WebRTC Software Stack [7]

In Fig. 2.6 we can observe the full software stack that need to be implemented in order to be compatible with the WebRTC/RTCWEB standard. As mentioned most of the browser or devices that qualify to be compatible to the standard need to implement all of the software layers. WebRTC compliant devices are not limited to browser but can be physical devices that implement the software stack. By implementing the standard non-homogeneous devices can communicate seamlessly with each other. The standard includes both P2P and Cloud based communications, as purely P2P solutions would be limited and may not meet latency requirements as well as bandwidth needed to handle multi-party communications. Each media interconnection between two end users is called a **Stream** and each stream has a source and a destination. In case of multiple destinations while using cloud forwarders the forwarding unit duplicates the stream for the source so the upload bandwidth on the source is reduced.

Of particular importance is the cloud backend that supports multi-party communications, such backend is composed of multiple clouds and constitutes a Inter-Cloud distributed application. In order to optimize latency toward the end users (End Point in the WebRTC/RTCWEB world) the backend needs to allocate stream forwarders in Clouds that are close to the user. In this work we will examine the impact of stream load of Cloud Forwarders toward stream quality. The forwarding units can both transcode

the stream online, or just select the best quality to forward for each stream destination in case that the source encodes multiple media qualities.

In this work we examine the correlation of stream load for cloud backend and stream media parameter quality. The core parameter that is further explored is video media quality, as the higher volume traffic concerning WebRTC/RTCWEB. A direct indicator of video media quality is video bit rate, for a given encoder technology and a given target resolution the optimal bit rate can be estimated. Deviation from the estimated optimal bit rate mean either quality loss in case lower bit rate than the optimal, while the opposite leads to superfluous traffic and higher costs. By using video bit rate as a indicator of stream quality we can observe interesting patterns that may emerge and relative correlations.

By exploiting patterns that correlate media quality and cloud backend load in terms of streams we can ensure that the system always meets SLOs and also optimizes quality for the end users. As previously introduced, the WebRTC Cloud backend spans multiple servers and data centers and as such algorithms need to be provided for data center wide stream allocation and multi-data center scenarios. This work provides some insight into temporal patterns of media quality and stream load per server/data center, as well as some allocation policies that exploit the patterns that were found.

2.7. General Concepts

2.7.1. Control theory

Control Theory is an engineering method that deals with the behaviour of complex systems and modifying such behaviour based on inputs, feedback mechanisms or error prediction. Generally we separate the control models in open loop and closed loop. When the model is based on open loop the controller does not account for outputs to modify the behaviour of the system. While on a closed loop control model, the feedback from the system outputs is taken in consideration in order to modify the system behaviour. The controller tries to modify the behaviour of the system in order to achieve a desired state based on past/current system inputs and outputs.

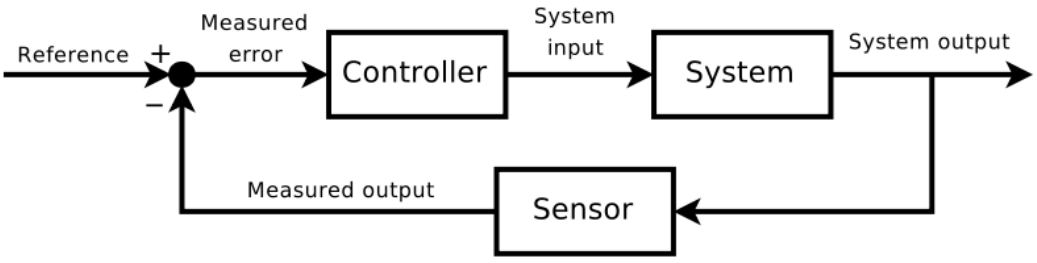

Figure 2.7 – Negative Feedback Loop control state

Fig. 2.7 shows the control cycle for a Negative Feedback loop controller in which the

output of the system give a negative feedback signal to the controller in order to adapt the system behaviour toward the specified goal. A particular case of controller that we use during this thesis work is the MAPE-K controller which separate decision making in the following phases:

1. **Monitor** Monitoring the current system state as a base point.
2. **Analyze** Analyze the values perceived through the Monitoring phase.
3. **Plan** Plan actuation needed in order to reach control objectives.
4. **Execute** Execute the plan of actuation to modify the system behaviour.
5. **Knowledge** Update the view of the system so far based on the monitored values, plan and actuation taken.

The control loop in this case updates the control model after actuating changes to the system behaviour, as well as based on previous inputs.

2.7.2. Machine Learning

Machine learning is a sub-field of computer science in which applications learn without explicitly being programmed to. In the context of this work, we extract data from real or simulated systems and process them though machine learning algorithms in order to train prediction algorithms for control systems. Machine learning algorithms are used to fit a function or a data model to a data source. The area of interest for this work borrowing from machine learning is called *Regression*, and tackles the problem of approximating a function or a data model of multiple variables in order to predict future performance output of a system. By using regression a model can be trained either off-line or on-line in order to build control systems for distributed applications.

2.7.3. Time Series Analysis

A time series is composed of data sampled at regular time intervals, and the study of such data can lead to interesting patterns that impact the analysis. For this work we analyze time series in order to observe load distributions and also train machine learning models to approximate and predict interesting metrics of a system. By combining analytical techniques with time series analysis we can better understand the behaviour of a complex dynamical system.

2.7.4. Conclusions

This chapter introduces the basic general concepts needed to understand the work developed as part of this thesis. Thus we introduce the concepts of Cloud and Inter-Cloud computing, Inter-Cloud deployment models including Cloud Federation and Multi-Clouds, Network provider architectures according to the latest standard to date,

Real-Time Media Stream processing applications (the crucial focus of this work), and finishes with a descriptions of techniques used in building novel algorithms or to conduct simulation studies (time series analysis, machine learning, control theory). We continue next in Chapter. 3 with a deeper introduction on state of the art for each contribution and further more specific descriptions of the related problems and solutions.

Chapter 3

Related Work

3.1. Introduction

In order to understand the contributions of this work and to differentiate it from the state-of-the-art techniques used in the field, we introduce for each contribution the relative research environment. At first we present the relative work existing for BwMan, our bandwidth manager, in terms of evaluating both different topologies and control system and how the differentiate from our approach. Next we describe the necessary related works in the field of Structured cloud federations, and how our novel architecture introduces various benefits in terms of application non-functional concerns that are not present in other related Cloud architectures. Continuing further on this introduction we present various works related with the Live streaming optimization approach, that deals with live media streaming. We compare both P2P and content delivery approaches on live media streaming and we see how our distributed overlay provides benefits in terms of stability, scale and low management overhead. We conclude this chapter with a study of the related work in media quality management for WebRTC media stream processing applications and workload management for Cloud based selective stream forwarders. In Chapter.4 we start describing the contributions of this work in more in-depth description per argument.

3.2. BwMan

The benefits of network bandwidth allocation and management is well understood as it allows improving performance of distributed services, effectively and efficiently meeting SLOs and, as consequence, improving end-users' experience with the services. There are different approaches to allocate and control network bandwidths, including controlling bandwidth at the network edges (e.g., of server interfaces); controlling bandwidth allocations in the network (e.g., of particular network flows in switches) using the software defined networking (SDN) approach; and a combination of those. A bandwidth manager in the SDN layer can be used to control the bandwidth allocation on a per-flow basis directly on the topology achieving the same goal as the BwMan controlling

bandwidth at the network edges. Extensive work and research has been done by the community in the SDN field, such as SDN using the OpenFlow interface [24].

A typical work of controlling bandwidth allocation in the network is presented in Seawall [25]. Seawall uses reconfigurable administrator-specified policies to share network bandwidth among services and enforces the bandwidth allocation by tunnelling traffic through congestion controlled, point to multipoint, edge to edge tunnels. In contrast, we propose a simpler yet effective solution. We let the controller itself dynamically decide the bandwidth quotas allocated to each services through a machine learning model. Administrator-specified policies are only used for tradeoffs when the bandwidth quota is not enough to support all the services on the same host. Using machine learning techniques for bandwidth allocation to different services allows BwMan to support the hosting of elastic services in the cloud, whose demand on the network bandwidth varies depending on the incoming workload.

A recent work of controlling the bandwidth on the edge of the network is presented in EyeQ [26]. EyeQ is implemented using virtual NICs to provide interfaces for clients to specify dedicated network bandwidth quotas to each service in a shared Cloud environment. Our work differs from EyeQ in a way that clients do not need to specify a dedicated bandwidth quota, instead, BwMan will manage the bandwidth allocation according to the desired SLO at a minimum bandwidth consumption.

The theoretical study of the trade-offs in the network bandwidth allocation is presented in [27]. It has revealed the challenges in providing bandwidth guarantees in a Cloud environment and identified a set of properties, including min-guarantee, proportionality and high utilization to guide the design of bandwidth allocation policies.

Theoretical estimation of latency capability profiles for composite cloud services and SLO oriented specifications for such profiles is proposed in [28]. The proposed theoretical framework uses queue theory based calculus in order to estimate the overall system latency profile. This approach is feasible and models the latencies of composite Cloud services given extensive information from global system sensors. In our approach we differ in the sense that the user needs not know the service topology and the architecture in order to estimate system parameters. Also our main metric of concern is the system throughput and not latency profile of the application. We treat the system as a black box and introduce machine learning predictive models in order to approximate prediction of throughput performance.

Also we are not only estimating performance measures but also controlling such measures for the composite Cloud services. Their work describes possible QoS parameters that can be used in order to do network resource management while we construct a controller enabling network bandwidth run-time arbitration. We also consider bandwidth trade-offs among the services as normal operations in the management context and provide extensive experimental results that show the feasibility of our approach and the performance of the management engine.

3.3. Structured Cloud Federations

Mobile cloud computing uses the cloud in order to optimize mobile device experience, power consumption, resource availability. In this approach a mobile device could use computation offloading, by sending the computation to be executed into the cloud, in order to gain more available and powerful resources but also to optimize battery consumption [29]. Our approach enhances this discipline with a new cloud paradigm that exploits locality in order to have better support for Mobile Cloud Computing as the cloud is brought closer to the mobile devices providing higher access bandwidth and decreasing traffic on the backbone as all of the interactions can happen between the mobile device and the local cloud close to the points of presence.

Work conducted in [30] describes the development ad deployment of micro-data centers in a wireless network in order to optimize locality to users and supply cost effective access to cloud enabled resources to mobile users. Wireless technology connected in a mesh, is used to provide the inter-connectivity between end users and cloud services. This work differs from ours in deployment models as ours Cloud model does not only provide enhanced locality but as well the possibility to trade-off between performance and locality. This trade-off possibility is provided though the hierarchical configuration of the micro-cloud backbone, while [30] provides a wireless mesh model that focuses only on locality and cost. Our work additionally takes in consideration all the federation requirements that would be needed in order to implement a highly distributed large scale hierarchical Cloud.

Nano-DataCenter work has been conducted as well in [31] but the federation model is a P2P model in which the geo-distributed data centers collaborate to form a cloud for a specific application. The intended application is a video on-demand streaming application in which energy consumption is lowered as considering the Internet Cloud scenario. This work focuses on energy efficiency and not on latency/locality as well as limits the study of the proposed Cloud model to one application. Our approach provides a more generic Cloud model in which the federation permits to optimize either for locality of performance, thus focusing more on the operational aspect of the cloud than in energy efficiency.

3.4. Live Streaming

This work is based on previous studies in the field of Peer-to-Peer (P2P) live streaming, and enhances these approaches by creating a new hybrid architecture where the clouds build a self-regulated structure to enhance latency. Other preceding systems [32] [33] [34] based on P2P overlays try to minimize mostly bandwidth as in general peers have a limited bandwidth to dedicate to the stream. Differently in case of cloud federations such restrain remains valid but the available bandwidth is in orders of magnitudes higher and does not pose a limiting factor. In general the bandwidth of the backbone is build to match the access network bandwidth for the clients, as such by construct the limiting factor remains latency.

Gradient based approaches to P2P live streaming like Sepidar [32], GradientTv [33] and GLive [34] are used to build a gradient overlay between peers in order to optimize

bandwidth and also provide incentive mechanisms in order to deal with free-riders. Such approaches do not include the notion of cloud and also do not minimize the traffic on the communication backbone. By depending only on the existing resources of the peers, no new resources can be allocated to meet load changes. Our approach uses the same overlay type for the clouds in order to build a locality aware federation that can scale to meet load requirements and is not influenced by churn or startup delay. Effectively the cloud federation overlay is not coupled to user churn.

In our architecture cloud resources for different streams can be reused as in AnySee [35], in order to efficiently distribute the load. Cliquestream[36] and Climber[37] try to promote peers to a set of peers and to super peers respectively. This approach is still limited to bandwidth optimization, is unable to scale beyond the resources of the participating peers, and give no guarantees on latency SLAs.

The approach introduced in this work is also based on previous work in high performance Content Delivery Networks (CDNs). The approach described in the workings of Akamai CDNs [38] presents a tree based delivery system with statically pre-allocated nodes. In this work, apart from envisioning dynamically allocated resources, we also organize the servers in a self-organized hierarchy with no central point of failure. By using the overlay, the naming scheme does not need to be centralized as the resources can use distributed discovery in the overlay. In Akamai's case DNS is used as a naming scheme but this introduces a lot of overhead to manage such scheme and also modifies DNS to serve as real-time consensus.

Other approaches [39] [40] [41] try to organize super peers on top of CDN server, but they still focus on a Tree like or DHT base architecture in order to build a overlay of the CDNs, and also deploy only statically pre-allocated resources. Our approach permits to scale-up the system both vertically and horizontally in order to meet clients QoS and lower costs by providing a dynamic cloud federation. The self regulating overlay mechanism provides a more dynamic and locality aware environment that both Tree and DHT based routing.

3.5. Web Real-Time Communications

Previous work conducted in [42] deals with both performance aspects and bit-rate estimation algorithms, but the study is limited to a small controlled environment. The study concerns a modified receiver side rate control algorithm while in our case we provide insight on the behaviour of the rate control algorithm implemented by Google (Google Congestion Control (GCC) [43]) for Chrome operating both sender (packet loss based) and receiver side (delay based). We conduct empirical analysis on both rate controlled traffic with single layer encoding as well as with simulcast streams. In our work additional empirical evidence is provided, which proves that as the scale of a session increases simulcast outperforms single layer video encoding.

Work conducted in [43] [44] explore the behaviour of multiple rate-control algorithms and streaming properties, as well as introducing novel rate-control algorithms. Specifically [43] provides a modified GCC congestion control algorithm that optimizes

coexistance of TCP streams with UDP WebRTC streams, thus not taking in account the case of having Cloud forwarded traffic. Our empirical analysis shows the impact of rate control and Video encoding techniques on stream quality as well as correlations of Cloud based stream forwarder load and video media quality. Another difference worth mentioning is the size of the clients involved in the study, we sample data passively from our test data center and range in hundred of thousands of stream while the previous mentioned work focuses on active measurements of up to three clients.

Resource allocation for WebRTC in [45] leads to a Network Virtual Functions based cloud architecture to interconnect IP Media Subsystems and WebRTC. While in [46] there is a focus on providing generic APIs of multi-purpose components that use HTML technologies for web communications. The study does not include resource allocation nor does it include any reference to the impact of application scale towards media quality in real-time communications. In this work we don't just examine the impact of scale toward video media quality, but we devise allocation policies for stream Cloud Forwarders that try to minimize server load.

3.6. Conclusions

To conclude the state-of-the-art we provide a high level summary of the main differences between previous work and the work presented in this thesis. In doing this we present just a highlight of the novelty introduced and leave to the following chapters an in-depth description of the contributions.

Network bandwidth in [24], [25] conducts trade-off and allocation on a quota based system and focuses on controlling this parameters at the topology level, while our work with BwMan provides a machine learning model that performs such activity at the end nodes, lowering the complexity of the system with comparable results. Work conducted in [25] enforces network bandwidths though tunneling and thus increases the latency of the network stream which is exactly what we want to optimize as a log lasting objective of this work. In [26] bandwidth is not allocated in SLO and thus is fairly distributed, while our approach tries to maintain some specific SLO giving enhanced guarantees on the system. At last we build on top of work conducted in [27] in network guarantees and properties in order to operate bandwidth management in a shared network environment.

Regarding cloud architectures, the access networks in general are build with a clear separation of concerns in which the main objective is to guarantee connectivity to clients. In this network architectures the Cloud is not part of the architecture but is seen as a service provided by other mediums. In this study we see an opportunity in providing a common cloud interface for both an enhanced cloud enabled Network infrastructure and Resource Providers outside of the network infrastructure. This common interface permits to easily go beyond the scale of multiple data center, and deploy services up to one hop away from the user with minimal impact on infrastructure. Work in both [30][31] focus more in energy efficiency rather than performance or locality. Our work differs both in deployment model and in performance guarantees. In or hierarchical Cloud-Network model the services can have predictable performance as well as predictable system

guarantees. While the mentioned work does not provide a common interface and neither multi-service guarantees.

Work related to enhancement of live streaming more exactly in [32] [33] [34] [36] and Climber[37] use P2P systems in order to directly build a streaming backend between the peers participating in a live stream session. The limitations of these architectures are that there is low control over the backend resources and that the span of a streaming chain can't be predicted. Another drawback of these approaches is that by being P2P the resources they can provide to the live stream are not optimized and are of limited availability. On the contrary in [39] [40] [41] the backend is enhanced with servers of Super Peers but the arrangements of the streaming backbone does not capture locality. In our work we provide a self-managed system which has predictable performance and uses a gradient topology to organize the Cloud based backend. As we use a Cloud based backend the resources can be scaled to fit the overall size of the streaming session, contrary to the P2P cases where the system can't scale to fit requirements.

In dealing with real-time web communications or WebRTC/RTCWEB we have that previous work has been conducted in introducing congestion control mechanism for live media streaming [42], [43]) and study the impact of these protocol at small scale. Our Work focuses on the impact of large scale toward the quality perceived by the user, and at scale we discover with empirical proof that single layer encoding is outperformed by simulcast. Additionally we introduce a new Cloud simulator and load balancing algorithm tailored specifically for the WebRTC case. The proposed algorithm is based on load temporal patterns discovered by an analysis of real passive user measurements at scale.

Chapter 4

Bandwidth Manager for Stream Processing Services in the Cloud

4.1. Introduction

This work is based on a paper [47] produced in collaboration with other coauthors and my specific contributions were in developing part of the workload generator used to run the experiments, in analytic work to derive the control and regression models and in interpreting the results of the work. It is reported in this thesis as it is the first step taken toward exploring Cloud and Network architectures correlations and motivates the followup work towards a complete big picture of the entire work. After this introductory note the rest of this section introduces the work toward network bandwidth actuation in order to enhance Cloud applications sharing the same network infrastructure.

The flexibility of Cloud computing allows elastic services to adapt to changes in workload patterns in order to achieve desired Service Level Objectives (SLOs) at a reduced cost. Typically, the service adapts to changes in workload by adding or removing service instances (VMs), which for stateful services will require moving data among instances. The SLOs of a distributed Cloud-based service are sensitive to the available network bandwidth, which is usually shared by multiple activities in a single service without being explicitly allocated and managed as a resource. We present the design and evaluation of BwMan, a network bandwidth manager for elastic services in the Cloud.

BwMan predicts and performs the bandwidth allocation and tradeoffs between multiple service activities in order to meet service specific SLOs and policies. To make management decisions, BwMan uses statistical machine learning (SML) to build predictive models. This allows BwMan to arbitrate and allocate bandwidth dynamically among different activities to satisfy specified SLOs. We have implemented and evaluated BwMan for the OpenStack Swift store. Our evaluation shows the feasibility and effectiveness of our approach to bandwidth management in an elastic service. The experiments show that network bandwidth management by BwMan can reduce SLO violations in Swift by a factor of two or more.

This first part of the work is a cornerstone in which the idea of the duality between the Network and Cloud is shown to be substantial and convertible in tangible performance metrics. We observe that network, latency and the Cloud services that share the same underlying network infrastructure correlate to give the overall system performance. When network is not optimized Cloud services would suffer from the reduced connectivity while at the same time if Cloud services are not properly deployed for a given network infrastructure, the performance of the service is impacted. Cloud computing with its pay-as-you-go pricing model and illusion of the infinite amount of resources drives our vision on the Internet industry, in part because it allows providing elastic services where resources are dynamically provisioned and reclaimed in response to fluctuations in workload while satisfying SLO requirements at a reduced cost. When the scale and complexity of Cloud-based applications and services increase, it is essential and challenging to automate the resource provisioning in order to handle dynamic workload without violating SLOs. Issues to be considered when building systems to be automatically scalable in terms of server capabilities, CPU and memory, are fairly well understood by the research community and discussed in literature, e.g., [48, 49, 50]. There are open issues to be solved, such as efficient and effective network resource management.

In Cloud-based systems, services, and applications, network bandwidth is usually not explicitly allocated and managed as a shared resource. Sharing bandwidth by multiple physical servers, virtual machines (VMs), or service threads communicating over the same network, may lead to SLO violations. Furthermore, network bandwidth can also be presented as a first class managed resource in the context of Internet Service Provider (ISP), inter-ISP communication, Clouds as well as community networks [51], where the network bandwidth is the major resource.

In our work, we demonstrate the necessity of managing the network bandwidth shared by services running on the same platform, especially when the services are bandwidth intensive. The sharing of network bandwidth can happen among multiple individual applications or within one application of multiple services deployed in the same platform. In essence, both cases can be solved using the same bandwidth management approach. The difference is in the granularity in which bandwidth allocation is conducted, for example, on VMs, applications or threads. In our work, we have implemented the finest bandwidth control granularity, i.e., network port level, which can be easily adapted in the usage scenario of VMs, applications, or services. Specifically, our approach is able to distinguish bandwidth allocations to different ports used by different services within the same application. In fact, this fine-grained control is needed in many distributed applications, where there are multiple concurrent threads creating workloads competing for bandwidth resources. A widely used application in such scenario is distributed storage service.

A distributed storage system provides a service that integrates physically separated and distributed storages into one logical storage unit, with which the client can interoperate as if it is one entity. There are two kinds of workload in a storage service. First, the system handles dynamic workload generated by the clients, that we call *user-centric workload*. Second, the system tackles with the workload related to system maintenance including load rebalancing, data migration, failure recovery, and dynamic reconfiguration (e.g., elasticity). We call this workload *system-centric workload*.

In a distributed storage service, the user-centric workload includes access requests issued by clients; whereas the system-centric workload includes the data replication, recovery, and rebalance activities performed to achieve and to ensure system availability and consistency. Typically the system-centric workload is triggered in the following situations. At runtime, when the system scales up, the number of servers and the storage capacity is increased, that leads to data transfer to the newly added servers. Similarly, when the system scales down, data need to be migrated before the servers are removed. In another situation, the system-centric workload is triggered in response to server failures or data corruptions. In this case, the failure recovery process replicates the under-replicated data or recover corrupted data. Rebalance and failure recovery workloads consume system resources including network bandwidth, thus may interfere with user-centric workload and affect SLOs.

From our experimental observations, in a distributed storage system, both user-centric and system-centric workloads are network bandwidth intensive. To arbitrate the allocation of bandwidth between these two kinds of workload is challenging. On the one hand, insufficient bandwidth allocation to user-centric workload might lead to the violation of

SLOs. On the other hand, the system may fail when insufficient bandwidth is allocated for data rebalance and failure recovery [48]. To tackle this problem, we arbitrate network bandwidth between user-centric workload and system-centric workload in a way to minimize SLO violations and keep the system operational.

We propose the design of BwMan, a network bandwidth manager for elastic Cloud services. BwMan arbitrates the bandwidth allocation among individual services and different service activities sharing the same Cloud infrastructure. Our control model is built using machine learning techniques [52]. A control loop is designed to continuously monitor the system status and dynamically allocate different bandwidth quotas to services depending on changing workloads. The bandwidth allocation is fine-grained to ports used by different services. Thus, each service can have a demanded and dedicated amount of bandwidth allocation without interfering among each other, when the total bandwidth in the shared platform is sufficient. Dynamic and dedicated bandwidth allocation to services supports their elasticity properties with reduced resource consumption and better performance guarantees. From our evaluation, we show that more than half of the SLO violations is prevented by using BwMan for an elastic distributed storage deployed in the Cloud. Furthermore, since BwMan controls bandwidth in port granularity, it can be easily extended to adapt to other usage scenarios where network bandwidth is a sharing resource and creates potential bottlenecks.

In this work, we build and evaluate BwMan for the case of a data center LAN topology deployment. BwMan assumes that bandwidth quotas for each application is given by data center policies. Within a limited bandwidth quota, BwMan tries to utilize it in the best way, by dividing it to workloads inside the applications. Specifically, BwMan arbitrates the available inbound and outbound bandwidth of servers , i.e., bandwidth at the network edges, to multiple hosted services; whereas the bandwidth allocation of particular network flows in switches is not under the BwMan control. In most of the deployments, control of the bandwidth allocation in the network by services might not be supported.

The contributions of this work are as follows.

- First, we propose a bandwidth manager for distributed Cloud-based services using predictive models to better guarantee SLOs.
- Second, we describe the BwMan design including the techniques and metrics of building predictive models for system performance under user-centric and system-centric workloads as a function of allocated bandwidth.
- Finally, we evaluate the effectiveness of BwMan using the OpenStack Swift Object Storage.

The rest of the chapter is organized as follows. In Section 4.2, we describe the background for this work. Section 4.3 presents the control model built for BwMan. In Section 4.4, we describe the design, architecture, and work-flow of BwMan. Section 4.5 shows the performance evaluation of the bandwidth manager. We conclude in Section 4.6.

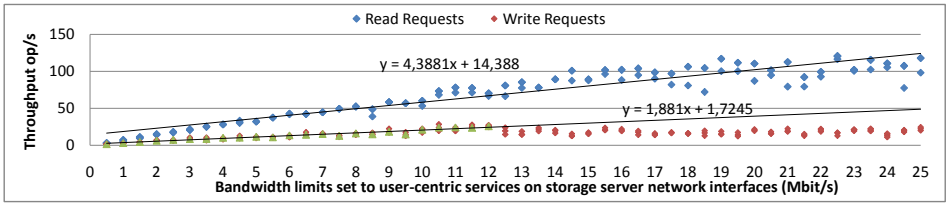

Figure 4.1 – Regression Model for System Throughput vs. Available Bandwidth

4.2. OpenStack Swift

A distributed storage service provides an illusion of a storage with infinite capacity by aggregating and managing a large number of storage servers. Storage solutions [53, 54, 55, 56] include relational databases, NoSQL databases, distributed file systems, array storages, and key-value stores. In this work, we consider an object store, namely OpenStack Swift, as a use case for our bandwidth management mechanism. Swift follows a key-value storage style, which offers a simple interface that allows to put, get, and delete data identified by keys. Such simple interface enables efficient partitioning and distribution of data among multiple servers and thus scaling well to a large number of servers. Examples of key-value storages are Amazon S3, OpenStack Swift, Cassandra [53] and Voldemort [54]. OpenStack Swift, considered in this work, is one of the storage services of OpenStack Cloud platform [55]. In Swift, there are one or many Name Nodes (representing a data entry point to the distributed storage) that are responsible for the management of the Data Nodes. Name Nodes may store the metadata describing the system or just be used as access hubs to the distributed storage. The Name Nodes may also be responsible for managing data replication, but leave actual data transfer to the Data Nodes themselves. Clients access the distributed storage through the Name Nodes using a mutually agreed upon protocol and the result of the operation is also returned by the same Name Node. Despite Name Nodes and Data Nodes, Swift consists of a number of other components, including Auditors, Updators and Replicators, together providing functionalities such as highly available storage, lookup service, and failure recovery. In our evaluation, we consider bandwidth allocation tradeoffs among these components.

4.3. Predictive Models of the Target System

BwMan bandwidth manager uses easily-computable predictive models to foresee system performance under a given workload in correlation to bandwidth allocation. As there are two types of workloads in the system, namely user-centric and system-centric, we show how to build two predictive models. The first model defines correlation between the user-oriented performance metrics under user-centric workload and the available bandwidth. The second model defines correlation between system-oriented performance metrics under system-centric workload and the available bandwidth.

We define user-oriented performance metrics as the system throughput measured in read/write operations per second (op/s). As a use case, we consider the system-centric

workload associated with failure recovery, that is triggered in response to server failures or data corruptions. The failure recovery process is responsible to replicate the under-replicated data or recover corrupted data. Thus, we define the system-oriented performance metrics as the recovery speed of the corrupted data in megabyte per second (MB/s). Due to the fine-grained control of network traffic on different service ports, the bandwidth arbitration by BwMan will not interfere with other background services in the application, such as services for failure detection and garbage collection.

The mathematical models we have used are regression models. The simplest case of such an approach is a one variable approximation, but for more complex scenarios, the number of features of the model can be extended to provide also higher order approximations. In the following subsections, we show the two derived models.

4.3.1. User-oriented Performance versus Available Bandwidth

First, we analyze the read/write (user-centric) performance of the system under a given network bandwidth allocation. In order to conduct decisions on bandwidth allocation against read/write performance, BwMan uses a regression model [49, 50, 57] of performance as a function of available bandwidth. The model can be built either off-line by conducting experiments on a rather wide (if not complete) operational region; or on-line by measuring performance at runtime. In this work, we present the model trained off-line for the OpenStack Swift store by varying the bandwidth allocation and measuring system throughput as shown in Fig. 4.1. The model is set up in each individual storage node. Based on the incoming workload monitoring, each storage node is assigned with demanded bandwidth accordingly by BwMan in one control loop. The simplest computable model that fits the gathered data is a linear regression of the following form:

$$Throughput[op/s] = \alpha_1 * Bandwidth + \alpha_2 \tag{4.1}$$

For example, in our experiments, we have identified the weights of the model for read throughput to be $\alpha_1 = 4.388$ and $\alpha_2 = 14.38$. As shown in Fig. 4.1, this model approximates with a relatively good precision the predictive control function. Note that the second half of the plot for write operations is not taken into consideration, since the write throughput in this region does not depend on the available bandwidth since there are other factors, which might become the bottlenecks, such as disk write access.

4.3.2. Data Recovery Speed versus Available Bandwidth

Next, we analyse the correlation between system-centric performance and available bandwidth, namely, data recovery speed under a given network bandwidth allocation. By analogy to the first model, the second model was trained off-line by varying the bandwidth allocation and measuring the recovery speed under a fixed failure rate. The difference is that the model predictive process is centrally conducted based on the monitored system data integrity and bandwidth are allocated homogeneously to all storage servers. For the moment, we do not consider the fine-grained monitor of data integrity on each storage node. We treat data integrity at the system level.

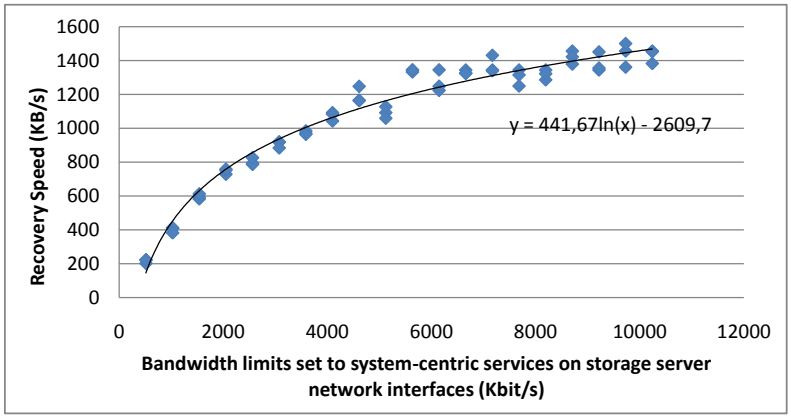

Figure 4.2 – Regression Model for Recovery Speed vs. Available Bandwidth

The model that fits the collected data and correlates the recovery speed with the available bandwidth is a regression model where the main feature is of logarithmic nature as shown in Fig. 4.2. The concise mathematical model is

$$RecoverySpeed[MB/s] = \alpha_1 * ln(Bandwidth) + \alpha_2 \tag{4.2}$$

Fig. 4.2 shows the collected data and the model that fits the data. Specifically, in our case, the weights in the logarithmic regression model are $\alpha_1 = 441.6$ and $\alpha_2 = -2609$.

4.4. BwMan: Bandwidth Manager

In this section, we describe the architecture of BwMan, a bandwidth manager which arbitrates bandwidth between user-centric workload and system-centric workload of the target distributed system. BwMan operates according to the MAPE-K loop [58] (Fig. 4.3) passing the following phases:

- Monitor: monitor user-defined SLOs, incoming workloads to each storage server and system data integrity;
- Analyze: feed monitored data to the regression models;
- Plan: use the predictive regression model of the target system to plan the bandwidth allocation including tradeoffs. In the case when the total network bandwidth has been exhausted and cannot satisfy all the workloads, the allocation decisions are made based on specified tradeoff policies (explained in Section 4.4.2);
- Execute: allocate bandwidth to sub-services (storage server performance and system failure recovery) according to the plan.

Control decisions are made by finding correlations through data using two regression models (Section 4.3). Each model defines correlations between a specific workload (user-centric or system-centric) and bandwidth.

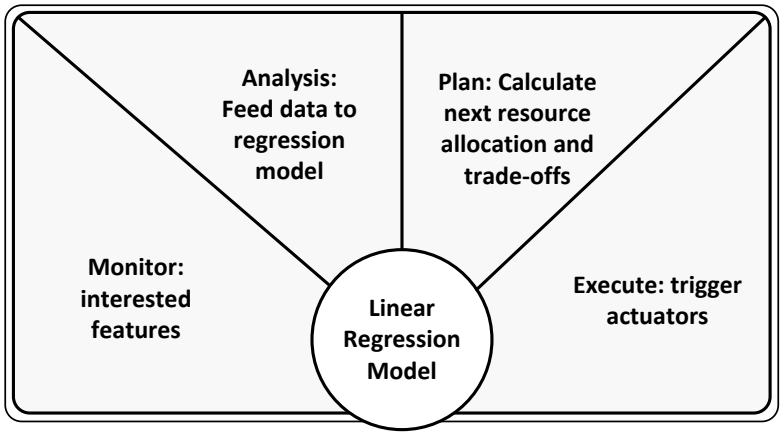

Figure 4.3 – MAPE Control Loop of Bandwidth Manager

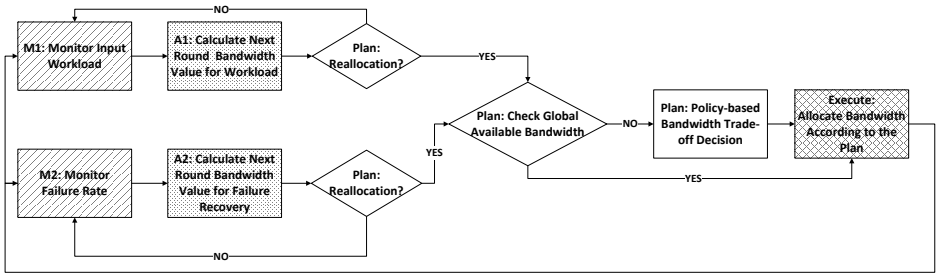

Figure 4.4 – Control Workflow

4.4.1. BwMan Control Work-flow

The flowchart of BwMan is shown in Fig. 4.4. BwMan monitors three signals, namely, user-centric throughput (defined in SLO), the workload to each storage server and data integrity in the system. At given time intervals, the gathered data are averaged and fed to analysis modules. Then the results of the analysis based on our regression model are passed to the planning phase to decide on actions based on SLOs and potentially make tradeoff decision. The results from the planning phase are executed by the actuators in the execution phase. Fig. 4.4 depicts the MAPE phases as designed for BwMan. For the Monitor phase, we have two separate monitor ports, one for user-centric throughput (M1) and the other one for data failure rates (M2). The outputs of these stages are passed to the Analysis phase represented by two calculation units, namely A1 and A2, that aggregate and calculate new bandwidth availability, allocation and metrics to be used during the Planning phase according to the trained models (Section 4.3). The best course of action to take during the Execution phase is chosen based on the calculated bandwidth necessary for user-centric workload (SLO) and the current data failure rate, estimated from system data integrity in the Planning phase. The execution plan may include also the tradeoff decision in the case of bandwidth saturation. Finally, during the Execution phase, the actuators are employed to

modify the current state of the system, which is the new bandwidth allocations for the user-centric workload and for the system-centric (failure recovery) workload to each storage server.

4.4.2. Tradeoff Scenario

BwMan is designed to manage a finite resource (bandwidth), so the resource may not always be available. We describe a tradeoff scenario where the bandwidth is shared among user-centric and system-centric workloads.

In order to meet specified SLOs, BwMan needs to tune the allocation of system resources in the distributed storage. In our case, we observe that the network bandwidth available for user-centric workload directly impact user-centric performance (request throughput). Thus, enough bandwidth allocation to the user-centric workload is essential to meet SLOs. On the other hand, system-centric workload, such as failure recovery and data rebalance, are executed in order to provide better reliability for data in a distributed storage. The rebalance and replication process moves copies of the data to other nodes in order to have more copies for availability and self-healing purposes. This activity indirectly limits user-centric performance by impacting the internal bandwidth of the storage system. While moving the data, the available bandwidth for user-centric workload is lowered as system-centric workload competes for the network bandwidth with user-centric workload.

By arbitrating the bandwidth allocated to user-centric and system-centric workloads, we can enforce more user-centric performance while penalizing system-centric functionalities or vice versa. This tradeoff decision is based on policies specified in the controller design.

The system can limit the bandwidth usage of an application by selecting the requests to process and those to ignore. This method is usually referred as admission control, which we do not consider here. Instead we employ actuators to arbitrate the bandwidth between user-centric workload and system-centric workload.

4.5. Evaluation

In this section, we present the evaluation of BwMan in OpenStack Swift. The storage service was deployed in an OpenStack Cloud in order to ensure complete isolation and sufficiently enough computational, memory, and storage resources.

4.5.1. OpenStack Swift Storage

As a case study, we evaluate our control system in OpenStack Swift, which is a widely used open source distributed object storage started from Rackspace [18]. We identify that, in Swift, user-centric workload (system throughput) and system-centric workload (data rebalance and recovery) are not explicitly managed. We observe that data rebalance and failure recovery mechanisms in Swift are essentially the same. These two services adopt a

set of replicator processes using the "rsync" Linux utility. In particular, we decide to focus on one of these two services: failure recovery.

4.5.2. Experiment Scenarios

The evaluation of BwMan in OpenStack Swift has been conducted under two scenarios. First, we evaluate the effectiveness of BwMan in Swift with specified throughput SLO for the user-centric workload, and failure rates that correspond to system-centric workload (failure recovery), under the condition that there is enough bandwidth to handle both workloads. These experiments demonstrate the ability of BwMan to manage bandwidth in a way that ensures user-centric and system-centric workloads with maximum fidelity.

Second, a policy-based decision making is performed by BwMan to tradeoff in the case of insufficient network bandwidth to handle both user-centric and system-centric workloads. In our experiments, we give higher priority to the user-centric workload compared to system-centric workload. We show that BwMan adapts Swift effectively by satisfying the user-defined SLO (desired throughput) with relatively stable performance.

4.5.3. Experiment Setup

Swift Setup

We have deployed a Swift cluster with a ratio of 1 proxy server to 8 storage servers as recommended in the OpenStack Swift documentation [19]. Under the assumption of uniform workload, the storage servers are equally loaded. This implies that the Swift cluster can scale linearly by adding more proxy servers and storage servers following the ratio of 1 to 8.

Workload Setup

We modified the Yahoo! Cloud Service Benchmark (YCSB) [59] to be able to generate workloads for a Swift cluster. Specifically, our modification allows YCSB to issue read, write, and delete operations to a Swift cluster with best effort or a specified steady throughput. The steady throughput is generated in a queue-based fashion, where the request incoming rate can be specified and generated on demand. If the rate cannot be met by the system, requests are queued for later execution. The Swift cluster is populated using randomly generated files with predefined sizes. The file sizes in our experiments are chosen based on one of the largest production Swift cluster configured by Wikipedia [60] to store static images, texts, and links. YCSB generates requests with file sizes of 100KB as like an average size in the Wikipedia scenario. YCSB is given 16 concurrent client threads and generates uniformly random read and write operations to the Swift cluster.

Failure Generator and Monitor

The injected file loss in the system is used to trigger the Swift's failure recovery process. We have developed a failure generator script that uniformly at random chooses a data node,

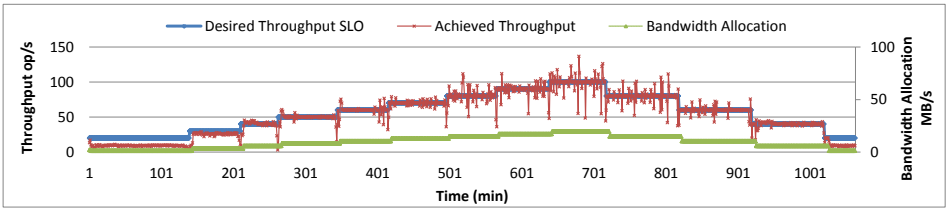

Figure 4.5 – Throughput under Dynamic Bandwidth Allocation using BwMan

in which it deletes a specific number of files within a defined period of time. This procedure is repeated until the requested failure rate is reached.

To conduct failure recovery experiments, we customized the swift-dispersion tool in order to populate and monitor the integrity of the whole data space. This customized tool functions also as our failure recovery monitor in BwMan by providing real-time metrics on data integrity.

The Actuator: Network Bandwidth Control

We apply NetEm's tc tools [61] in the token buffer mode to control the inbound and outbound network bandwidth associated with the network interfaces and service ports. In this way, we are able to manage the bandwidth quotas for different activities in the controlled system. In our deployment, all the services run on different ports, and thus, we can apply different network management policies to each of the services.

4.5.4. User-centric Workload Experiment

Fig. 4.5 presents the effectiveness of using BwMan in Swift with dynamic user-centric SLOs. The x-axis of the plot shows the experiment timeline, whereas the left y-axis corresponds to throughput in op/s, and the right y-axis corresponds to allocated bandwidth in MB/s.

In these experiments, the user-centric workload is a mix of 80% read requests and 20% write requests, that, in our view, represents a typical workload in a read-dominant application.

Fig. 4.5 shows the desired throughput specified as SLO, the bandwidth allocation calculated using the linear regression model of the user-centric workload (Section 4.3), and achieved throughput. Results demonstrate that BwMan is able to reconfigure the bandwidth allocated to dynamic user-centric workloads in order to achieve the requested SLOs.

4.5.5. System-centric Workload Experiment

Fig. 4.6 presents the results of the data recovery process, the system-centric workloads, conducted by Swift background process when there are data corruption and data loss in the system. The dotted curve sums up the monitoring results, which constitute the 1% random sample of the whole data space. The sample represents data integrity in the system with

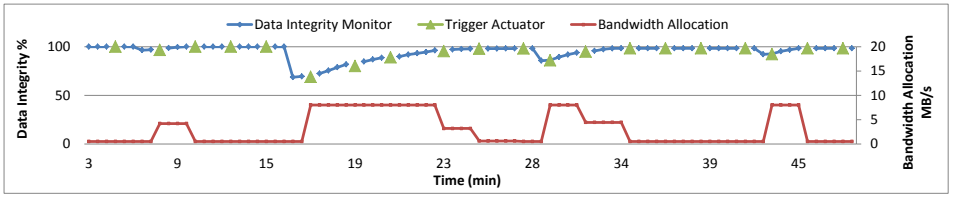

Figure 4.6 – Data Recovery under Dynamic Bandwidth Allocation using BwMan

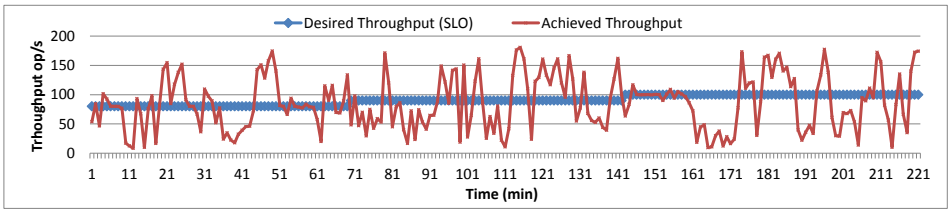

Figure 4.7 – Throughput of Swift without BwMan

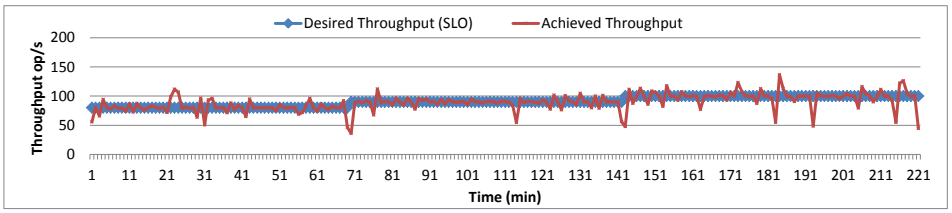

Figure 4.8 – Throughput of Swift with BwMan

max value at 100%. The control cycle activation is illustrated as triangles. The solid curve stands for the bandwidth allocation by BwMan after each control cycle. The calculation of bandwidth allocation is based on a logarithmic regression model obtained from Fig. 4.2 in Section 4.3.

4.5.6. Policy-based Tradeoff Scenario

In this section, we demonstrate that BwMan allows meeting the SLO according to specified policies in tradeoff decisions when the total available bandwidth is saturated by user-centric and system-centric workloads. In our experiments, we have chosen to give preference to user-centric workload, namely system throughput, instead of system-centric workload, namely data recovery. Thus, bandwidth allocation to data recovery may be sacrificed to ensure conformance to system throughput in case of tradeoffs.

In order to simulate the tradeoff scenario, the workload generator is configured to generate 80 op/s, 90 op/s, and 100 op/s. The generator applies a queue-based model, where requests that are not served are queued for later execution. The bandwidth is dynamically allocated to meet the throughput SLO for user-centric workload.

Fig. 4.7 and Fig. 4.8 depict the results of our experiments conducted simultaneously in the same time frame; the x-axis shares the same timeline. The failure scenario introduced

Table 4.1 – Percentage of SLO Violations in Swift with/out BwMan

SLO confidence interval	Percentage of SLO violation	
	With BwMan	Without BwMan
5%	19.5%	43.2%
10%	13.6%	40.6%
15%	8.5%	37.1%

by our failure simulator is the same as in the first series of experiments (see data integrity experiment in Fig. 4.6).

Fig. 4.7 presents the achieved throughput executing user-centric workload without bandwidth management, i.e., without BwMan. In these experiments, the desired throughput starts at 80 op/s, then increases to 90 op/s at about 70 min, and then to 100 op/s at about 140 min. Results indicate high presence of SLO violations (about 37.1%) with relatively high fluctuations of achieved throughput.

Fig. 4.8 shows the achieved throughput in Swift with BwMan. In contrast to Swift without bandwidth management, the use of BwMan in Swift allows the service to achieve required throughput (meet SLO) most of the time (about 8.5% of violation) with relatively low fluctuations of achieved throughput.

Table 4.1 summarizes the percentage of SLO violations within three given confidence intervals (5%, 10%, and 15%) in Swift with/out bandwidth management, i.e., with/out BwMan. The results demonstrate the benefits of BwMan in reducing the SLO violations with at least a factor of 2 given a 5% interval and a factor of 4 given a 15% interval.

4.6. Conclusion and Future Work

We have presented the design and evaluation of BwMan, a network bandwidth manager providing model-predictive policy-based bandwidth allocation for elastic services in the Cloud. For dynamic bandwidth allocation, BwMan uses predictive models, built from statistical machine learning, to decide bandwidth quotas for each service with respect to specified SLOs and policies. Tradeoffs need to be handled among services sharing the same network resource. Specific tradeoff policies can be easily integrated in BwMan.

We have implemented and evaluated BwMan for the OpenStack Swift store. Our evaluation has shown that by controlling the bandwidth in Swift, we can assure that the network bandwidth is effectively arbitrated and allocated for user-centric and system-centric workloads according to specified SLOs and policies. Our experiments show that network bandwidth management by BwMan can reduce SLO violations in Swift by a factor of two or more.

Finally, we demonstrate that the control of bandwidth allocation in a Cloud environment can be useful in managing and ensuring the performance of elastic services that share the same network infrastructure. This SLO-based control allows us to consider

network bandwidth as a first-class controllable resource for services in Cloud environment.

In our future work, we will focus on possible alternative control models and methodology of controller designs for multiple Cloud-based services sharing the network infrastructure in Clouds and Cloud federations. In addition, we will investigate impact of network topology and link capacities on the network bottlenecks within or between data centers, and how to integrate controlling bandwidth on edges of the network with bandwidth allocation and with allocation in the network topology using SDN approach.

Building on top of the idea presented in this work to use the network in order to influence overall behaviour of multiple services sharing the same network infrastructure, we try to push the idea further and create a new model to merge both the Network and the Cloud. The Two Tier Federate Cloud model presented in Chapter.5 presents exactly this notion, with a massively scaled Cloud architectural blueprint. The network providers are converted through a massively geo-distributed Cloud deployment in both connectivity provider and Cloud provider.

Chapter 5

Structured Cloud federation for Carrier and ISP infrastructure

5.1. Introduction

Previously in Chapter. 4 we provide a network bandwidth manager, that arbitrates bandwidth between end-points. Such approach shows that there is a strong correlation between services sharing the same network infrastructure and their performance. We continue to build on this work in order to provide a novel Cloud model in which applications can be deployed anywhere from traditional Internet Public Clouds to up to 1 hop away from the user, as such providing a generalized view of a highly distributed Cloud environment. The structured cloud application provided in this chapter, enables trade-offs between locality and performance and as well tries to minimize federation backbone traffic. The importance of providing federation requirements and a federation model is presented not only to provide a generalized Cloud model but also to provide additional guarantees of minimal management overhead and operations in times of network partitions. This work further builds toward a merged hybrid Cloud-Network model and respective structured federation providing incremental contributions toward a massively scaled and geo-distributed hierarchical Cloud.

Cloud Computing in recent years has seen enhanced growth and extensive support by the research community and industry. The advent of cloud computing realized the concept of commodity computing, in which infrastructure (resources) can be allocated on demand giving the illusion of infinite resource availability. The state-of-art Carrier and ISP infrastructure technology is composed of tightly coupled software services with the underlying customized hardware architecture. The fast growth of cloud computing as a vastly consolidated and stabilized technology is appealing to Carrier Providers in order to reduce Carrier deployment costs and enable a future of Carrier Clouds with easily accessible virtual carriers. For such migration to happen software services need to be generalized, to decouple hardware and software, and prepared to move into the Cloud.

The network backbone is centrally managed and only provides network connectivity; we believe this presents an opportunity. The edges of such networks and the core are interconnected with high performance links. If services could be deployed in these edges they would benefit from enhanced locality to the user. In this position work we propose a distributed cloud architecture (precisely a structured multi-cloud federated infrastructure), with minimal impact on existing infrastructure, as a first step to incorporate the Cloud into the network infrastructure of such providers, enabling and enhancing novel and existing applications.

Moreover the architecture of Carrier and ISP providers infrastructure is constructed in such a way that interconnections from the central data centers to the last-mile from the client is high speed and network bandwidth is then partitioned among the costumers in the last-mile. The whole backbone of these networks is managed in a centralized way and provides only network connectivity. In this work we propose a distributed cloud architecture (precisely a structured multi-cloud federated infrastructure), with minimal impact on existing technology, as a first step to incorporate the Cloud into the network infrastructure of such providers, enabling and enhancing novel and existing applications.

In both the cases of Mobile and ISP providers specialized high performance hardware can be deployed to this last-mile routing facility, which in case of mobile providers we

can consider the Base Stations as routing facilities and in case of ISP we can consider the quarter or building Routers. The deployment of cloud hardware so close to the user enables new possibilities for distributed services with the data centers being at the same time close to the user and highly inter-connected.

This work focuses on providing an architectural blueprint for structured multi-cloud federated infrastructure to enable and enhance novel and existing applications in the present Carrier and ISP architecture. A first architecture design and possible implementation blueprint is given. In the remaining of this work we will show also a series of use-cases that would benefit from such architecture design and properties that this architecture exposes toward service providers and client satisfaction. Cloud computing is a general term referring to the successor of GRID computing and is generally known as commodity computing. The concept of the Cloud [62] enables infrastructure to be allocated on demand and to be managed by software services. Virtualization techniques are used to give the impression of possibly infinite resource allocation, with virtual resources sharing the same real resource by use of time sharing or computation alternation in multi-core architectures.

The Cloud is generally seen as a stack model composed of the IaaS layer (Infrastructure as a Service) providing to the user virtual machines allocatable on demand, PaaS layer(Platform as a Service) providing abstractions of components and a language to manage the infrastructure and SaaS layer (Software as a Service) providing software capable of providing services to multiple organizations at the same time. Recently a new term is being used to provide the entire real resource as a commodity resource, Machine as a Service. This permits real specialized hardware to be allocated on-demand when needed. The users of these services optimize costs by only paying for the resources they need. Some of the most well known open source cloud management software at the time of writing are Openstack [2], OpenNebula [63], CloudStack [64].

Generally modern datacenters provide the users the ability to allocate virtual resources and real resources on demand, providing additional platforms to facilitate the development of new services. Some datacenters offer also the possibility to have storage as a platform, where the user can have scalable key-value stores, SQL-like databases etc. The whole system infrastructure is designed and deployed software wise, with no or limited knowledge of the real hardware that supports the virtual environment.

A multi-cloud federation system enables resource provisioning and life-cycle monitoring among different cloud providers. Users may need to access and create resources on multiple clouds for locality or economical advantages without penalizing their Quality of Services (QoS). A federated system handles cross-cloud interactions and integration in order to achieve higher levels of usability and locality. By combining together multiple cloud the quality of service can be augmented by diminishing the cost of resources since multiple providers means multiple choices and also by augmenting locality, since different parts of the service can be deployed in multiple data centers close to users of the service.

Infrastructure Providers (IP) are the organizations providing the hardware and software that make possible the operation of a Carrier System and Internet Provider system. Compared to the IP we will classify into Service Providers(SP), ISPs and Carrier

Providers as the organizations that use the infrastructure to provide network services. The architecture of the network backbone, maintained by these providers, is based on specialized hardware and follows a strictly centralized architecture, where the edges of the network provide access and the centralized Core Network provides services such as accounting, routing, etc. In this work we propose enriching these core and edge networks with our novel cloud architecture, enabling them to host cloud services. The proposed structured multi-cloud federation infrastructure is a first step in moving these providers into the cloud, and introduces a minimum technology impact on the existing infrastructure.

The rest of the chapter is organized as follows. Section 5.2 provides the necessary background information for both Cloud Infrastructure and Network Topologies. In Section 5.3, we describe a unified network model for Carrier and ISP networks. Section 5.4 presents the system design for both cloud deployment and federation model and properties. In Section 5.5, we describe different federation models and the model selected for our architecture. Section 5.7 shows possible applications based on this infrastructure. We conclude in Section 5.8.

5.2. Background

5.2.1. General Cloud Infrastructure

Cloud infrastructure management software are of various types and natures, at the time of writing of this work the mainstream open-source softwares are OpenStack, OpenNebula, Cloudstack. Companies providing cloud infrastructure mostly use open-source ranging from the enlisted open-source solutions or proprietary software. Some of the companies and big names in the cloud market are Amazon, Microsoft, VMware, etc.

Cloud management software are referred to as Virtual Infrastructure Manager (VIM) but this term is more adequate to describe local virtual management control components used for the local machine hypervisor. Yet another denomination found in the sites of cloud management open source software [2] refers to such components as a Cloud Operating System, which is easily confused with recent approaches to having the actual Operating System(OS) in the cloud. For the sake of this work we will refer to such management systems as the Cloud Infrastructure Manager (CIM).

The general architecture of a Cloud Infrastructure Manager is shown in Fig. 5.1. The main entities present in most of the CIM are the Cloud Controller, Virtual Infrastructure Manager (per node basis, or compute engine), Data Store Provider(DS) and Network Controller(NC). These components appear in different CIMs as a single component or as a family of components providing specialized behaviors, as an example Openstack has an Image Repository (Glance), a key-value store (Swift) and a block storage (Cinder) implementing different types of data stores.

Cloud Controller (CC) is the main orchestration entity, governing cloud orchestration for an entire cloud. Apart from cloud orchestration the CC handles also user management, security policies, resource scheduling, resource deployment, monitoring and billing. The CC is the core of the cloud managing infrastructure from a centralized point of view.

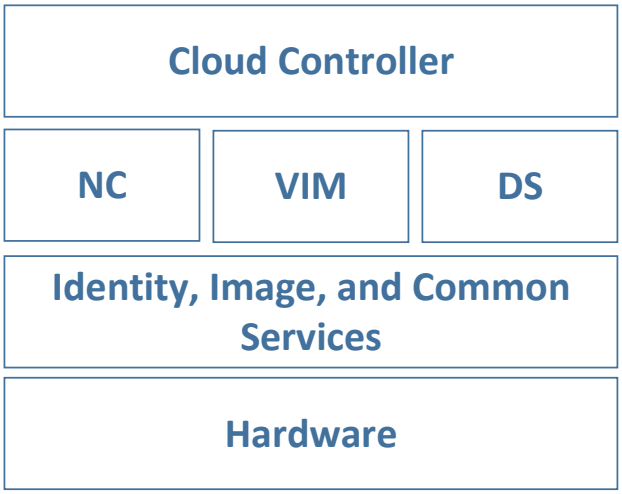

Figure 5.1 – Cloud Infrastructure Manager structure

Virtual Infrastructure Manager (VIM) is the component that manages local resources to a physical node in a cloud, by handling tasks such as VM scheduling, creation and monitoring. The virtual networking infrastructure is managed by the Networking Controller (NC) and defines software defined networks between various VMs in a datacenter. The whole setup is managed through the CC interface and the user need only take care of specifying a logical architecture which is then actuated by the Cloud Controller. The Data Store (DS) generally provides various abstractions of distributed storage for the cloud. All these components together provide a full cloud experience ready for service deployment.

The Data Store (DS) generally provides various abstractions of distributed storage for the cloud.The main type of data store implemented is the Image Repository (takes care of saving the images needed by the virtual machines to start operations). Other types of storage that some CIM software provide are also key-value store (for key-value databases) and block storage (virtual hardrives that can be plug-ed into VMs). All these components together provide a full cloud experience ready for service deployment.

5.2.2. Cloud Federation

Defining the term Cloud Federation is a difficult task, as in different cloud contexts it is used to represented different concepts. In general we have a federation when two or more administrative domains collaborate in order to achieve a common goal. In case of Cloud Federation, there are multiple types of federation possible depending on the type and the layer in the cloud stack in which federation is provided. The federation model we assume in this work is non transparent federation in which different sites have different Cloud Controllers and all of them know of each other, and collaborate through a Federation

Middleware.

Extrapolating on the above definition of federation model, a cloud is a self-sustained cloud entity with a whole cloud software stack deployment. This means that every entity has its own cloud controller, VIM, network controller and data store. The Federation of such a conglomerate of clouds is conducted through a central federation software that has the ability to access the clouds APIs transparently and orchestrates the different clouds in order to provide resources.

The clouds comprising such federation can be heterogeneous in nature as long as images are provided for every type of cloud or conversion utilities can be produced to translate the virtual images from one format to another. In the following sections we will discuss the nature and design choices for the cloud federation to be used on Carrier and ISP provider networks.

5.2.3. Carrier and ISP to the Cloud

Carrier and ISP providers generally follow a close market with software and hardware tightly coupled in order to get the best performance from both sides of the environment. Providing a generalized enough software stack for all the services of a provider in order to be allocated on demand may prove to be not feasible without rewriting most of the subsystems. There are also other limitations on Cloud adoption for the core system of such providers and these limitations are generally related to QoS concerns that can be only met by specialized hardware and software.

The possibility to integrate Machine as a Service(MaaS) into the cloud stack, cloud provide the means to allocate on demand specialized hardware as needed by the infrastructure. This in conjunction with generalized components allocated in the Cloud could provide a solution also to QoS concerns and generate a general enough architecture to deploy the whole Provider services on a cloud infrastructure. This possible solution requires a vast amount of work and modifications to the subsystems in order to migrate the services to the cloud. If the infrastructure providers would move their product to the cloud they would incur into vast amount of benefits, their data centers could accommodate and provide resources for multiple virtual Mobile Providers and ISP service providers and enable economy of scale but also energy benefits as resources would be managed dynamically following demand without necessity of over-provisioning to maintain SLA contracts. The current state-of-the art Carrier and ISP network services are not general enough to be moved to the cloud, and cloud technologies are not yet adopted in their infrastructure.

In this work we propose a different approach to integrate cloud services inside the existing network environment in use by these providers, in order to achieve better and novel services and also permit SPs and third parties to easily deploy services in the IP networks. The Cloud Federation that will be described in the next sections will use the nature and topology of the existing IP based Carrier and ISP networks.

5.3. Carrier and ISP System Topology

Carriers for Mobile and WiFi networks have a pretty close theoretical system topology to the ISP providers infrastructure. Some of the differences are in the protocols and the mediums of the last mile and the routing, else these two topologies are very affine to each other. In Fig. 5.2 we find a high level simplified star topology that unifies the different topologies and assumes IP as network transport protocol. The high level topology view simplifies dealing with different protocols at different stage of the carrier network and gives a much simpler framework to work with by decoupling the architecture from the actual communication protocols in the real topology. However there may be some protocol restrictions to the real topologies as some protocols may not include IP in all of the points of presence. This point will be a focus of further discussion when dealing with real deployments of such technology.

The subsystems of a normal Carrier and ISP provider are introduced in Fig. 5.2, where we have a main access to the internet protected by a firewall then we have the Core Network (CN), which compromises the main routing activities for data and voice traffic. The CN takes also care of the main activities like monitoring, provisioning of resources, accounting and also intra provider connectivity and handover. We assume we have one CN for each country which constitutes the main backbone of the infrastructure.

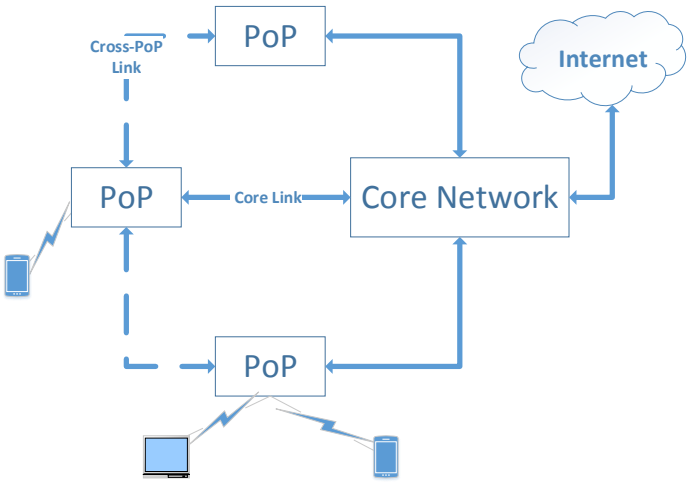

Figure 5.2 – High Level Provider Network Topology

In this generalized view of the architecture we have geo-distributed points of presence which provide network access to the clients. These points of presence are the edges of the access network and the one responsible to give connectivity to the clients. In case of the Carrier Network we have the Radio Base Stations providing local points of presence, while in the case of the ISP we have the Point of Presence (POP) which offer routing facilities and access and are connected with fiber to the CN.

The clients then connect to one of the POP with a different or same media but the bandwidth of the POP is subdivided between the clients that connect to the POP. In the case of Carrier networks, the transmission medium from POP to client is radio while in the case of ISP, it can either be radio (WiFi access points), cable or fiber. The total bandwidth toward the internet of the clients is less or equal to the bandwidth from the POP to the backbone. In general, different deployments are setup so that the total client bandwidth does not saturate the backbone link in order to permit also control signal bandwidth for inbound control. In the Carrier cases neighboring base stations have cross-connections that are used for traffic handover on user mobility. These links provide also a major way to optimize locality and present additional bandwidth that normally is used only in particular cases.

In general the topology (Fig. 5.2) is a star topology at the core, with core connections build up by high performance mediums (like fiber) and have lower bandwidth connections at the edges toward the client. In this work a multi-cloud distributed environment will be discussed in order to use the strong points of the topology and enable novel application and services to be deployed on the providers networks.

5.4. Structured Multi-Cloud Architecture

Section 5.3 introduces the abstract network topology for normal deployments of Carrier and ISP provider networks. The Star topology described previously as the main topology has the benefits of high speed links at the core and separate local bandwidth at the edges of the network (the clients). Let us discuss a small user-case to show potential benefits and give a realistic view on benefits brought by the new multi-cloud architecture.

In an ISP provider, if some services could be moved at the POP, the clients would have a full 100Mbs connection to such service, under the assumption of having cable as the last mile medium, but only a 2-10Mbs connection to the internet from the backbone. Assuming the required service from the user could be elastic enough to be moved at the POP the user could interact with the service at a far higher bandwidth than that of a cloud service positioned somewhere in a centralized datacenter reachable via the backbone. Thus we effectively augment the available bandwidth toward services from the user and also generate new bandwidth by using previously not usable bandwidth.

The simple case described in the previous paragraph introduces hints to a more optimized cloud environment in which the whole Carrier and ISP provider network could be transformed into a service enabled multi-cloud federation. Let us discuss in details the architecture design of such cloud and also various aspects of performance, control, usability, applications and stability of such architecture.

5.4.1. General Architecture

The proposed cloud architecture is constructed in order to take advantage of the network topology in order to achieve better services but also the possibility for both Carriers and ISP to enable in-house and third-party application and protocols to be deployed in a most secure and isolated manner. The deployment of such applications and protocols would not impact

the existing architecture and their deployment would be as easy as requesting the cloud manager to deploy some service images.

In the current state of the art such deployment would require provisioning of new hardware to the POPs or even software modifications to the POPs operating system which may lead to down-times if not done correctly and may require downtime in order to do the necessary system reconfiguration.

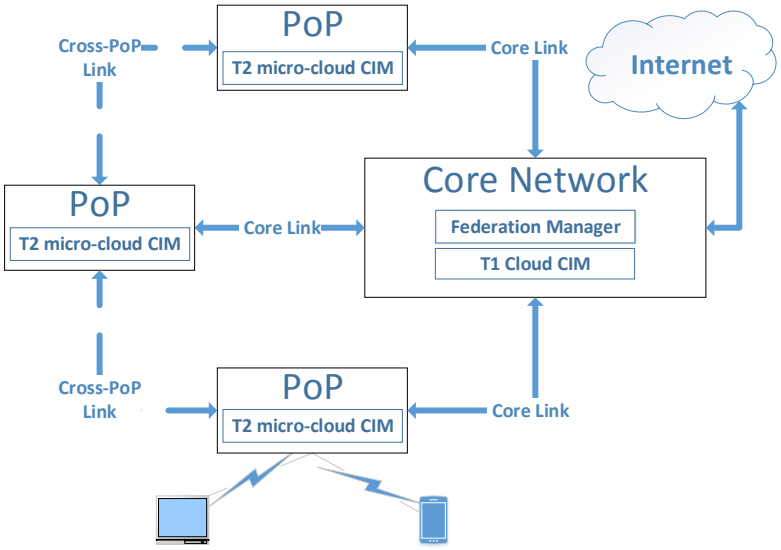

Figure 5.3 – Extended Infrastructure and Federation Model

The structured multi-cloud federated deployment architecture is shown in Fig. 5.3. The main aspects of this architecture are Tier-1 Cloud (T1, a datacenter grade cloud) and POP Tier2 (T2, POP local cloud). The combination of this two clouds permits a hierarchical cloud architecture that exploits locality and also the network topology for optimized bandwidth usage and sometimes also augmented bandwidth usage. This novel cloud architecture, could use local cross-POP links if present, that at the moment are used only for particular cases, in order to have full link usage and much more bandwidth than the normal POP bandwidth. This links in Carrier networks at the moment are used only for handover procedures.

The T1 cloud is a datacenter cloud deployment located, inside or highly connected to, the core network. For all intent and purposes, connection wise, it is an integral part of the core network and has high performance links to the internet. This core datacenter provides the necessary strong backbone in order to have a hybrid architecture. Data and application if deemed necessary can be moved around from the T2 cloud to the T1 cloud in order to have stronger Service Level Agreement (SLA) guarantees.

The T2 clouds are geographically distributed close to the POP or part of the POP OS itself, in which a VM acts as a routing facility, giving the possibility to allocate resources

on demand very close to the users of the service. In some cases bringing the user as close as one-hop distance from the desired service. Each POP has its own T2 micro-cloud and the conglomeration of all the micro-clouds builds the T2 distributed cloud layer. T2 micro clouds are self-managed full cloud deployments in order to have full support for application deployment.

The micro-clouds fabric is federated or managed by a centralized federation controller (CIM) installed in the T1 cloud. The federation type that is used is non-transparent federation which will be explained in describing possible control mechanisms of such a cloud concept.

Clients connecting to the WAN topology have generally higher bandwidth to the POP than the total client bandwidth toward the internet backbone which is limited by the backbone link from the POP to the CN. By having the T2 micro-clouds to the edges of the network the clients can interact with services with higher bandwidth and the total available bandwidth is increased as compared to the case in which the services are deployed on an internet cloud.

Since resources in the T2 cloud are limited we have the T1 cloud acting as a cloud helper in which load could be redirected in order to handle high traffic services which would saturate the resources in the T2 micro-clouds. In this case performance of the service in terms of locality and connectivity are limited as compared to the local micro-clouds, but still better than an internet cloud, since we are still inside the same WAN.

This architecture can further increase apparent total bandwidth if cross-POP links exist. This links can be used to shortcut application data between services running on neighbor POPs so that the backbone is left free for user internet access. In this case we have new data paths that normally are not used for user access becoming available to service data and thus increasing the efficiency of the network usage.

5.4.2. Bandwidth Control

The proposed structured multi-cloud architecture optimizes locality and efficiency of network links and also enables novel applications to be deployed on the described network topologies. The efficiency of the network is increased by rendering available to service and user traffic unused network bandwidth, that is normally not usable as limited by the network backbone and also new data paths that normally are used only for control or exceptional operations.

The efficiency comes with some restrictions as now data paths that were used for system specific tasks handle also user and cloud service traffic. The two types of traffic would be contending the same network resource and without some kind of management this may compromise system integrity.

We propose a solution based on previous work done in [65] a network bandwidth manager for cloud services. The principle following this network manager is in having bandwidth managed so that user-centric workloads and system-centric workloads could be managed and allocated at the endpoints independent of the network topology. This model permits to have SLA guarantees independent of the topology of the underlying physical network. In case this bandwidth management is still not enough to have system stability,

services could be migrated from the T2 micro-clouds to the T1 micro-cloud. The system in that case would revert to a Carrier Cloud infrastructure where the providers enabled connectivity and services are on the CN cloud.

5.4.3. Service Migration

In case of availability of cross-POP links, as previously mentioned we have new data paths on neighbor POPs that can be used to deliver data and build new out-of-band services. This links can be used as newly available data paths but also for a more advanced usage of cross-POP service migration. In case a micro-cloud being saturated the federation manager could migrate some of the services to a neighbor POP and the clients can exclusively use the cross-POP links.

In this scenario we have multiple possibilities for service migration, we can either migrate the service from the micro-cloud to a neighbor micro-cloud or to the centralized T1 datacenter cloud. This migration process could be implemented on a policy based approach so that it can be possible to be modified on a per deployment basis depending on the system administrator priorities. Further study of such scheduling policies is delegated to future work.

Service migration can serve also as a mean to deal with mobility of devices in terms of geographical sparsity. Frequent movement of clients between different POPs can be accounted by moving the data slowly between POPs, but such model works only if the speed with which the client is moving between POPs is lower than the cost in terms of speed of moving the data between POPs. When clients have POP switching speed of a highly sporadic nature or of high frequency the data can be moved higher in the hierarchy to the T1 cloud so the services migrate but the data is static in T1.

5.5. Federation Manager

The proposed cloud infrastructure leads to multiple control possibilities for the cloud federation, the Federation management infrastructure, the entity coordinating multiple clouds together to produce a usable service. The control system for the federation leads also to design choices concerning the cloud infrastructure. In the Star topology we have assumed for this work, clouds are distributed in two variants; a micro-cloud fabric composed of multiple T2 local clouds on the edges of the network and T1 cloud a datacenter level cloud part of the WAN. By considering the Star topology and the placement of the clouds, control can either be centralized thus having transparent federation or decentralized by having a non transparent federation. Both of the federation models are viable alternatives for the proposed federation architecture. The chosen federation model for this architecture will be centralized and non transparent. They will be discussed separately with cons and pros for each choice and then one of the model will be chosen to be the main federation model. The federation model will be assumed to be centralized and non transparent, more details will be shown when talking about the positive side of such choice in the following subsections.

5.5.1. Federation Layer

In a multi-cloud federation, federation could happen on any layer of the cloud system. Some system may implement federation on the IaaS layer by rendering invisible to the users of the PaaS or IaaS that the federation exists. The IaaS layers of all the clouds would handle resource provisioning between multiple IaaS providers and hide to the upper layers the fact that a federation exists at all. Or by exporting federation specific functions though the IaaS interface, but normally the upper layers PaaS, SaaS and client don't need to necessarily know that the IaaS is actually a multi-cloud federation [66].

Another way a multi-cloud federation could be implemented is by federating at the PaaS layer in which the IaaS-es of different clouds don't have any idea of each others existence. The federation is executed on the upper layer, the PaaS. In such federation scenario it is the platform or the client, in case a platform is not present, who is responsible for the federation mechanisms. It needs to implement metadata and multi-cloud resource provisioning to interact with each of the clouds in the federation. When resources are requested from a client application, the PaaS or the client contacts and interacts with each cloud in order to satisfy this requests. This approach has a low footprint on existing IaaS because of no need of actual modification, but elevates the complexity level on the PaaS side. For each cloud a driver would need to be implemented for accessing the clouds homogeneously and problems may arise as different IaaS providers may use different cloud technologies and may be unable to hide the heterogeneous nature of the multi-cloud.

Other ways to implement federation are by implementing it on the PaaS or SaaS layer, the problem with implementing federation on this layers is that generally different PaaS provide different programming models, run-time environments and tools to manage execution of programs. These languages and programming models are not standardized and differ from each other greatly thus are not compatible. The complexity of achieving such federation is fairly high and without standardization may prove to be infeasible.

5.5.2. Transparent Federation Model

With transparent federation we describe physically separated clouds on the same or different WANs (T2 micro-clouds) in which only one CIM (T1 Cloud CIM) exists for the whole cloud federation. In this approach no separate federation controller is needed as the CIM manages resources in the federation as if it was one big cloud.

The multiple micro-clouds (T2 micro-clouds), composed of at least a server grade machine, are distributed in different LANs. Of these clouds there exists one, which provides also the CIM (T1 Cloud CIM). The other clouds are connected to the CIM enabled cloud, using virtual LAN or tunneling technologies to build a unique LAN overlay and give the impression that all of these clouds exist on the same physical network.

This approach provides an easy implementation of a federation system as no additional changes to the existing CIMs would be needed. However problems arise from such a configuration as there is no actual distinctions between VMs on different clouds, or this distinction needs to be added to the CIM, maybe by separating different clouds in different IP ranges and use latency as metric.

Another drawback of this technique is that the added complexity of having a network overlay deteriorates performance of the network between different clouds but also between the VMs on the same machine as traffic would always need to pass through the VLAN or LAN tunnels. Performance deterioration is due to the overhead of the tunneling technologies as packets need to be encapsulated/decapsulated in order to reach the machines and VMs on different LANs.

Apart from performance concerns this model assumes that the connection between the clouds and the CIM is always persistent, if such connection breaks then the distributed partitioned micro-clouds would be left without any control system and thus rendered for all intent and purposes unusable.

5.5.3. Non-Transparent Federation Model

Non-transparent federation model is a model in which the distributed multi-clouds (T2 micro-clouds) are full cloud deployments with each having its own CIM (T2 micro-clouds) management system, in this case a third party software is needed to perform the federation. The Federation Controller (FC) would run in a centralized fashion on one of the datacenters (T1 Cloud) or a distributed software running on each one of the federated clouds.

The FC would be the entry point to all the resource provisioning system and also would need to care about scheduling, error recovery and multi-cloud monitoring and authentication mechanisms. Also the federation controller would need to have a universal interface to access heterogeneous clouds uniformly.

The network topology of the VMs from such federation would be by using public IPs thus flat networking or a reserved private network in case of private multi-cloud. In this scenario the networking is not penalized in performance as no amount of tunneling or virtual networking is involved. All the machines reach each other through the WANs or LANs by using the flat network IP addresses.

This model also accounts for topology partitioning as each cloud is a self-sustained entity. Each cloud has its own CIM manager so even in case that the main federation controller is offline operation of the cloud can still continue in an unsupervised fashion. When the main federation controller is returned to full functionality then the only data who needs to be updated would be the federation metadata. By having such behavior this model provides network partition stability for the cloud management and the cloud can still be operated locally, even in the absence of the federation controller.

This approach also minimizes traffic needed between the federation manager and the distributed clouds as monitoring and normal operation commands are delivered locally by the local CIM (T2 micro-cloud CIM), only resource provisioning and resource scheduling are done non-locally by iterating with the federation controller. Thus the majority of the external traffic would be functional traffic inherent to the application running on the machines and few federation management and monitoring traffic.

5.5.4. Storage and Identity

Identity management and distributed storage are two aspects crucial to the implementation of any such cloud federation. Depending on the infrastructure in place for such topologies there are different approaches to implementing these aspects.

The area of federated identity manager is still a hot research topic on cloud federation technologies, approaches to achieve such identity federation include modification to single cloud proprietary protocols to include multi-cloud identity management and third party authentication authorities. A work conducted on OpenStack [11] uses both approaches by modifying the local cloud identity management protocols to include third party authentication servers.

As for the distributed storage the model that best fits the topology in our opinion would be local storage for each T2 micro-cloud and Peer-to-Peer deployment of images in local repositories. This solution provides also a reliable system as in presence of partitions micro clouds can continue operations without central supervision.

5.5.5. Final remarks on federation dynamics

The federation model chosen for the multi-clouds infrastructure described in this work is the non-transparent federation with centralized controller. In our opinion this model provides the best stability of operations and also minimizes clouds management overhead on the links that in the Carrier and ISP case is a primary resource.

The Federation Controller is placed in the Core Network T1 datacenter while each one of the micro-dataclouds if treated as a full cloud deployment on at least a server grade machine. The FC provides the main activities for resource scheduling and resource provisioning, and delegates to the local CIMs the management of the lifecycle of the provisioned resources, monitoring and local resource scheduling.

5.6. Real Implementation on Carrier Networks

In this section, a possible deployment for the cloud infrastructure is described in the context of the Carrier Network infrastructure. Fig. 5.4 shows a generalized view of the architecture and the placement of the T1 and T2 cloud enabled hardware. This deployment is one possible way to deploy the clouds, each provider could customize it to fit production and deployment needs.

As shown in Fig. 5.4 the T1 datacenter cloud is an integral part of the Core Network. We can suppose for the sake of 3GPP standard that this data center, connection wise, is placed between the Core Network and the firewall connecting it to the internet. This way Core Network functionality is not compromised and routing can be done easily while maintaining good connectivity.

As for the T2 micro-cloud fabric, it can be either an external cloud enabled hardware plugged into the base station or dedicated cloud enabled hardware inside the base station hardware. We have a separate control network connecting the T2 clouds with the T1 cloud.

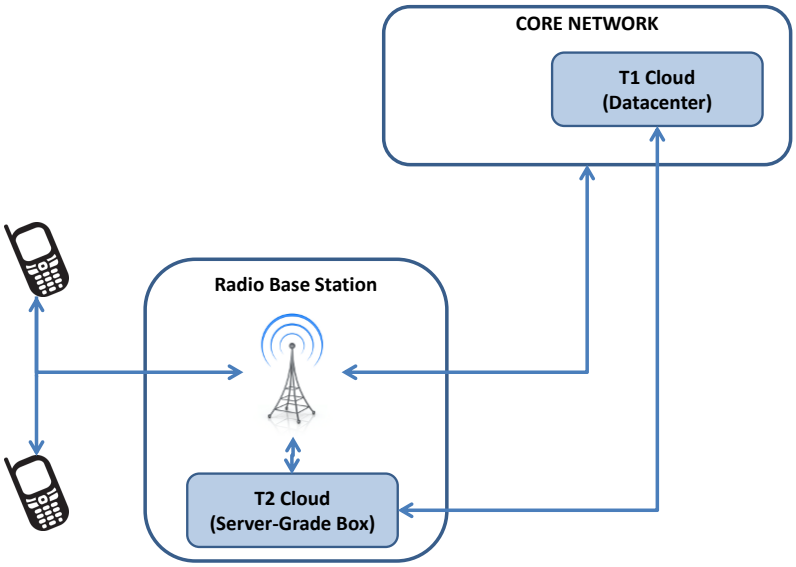

Figure 5.4 – Carrier Network Implementation

The base station is considered to have routing capabilities, forwarding data either to the T2 cloud or to the Core Network.

In the current 3GPP standard it is not possible for the base station to have such routing facilities, but such functionality could be provided through different techniques outside of such standard. The VMs started on both of the T1 and T2 clouds could be part of a private network IP allocation range or be provided with public IPs and provide network isolation through VLAN an Network Overlays.

5.7. Enabled and Enhanched Applications

In this section we discuss some potential novel applications in this context, that would benefit from this cloud infrastructure and cloud provide hints on how to develop this technology further.

5.7.1. Internet of things

Recent developments have seen a lot of attention shifting to the so called "Internet of Things", or the idea of having all of the electronics present in our environment to be connected to the internet [67]. By having a cloud infrastructure close to the clients and thus close to the devices a client uses, these devices could use the micro-clouds in order to achieve better performance or even as a helper for their tasks.

5.7.2. Mobile cloud Computing

As previously stated, Mobile Cloud Computing is to be understood as taking computation and data away from mobile devices into the cloud [29], enabling reduced power consumption and availability of additional resources. Drawbacks of this approach are introduced by the sporadic nature of network latency and that of network partitioning. Both these drawbacks can be addressed by having the T2 micro-cloud close to the user, so that the user can offload computation to the local T2 cloud. In case of user mobility as mentioned, the data of the service could be moved either to the next local T2 cloud or higher up in the hierarchy to the T1 cloud.

5.7.3. Third party applications

Third parties will be able to deploy their own services on top of the Carrier or ISP networks, and the Carrier and ISP would be able to charge for such services as they provide the infrastructure. These applications could be deployed easily through a appstore approach and different business model may be applicable. At the moment the network providers are unable to charge service providers for the network utilization. This approach would be acceptable by both parties as the service provider is assured to have better connectivity to the clients and stability of execution environment, while the network provider is able to charge the service for network and execution environment costs. This model may lead to new streams of income for all the involved providers.

As a real life example, Akamai a well known CDN provider at the time of writing of this work, provides its own boxes to network providers in order to deploy it's cache-ing services. If the proposed architecture on this work would be in place then the network providers may provide such infrastructure and charge for it, while Akamai would have better scalability for their distributed caches and better locality to the users.

5.8. Conclusions and Future Work

To summarize the contributions, this work devises a cloud enabled architecture for Carrier and ISP Networks in which the topology is augmented with cloud infrastructure in order to provide cloud services. The proposed cloud architecture is based on a structured multi-cloud federation, in which micro-clouds are distributed in the PoPs of the network topology (namely T2 micro-clouds) and a central datacenter cloud, namely the T1 cloud. As a control mechanism for such distributed architecture we chose non transparent federation, managed by a centralized cloud federation manager running on the T1 cloud. The T2 clouds provide locality augmentation for services and the T1 cloud provides augmented performance for more performance oriented services.

In order to give a complete view of the architecture, various aspects were discussed including mobility, cloud properties and service migration policies, etc. To conclude the discussion of the cloud architecture, possible applications benefiting such technology are presented. Future work will focus on further study of such cloud federation middle-ware and on possible services running on such a distributed architecture. In details we will

conduct followup work into resource management for such distributed environment in order to optimize not only service latency but a multi-factor optimization of both service latency and deployment/run-time costs. Latency is a very impacting factor in real-time media stream processing applications but lowering operational costs is also of primary concern, thus a trade-off needs to be made.

Having provided both a network bandwidth allocation broker (or bandwidth manager) and a generalized Cloud architecture and related structured model, there is a need to identify and optimized applications that would benefit from such architecture. One type of applications that can be enhanced through the use of these two previous works is live streaming of audio and video. In Chapter. 6 we provide a new self-managed overlay that optimizes live streaming backend paths. The proposed Management Overlay is developed to use the hierarchical structured Cloud Federation (proposed in this chapter) to enhance live streaming applications and lower the impact of scaling number of micro clouds involved in a video stream.

Chapter 6

Enhancing Real-Time Applications by means of Multi-Tier Cloud Federations

6.1. Introduction

In the previous Chapters. 4-5 we develop respectively a network bandwidth manager (managing network bandwidth allocation on the end-points of different Cloud services sharing the same network infrastructure) and a Cloud model based on Structured Cloud Federation for Carrier and ISP infrastructures. We complete such work in this chapter by presenting an application that is enhanced through the use of both these techniques. The selected type of application is live streaming, which is relatively stateless application but with hard SLOs on real-time and no-deal tolerance. The proposed application uses the proposed Cloud architecture in order to exploit the benefits of locality and uses a self-managed federation management overlay that is reconfigures itself with low overhead.

The evolution of Cloud Computing beyond the frontier of a single datacenter is justified as a mean to enhance various system architecture aspects like: cost, geo-locality, energy efficiency and structural properties. These multi-cloud federations represent the ability to orchestrate different cloud providers in order to provide better. Orchestration of different cloud providers in order to ensure better Quality of Service (QoS) is refered to as Multi-Cloud. One such multi-cloud federation model is Community clouds, a federation concept composed of various micro-clouds owned by the community or various management authorities.

One such multi-cloud federation model is Community Clouds, a federation concept in which various micro-clouds are owned by different communities of individuals contributing cloud resources. Community Clouds provide high locality to services towards the user by bringing the Cloud in the edges of the network, and in some cases as an integral part of the access network. Another such type of multi-cloud federation is Structured Cloud Federation for Internet Service Providers and Telephony Providers, composed of geo-distributed micro-clouds placed on the edges of the network and a centralized datacenter cloud on the core network. Services deployed in the micro-clouds have augmented locality, while services on the datacenter cloud enhanced performance. In this work we identify how these approaches to multi-cloud, through the ability to deploy services close to the final user, can be exploited to enhance existing real-time streaming applications. Furthermore a novel multi-cloud overlay algorithm is introduced that provides both latency and backbone traffic optimization for latency critical existing and novel services (concretely focusing on stream computation, live streaming). The advent and advance of Cloud technology led to a growth in service complexity. Commodity computing, through dynamic resource allocation, enables services to modify their structure at runtime in order to comply with QoS agreements between provider and cloud user. Such software governed growth, in service complexity, encompasses and dictates substantial growth in all the layers of the Cloud stack from Infrastructure as a Service (IaaS) to Platform as a Service (PaaS) and Software as a Service (SaaS).

Service complexity and the need to overcome cloud computing limitations (vendor-lockin, performance, geo-locality, resource availability) [62] [68] [69] has contributed to the move of such services into Cloud Federations or Multi-Clouds. In this work we analyze and provide new mechanism for multi-cloud federations, that we call *Structured Cloud Federation* for Telephony and ISP providers and federations of

Community Clouds, that improve existing Live Streaming services through augmented service locality leading to better and more predictable overall application performance. This work focuses particularly on how such distributed architecture can optimize live streaming applications. Such applications provide a hot topic on present access networks, with examples like WhatssApp [70] (live messaging) reaching millions of active connections, and live video streaming with global consumer Internet video traffic rising to 80 percent of all consumer Internet traffic by 2019 [71].

Cloud federations of highly distributed nature like the one considered in this work, provide good candidates to cloud enhancements of existing live streaming application techniques. The contributions of this work are as follows.

1. First, we define **a Multi-Tier Cloud Federation Model** that allows to enhance existing Cloud-based application as well as enables new classes of applications, such as latency-sensitive real-time edge services that need to be located in geographic proximity of mobile devices in order to provide required QoS and improve end-user experience with the services.

2. Second, we propose **a novel algorithm for Cloud Federation Construction and Maintenance** that self-organizes the clouds in a minimum latency configuration, from source to destinations, including also a fallback and high scalability mechanism.

3. Finally, we **evaluate and demonstrate the benefits that can be achieved from deploying real-time applications and services on a multi-tier cloud federation**, by considering an enhanced live streaming application as a use-case.

This chapter is organized as follows. Section 6.2 provides an introduction to the basic concepts needed to understand the various entities of the system model. In Section 6.3, we describe an overlay based cloud federation model that codifies the augmented locality approach. Section 6.4 presents overlay construction and maintenance algorithms. In Section 6.5, we describe enhanced applications, that benefit by the use of this architecture. We conclude and discuss future work in Section 6.6.

6.2. Stream computing applications

The targeted services for optimization, are general stream computing applications, where a stream of data needs to be processed in a distributed fashion. Live streaming applications, like WhatsApp [70] live messaging, live video with message interaction Periscope [72] or live multi-party interactive communications like Skype [73], should provide a service and communication time lower than the human perception time in order to provide a, as close as possible, natural communication medium. The human reaction time can be accounted at the order of 150ms [74], and anything more than that would create highly unnatural interactions. Live Streaming services could be subject to other Service Level Agreement (SLA) related concerns in order to support third party

application orchestration, thus not only human reaction time. One such example could be live feedback from the viewers of the live stream, in other words live interaction of the viewers with the content being streamed.

6.2.1. System model

The system model is composed of access networks providing highly geo-distributed clouds, backbone and public internet clouds. A high level view of such model is shown in Fig. 6.1 with all the related components. Each access network provider enables service deployment in any of the distributed clouds and the backbone cloud. The clouds inside the same provider are interconnected with high performance links and the providers have high performance links to the internet clouds or direct high performance links to a limited set of neighbouring providers. In this work we provide a way to build a self-organized structure between such clouds, in order for the clients to fetch the live stream directly from the local clouds, the closest cloud, minimizing both backbone traffic and stream latency. The introduction of a self-organized structure between the clouds eliminates the possibility of central management becoming a central point of load and failure.

A relevant research question in this case is using the architecture to have a distributed multi-cloud algorithm that produces a good enough approximation of the shortest path between the source of the stream and the receivers, and at the same time minimizing payload duplication along the backbone. In developing such algorithm we discovered other properties that benefit the application model of live streaming. Such properties are client churn decoupling, enhanced support for client mobility and on-the-fly scaling of the resources to meet load demand.

6.2.2. Cloud Federations

Cloud federations as introduced in [75], [76] define a collection of cooperating Clouds, managed by the same or different administrative authorities, in order to provide a set of common services or applications. The nature of interaction between the clouds defines the federation type, thus differentiating between Cross-Cloud, Cloud Federation (providers cooperate in the federation) and Multi-Cloud (provider has no idea of the federation, based on client middleware) [76]. The rest of the work will use the term cloud federation, but no limits are imposed on the federated entities, as such the terms can be used interchangeably. In this work we consider highly distributed cloud federations like those described in [15] and [77]. To generalize, a fabric of highly distributed Micro-Clouds are used in conjunction with datacenter grade Internet Clouds in order to provide distributed services. The key idea of such cloud federations is the ability to have augmented service locality through the micro-cloud fabric (ability to deploy services on the edges of the network) and a stable backbone through private or public cloud to provide performance oriented resources. Such federation provides to the services the possibility to trade-off between performance and locality and in some cases the best of both worlds, when services are composed of locality components and performance components. Thus these cloud federations are needed in order to have a highly scalable and highly distributed architecture, with augmented locality.

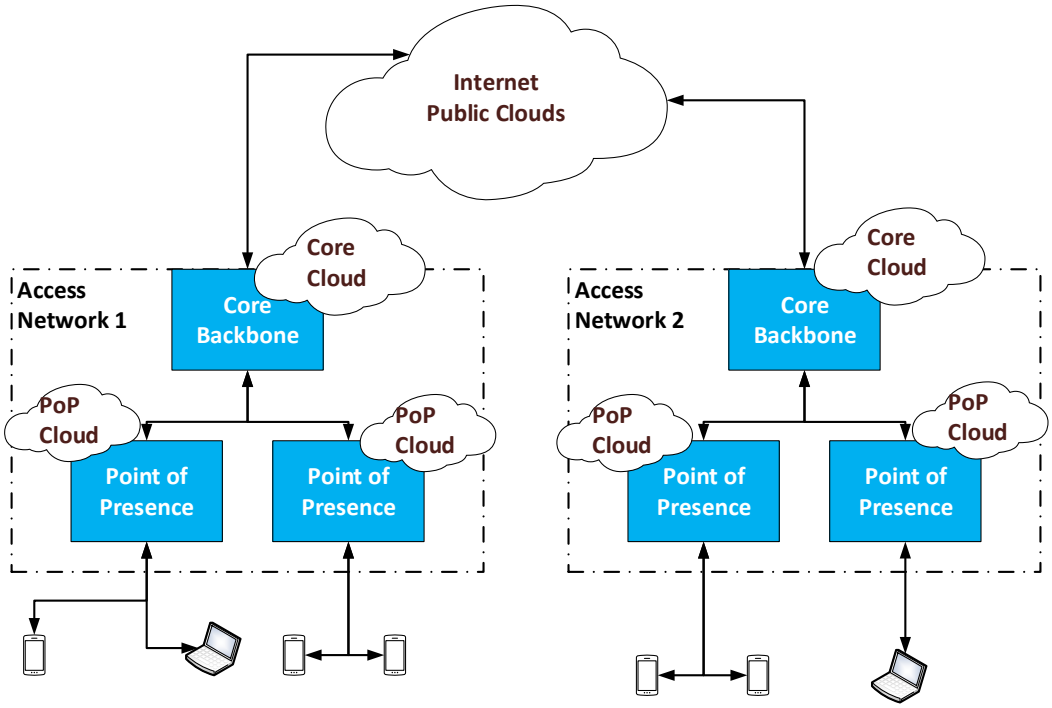

Figure 6.1 – Extended Cloud Infrastructure and Cloud Federation Model

In some cases due to the scale of the distributed environment having a one cloud solution may not even be an option, as the system traffic would greatly influence the client traffic, with which it shares the access network. In such cases self-managed micro-clouds are the only option in order to have a distributed cloud infrastructure.

6.2.3. Stream Computing (Live Streaming)

Crucial use cases are enhancements to real-time applications achieved through the use of highly distributed cloud federations as previously described. The applications that will be discussed, namely Live Video Streaming, are to be treated as a case study for such technology and as a small sample of application types that can be enhanced through these architectures and techniques. Live Streaming refers to streaming of live video channels by means of HTTP communication, where not only the quality of the stream is the issue (network bandwidth) but also the latency of the stream. Real life examples of such case are Web TV or IPTV and live event multicasting where the latency profile of the stream should be restrained by a minimum QoS. Virtual Reality dictates a bi-directional stream of information, as such latency is of utmost importance considering that the cloud is not a mere repeater of the stream but also a manipulator and all should be restrained within the human action perceptible time. Such limitation ensures a flawless user experience and good QoS parameters.

6.2.4. Overlays

The concept of overlay is that of creating a structure on top of an existing physical interconnection between resources. Overlay structures may be constructed in such a way to encapsulate system properties that enhance applications. Such overlays are common in publish/subscribe systems and in P2P systems where peers are organized through a distributed overlay. The presented system design makes use of an overlay construct in order to build a locality enhanced, cloud enabled, environment in order to optimize service latencies. In Section 6.3 a federation overlay is constructed in order to optimize latency of real time applications such as Live video Streaming.

6.3. Federation model

The federation model developed in this section is a self-organized cross-cloud service overlay, that enables enhanced locality cloud-based services. The cloud federation mechanism embeds in the federation structure the concept of locality. A software level overlay is built and maintained between the clouds in order to provide minimized stream latency. In order to give a clear picture of the proposed solution, we define some requirements that the federation should conform to. Following such federation requirements the remaining subsection of this chapter presents an architecture blue print.

6.3.1. Federation and System Model

The system model presented in this section clearly defines the cloud federation nature and system components. A high-level view of the considered federation and system components is shown in Fig. 6.2. As previously described we assume the presence of Micro-Clouds (PoP Clouds) in the access networks of various providers and also a datacenter grade Core Cloud, in such AS backbone. PoP Clouds are physically interconnected to a small number of physically close micro-clouds and also to the Core Clouds with a high performance link backbone. The Core Clouds posses high-performance links to a small number of neighboring Core Clouds, to all the PoP Clouds of the AS and also to the various Internet Clouds.

As for the FM introduced in the following subsection, it is placed in the Public Clouds in order to have ease of visibility to the other components of the proposed federation model. The proposed self-organized algorithm builds a locality aware overlay based on distance from the source PoP Cloud toward all the members of the live streaming multicast. The assumption regarding the source of the stream, is that the source client, uploads the stream to the local PoP Cloud which effectively becomes the source cloud toward which the other clouds try to build a proximity aware overlay.

In this first version of the system design we assume the setup is being used for a single stream, in order to focus more on the overlay construction mechanism and leaving to future work additional concerns such as resource allocation within the clouds and multiple streams. The overlay is build on top of the physical setup shown in Fig. 6.2, and all the resources inside the clouds use the overlay setup for communication. This simplifications

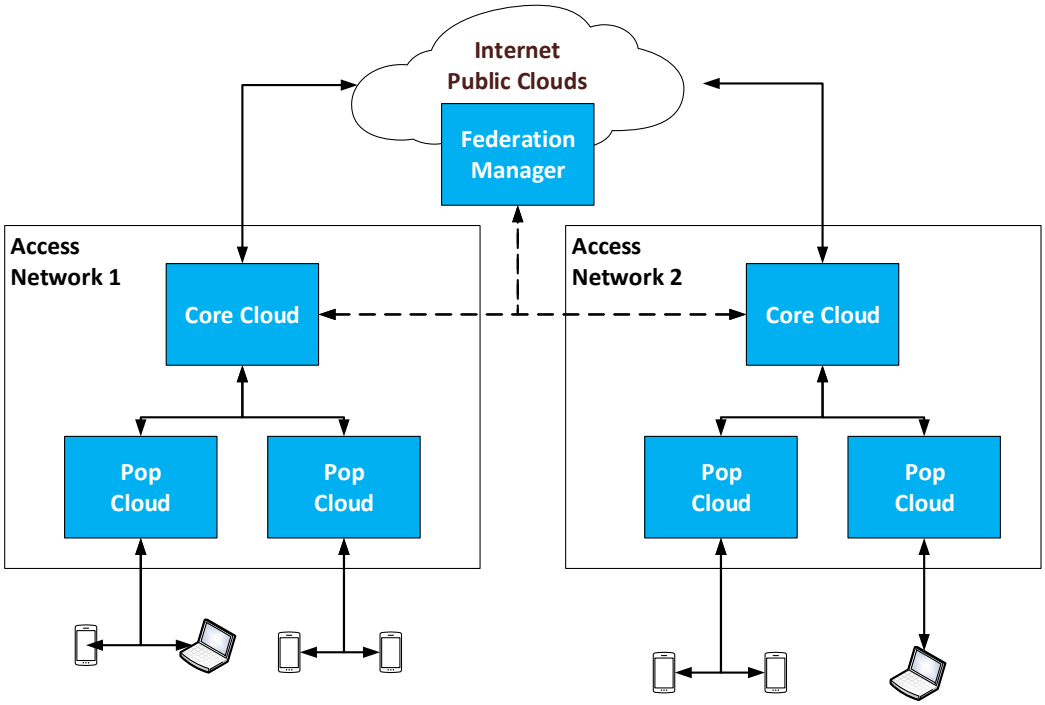

Figure 6.2 – Layered Federation Model

of the system view enable the study of the system as a graph of which the edges are Wheel graphs with the Core Clouds as center and in turn the Core Clouds and the Internet Clouds form a random interconnected graph (Fig. 6.5).

6.3.2. Federation Components

Enhanced locality is a crucial part of the system design as such will form the base of the system design. The architecture is composed of a geo-distributed decentralized cloud infrastructure built of fully functional cloud deployments. In case of Community Clouds, such micro clouds are provided by the community and provide resources on a cooperative basis or on pay-as-you-go plan. In case of ISP and Telephony clouds, the decentralized infrastructure is managed by one provider or collaborating providers as a paid service.

In the design of cross-cloud connectivity we provide a structureless interconnection and leave freedom to the architecture implementer to build specialized inter-connectivity as dictated by the actual service being implemented. The underlying cross-cloud network infrastructure should provide a uniform network access to the micro clouds such as public IP addressing or VPN access for cross-cloud communications. The clouds should be accessible to all other clouds and clients of the service. IP is chosen as the best inter-connectivity protocol as it is the de facto standard for internet applications, and if is fairly supported by any live streaming application.

Hence forth the decentralized cloud infrastructure will be referred to as micro-cloud fabric or micro-clouds. Such denomination does not imply anything about the performance of the clouds. The micro-cloud can be anything between a single machine cloud deployment to a highly geo-distributed datacenter. The name implies only that the resources available are quantitatively less then that of a public or private backbone cloud.

In order to enhance this cloud infrastructure with a more stable backbone and the ability to cross connect clouds in different regions a private or public cloud datacenter is introduced in the design. At worst on geo-distributed resource exhaustion, the system falls back to an Internet model by using the private or public cloud, until new local resources are available. A central entity manages resource allocation and hosts the Federation Manager (FM). Such manager is in charge of monitoring and de/allocating resources in the federation in order to scale the services. The FM manager is going to be used solely for resource de/allocation for the clouds taking part in the federation, while the overlay structure implements the discovery algorithm for the clients of the service. By doing so the FM does not become a bottleneck for the system and client resource discovery, distributing such service to the clouds.

A naming scheme or cloud transaction provider is needed in order for dynamic resource discovery in the cloud federations. The scheme defines also reachability of the resource, as through the naming aliases different providers can take over service deployment on run-time with minimal impact to the existing infrastructure. Such provider could be a P2P gossiping algorithm, where the clouds periodically update resource availability, or on the other hand such service could be provided by a third party resource market place.

In our solution the cloud overlay will use the gossip based protocol for resource discovery or control statements from client toward the service or for cloud-to-cloud service data. As for cloud resource allocation, the centralized FM entity will allocate resources based on maintenance cost. The FM in our case will be the source datacenter cloud or a cloud in the public domain. We choose a gossip based algorithm as it alleviates the load of managing mobility of clients through a central entity and also it provides a good distributed approximation to the optimal shortest path from the source to the clients.

More properties of such tiered cloud federation are shown in Fig. 6.3, as we see the more we distribute the clouds the more we gain in locality but loose in resource availability. The federation overhead is higher in the lower tiers, as higher decentralization implies greater complexity in managing the cloud services.

Having discussed the main system and federation components we move on to the description of the federation dynamics as a factor of service implementation.

6.3.3. Federation Dynamics

In this work we provide a dynamic federation overlay between the clouds in order to have a loosely coupled federation. The federation model mimics the application restraints in order to provide the best available resource locality for live stream computing. Providing a common interface and a tightly coupled federation between the service and the resources, the application enhancements are encapsulated in the structure of the federation and as such management overhead is minimized.

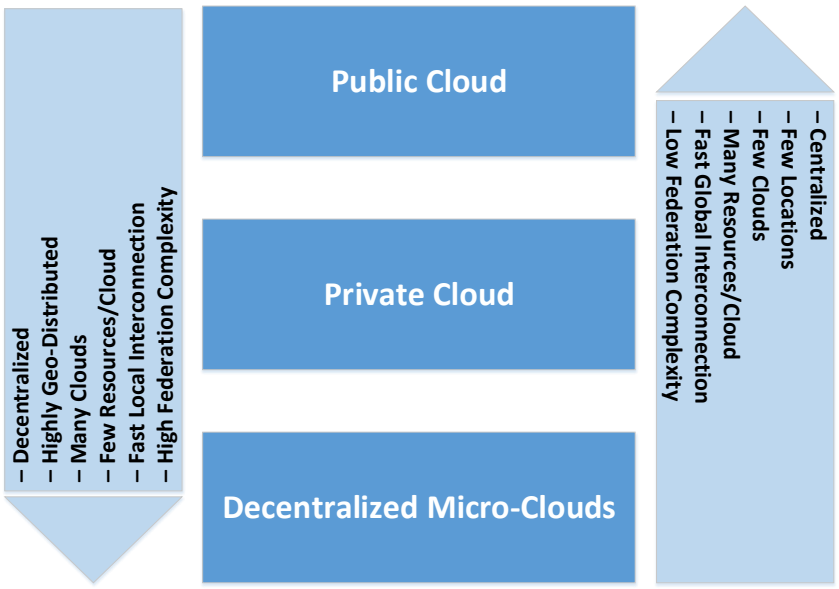

Figure 6.3 – Layered Federation Model Properties

The discovery service provides the clouds with the available resources that can be scheduled and the federation can allocate. Each application then builds its own federation overlay on top of the federated resources in order to optimize the runtime. As such for the live streaming scenario we developed a hybrid federation where the federation is built as an "Gradient" variant overlay.

The system as shown in Fig. 6.3 is composed of three service layers, in a multi-tier federation model. The upper layers of the federation model provide service stability, more centralization, enhanced performance. By moving from the upper layers to the lower layers we make compromises while gaining on important factors. The lower layers of the federation provide enhanced locality for services, and are more geo-distributed and sparse, but provide lower service performance and higher coordination cost. Fig. 6.3 shows more properties and trade-offs of having the service in a specific layer.

A similar architecture of structured federation architecture is introduced in our previous work [77], where services can move between the various layers of the cloud federation in order to trade between cost, stability and locality. We continue building on the concepts introduced in that work in order to optimize real time streaming applications.

6.4. General Architecture

This section provides an architecture of the proposed solution for live streaming by use of a cloud federation overlay. The solution decouples user churn from the architecture, enhancing stream latencies and also decreasing backbone traffic for the different AS-s involved in a stream session.

The cloud architecture is based on a self-regulated "Gradient" structure between the clouds built by using a distance metrics from the source of the stream. In our experiments such distance metric is path length from the source. Fig. 6.4 shows a possible evolution of the system and the organization of the clouds back-end in the gradient configuration.

We chose a Gradient topology as it provides several benefits in the live streaming scenario. As a first benefit, the distance from the source cloud is cumulatively calculated as the algorithm proceeds and at the same time the gradient is consolidated. On every cloud executing the algorithm in rounds, the best neighbors in terms of distance to source and distance from the node to such neighbor clouds are selected. Each round of the algorithm, creates a better distance approximation and every round neighbors are guaranteed never to give a worst distance metric than the previous round.

Another benefit is that the similarity set of the gradient for each node, provides also a fallback mechanism for the cloud backbones, when the closest cloud fails then the stream can be fetched by the second next closest cloud and so on. A third benefit provided by such approach is that mobility of the clients is taken care of locally to all the entities of the system. The management of the locality is achieved by gossiping to neighbors and updating the proximity view metrics fetching the stream from the similarity view.

A Gradient overlay structures the clouds in layers based on the distance that the proximity function calculates from the source of the stream. In our case the proximity metric can be latency or number of hops, as an approximation of the end-to-end latency parameter. In this work the latency is considered as seen by the end-to-end propagation latency from the micro-clouds, clouds, or clients, and the forwarding latency is included in the end-to-end measurement. This is justified by the fact that the inter-connectivity between the micro-clouds and backbone clouds are optimized by construction of the access network, and as such the end-to-end latency is a more adequate measure.

The clouds from different AS-s which are closer to the source receive directly the stream from the source while the other clouds further down the hierarchy use these clouds as forwarders. By doing so we lower packet duplication that would result from having each cloud fetch the content from the source cloud, or across peers in a P2P system without topology information codified in the system design. Effectively implementing a multi-cast protocol across different AS-s.

The cloud federation back-end is constructed in self-organized tiers, while the clients can use such back-end as a multiple source to receive the stream, and as far as mobility goes in the contest of this application, when the user moves the metrics of proximity get updated and the user fetches the stream from the newly discovered close-by cloud. In order to allocate the resources needed between the different clouds, as mentioned previously, a centralized FM is used. Such federation manager will be introduced in the remainder of this section, and is responsible for scaling the application by allocating/releasing resources in a dynamic fashion. This manager is necessary for accounting reasons, but also as a cost allocation entity for the federation. The manager is not concerned with functional concerns of the actual streaming.

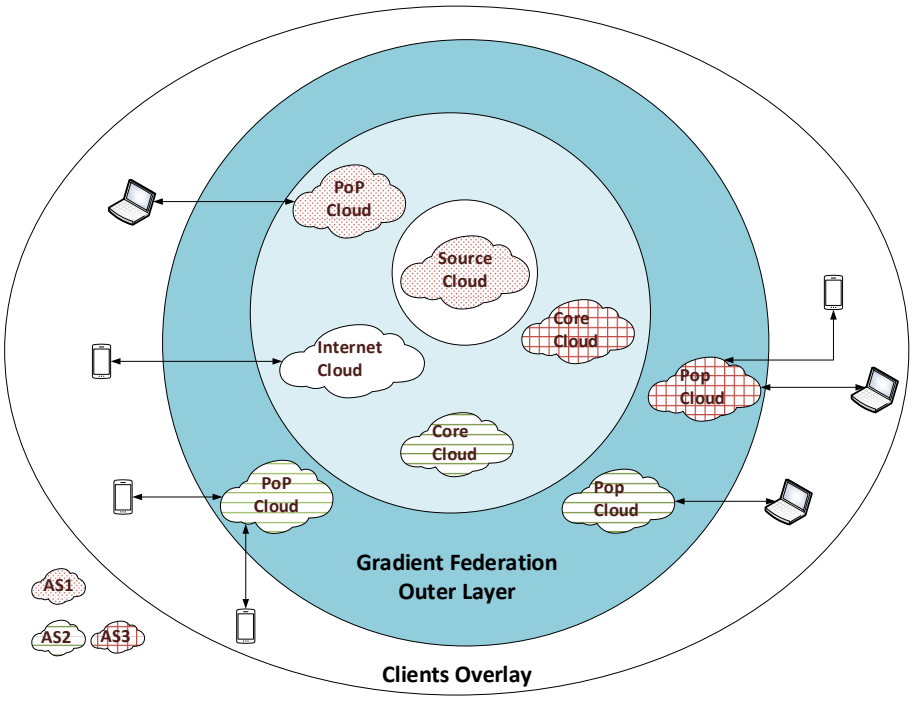

Figure 6.4 – Layered Federation Model Instance

6.4.1. Software Architecture

The key component of the system design is the Federation Manager. Such component de/allocates cross-cloud resources in order to scale the system to meet client demand. Accounting for costs and resources is attributed to the FM. If such task would be distributed, the decision making would over complicate management of the system and tracking of costs. When any of the clouds is exhausting their resources they notify the FM to scale the system either horizontally (scale vm parameters for that particular instance) or vertically (allocate more vm-s in the AS in order to handle the load).

The FM and each cloud in the federation can map hosts to AS-s by means of prefix matching and also by querying the involved AS clouds. Since the federation is built by paying for resources, such information is crucial and should be provided by the clouds on which resources can be allocated. The routing information can be also derived by matching prefixes to a known IP block allocation. More precise info can be given by the specific AS cloud provider with the finality of selling the resources.

Clouds participating in the live streaming overlay are bootstrapped by the FM. When new cloud resources are allocated the FM provides a initial seed list of other clouds part of the system, from which the cloud can start integrating itself in the gradient overlay. The virtual servers reorganize themselves on a proximity based gradient in order to provide a locality aware spanning architecture. A high level view of the system is shown in Fig. 6.4,

and as shown the cloud resources are organized in a gradient. The clients connect to the architecture gradient back-end transparently and update their proximity measure to ensure that they are getting the stream from the closest source.

For scaling purposes the clouds simply monitor the rate of incoming clients and if such rate is equal or higher than the setup speed of new resources, notifies the FM to allocate new resources and starts admission control on resource exhaustion. The clients keep an updated list of servers ordered by cloud proximity to itself. As such when admission control is operating the clients simply try different clouds until a cloud with available resources is found. More advanced and efficient allocation and scheduling policies will be considered in future related work.

6.4.2. Overlay Construction, Maintenance and Resource Scaling

Algorithm. 1 presents the overlay construction and maintenance operation for the cloud federation service oriented architecture. Each cloud runs a cloud virtual infrastructure manager (VIM) which in turns execute the following algorithm to maintain a local view of the neighboring clouds closest to the source cloud and to itself. Constantly the clouds exchange their views in order to derive a stable gradient topology, where the center of the gradient is the source and the farthest from the source a cloud is, the furthest it is placed in the gradient.

The Virtual Cloud Managers of each cloud periodically executes Algorithm. 1, a gossip based algorithm, in order to build the cross-cloud communication network for the live streaming model described in this work. The VIM at every moment in time maintains a local view of the whole system that is based on the clouds that enhance the locality measure compared to the distance from the source that the current cloud has. Based on the "Gradient" gossip algorithm, the view consist of a set of clouds (similarity set) that have a better utility function then the actual cloud running the algorithm and the criteria for new clouds to join the similarity set is as follows: the candidate cloud to enter the similarity set, needs to improve or at least not deteriorate the utility function over all the other clouds in the set. This is translated into the fact of having better triangular distance between the actual cloud and the source (not worst proximity to either the source and the actual cloud running the algorithm).

Line 1-3 respectively pick a random cloud VIM partner to exchange views with, sends the local view to the partner with a request to exchange views, and create a copy of the view to process during the algorithm. Proceeding on lines 4-6 the cloud receives the best representatives of the view of the partner, and measure the distance between itself and the clouds in such view. The following lines 7-10 check if any of the new clouds takes us closer to the source and updates the similarity set, the view of the system. Finally terminating the algorithm in lines 11-13 the new view of the neighboring clouds is sorted and trimmed to match the k parameter dictated by the system, and updates the proximity measure for the cloud executing the algorithm toward the source by using the best candidate, the first element of the sorted view. The order of the elements in the similarity set is generated by calculating for each element a tuple composed of: Triangular distance to source through this neighbor, distance to this neighbor and last the cloud ID. This order shows the preferences

Algorithm 1 Overlay Construction

Require: L^k View ordered by tuple (proximity to source cloud, proximity to cloud executing algorithm, cloud ID)
Require: k view length
Require: tr_window transmission window length
Ensure: D^k New view of neighbors
Ensure: $dist\_to\_src$ Cloud distance to source cloud
Initialisation :
$round\_partner$ = $random\_sample(L^k, 1)$
S = $push\_to\_query(round\_partner,\ L_{1-tr_window})$
D = L
Measure neighbor sample distances
for $i = 1$ to k **do**
$S_i.distance\_to$ = $measure\_distance(i)$
end for

Update View
for $cloud$ in S **do**
if $closer\_to\_source(cloud, D)$
$D.append(S)$
end for
#Sort View by Tuple(proximity to source cloud, proximity to cloud executing algorithm, cloud ID)
D=$trim(sort(D), k)$
$dist\_to\_src$ = $measure\_distance(D_0) + D_0.dist\_to\_src$ **return** D, $dist\_to\_src$

the algorithm implements toward the forwarding cloud.

The initial view of the system is obtained by the clouds running the algorithm directly by the central FM, which acts as a bootstrap server. In the basic implementation of the system, the FM allocates resources in a greedy fashion in order to satisfy all clients with local connection. A problem of the P2P live streaming gossip based algorithms is the inability to limit or foresee the depth of the spanning tree or of the formed graph in general and as such no strict guarantees can be made on latency. The following subsection introduces some observations to show that for this system model, such limits do exist.

The resources of a multi-cloud federation have predictable availability based on SLA-s contracted with the cloud providers. In case of cloud failure the algorithm can be modified to simply remove such cloud from the proximity list of neighbors and run with the reduced neighbor view.

6.4.3. Limited spanning depth

The key idea in order to enhance latency in the gradient overlay is not only the "Greedy" approach used on building the overlay by means of locality, the other idea is a limited height

of the overlay levels. In order to prove that the latency growth is limited we present the following observations. By construction Access Networks interchange data through special point of presents (different from the clients points of presence described in this work), that provide high bandwidth low-latency cross-AS connectivity. This network topology enables simplified network control, accounting and management. Thus the shortest and fastest path between two hosts in different AS-s is to pass through the Core Networks. If the edges of the two AS-s could be able to exchange data between client PoP-s the data path would be significantly slower and bandwidth limited.This reasoning leads to the observation that Core Network Clouds, by construction, can only be interconnected directly through each other or through a Public Cloud.

A second observation is based on the fact that at worst case, overlay latency and depth spread, is bounded by the largest pair-wise distance between Core Clouds. Such pair wise distance between core clouds at most can be of the same dimension of the Internet Clouds and Core Clouds. The worst case is shown when such interconnecting clouds are arranges in a chain, thus the two edges of the chain are $O(n)$ hops away from each-other. We make this observation by considering the possibilities of the paths connecting two random hosts in the overlay. Considering any two hosts H_a and H_b and a possible path connecting the two in the overlay $p_{(a,b)}$. Such path is of the form $p_{(a,x)}...p_{(x_n,b)}$. Let us consider the types of possible intermediary paths, elements H_{x_i} and $H_{x_{i+1}}$ can be of types Core-to-Core, PoP-to-PoP or PoP-to-Cloud. As per our first observation in this section, on crossing towards a core network intermediary, by construction of the system, we are guaranteed that the chosen path can be a candidate best proximity based possible path. Cross-AS crossings of the type Core-to-Core, and PoP-to-Core, comply with system design and could be accepted as part of a best path between two different AS hosts. In case of PoP-to-PoP intermediary links, let us suppose by absurd that the shortest path constructed between two AS-es is eventually only made of PoP-to-PoP links. By means of this assumption we have a best proximity wise path between Cross-AS hosts that is made of only PoP-to-PoP links. This implies that there exists a path between two AS not crossing their Core Clouds, that is the best possible path between such hosts, which contradicts our assumptions from the structural properties of the system, and proves that a chain of forwarders based on proximity can't grow more than the best alternative path that crosses a Core Cloud link. As such in a worst case scenario, the maximum pairwise distance of Core Clouds is the maximum possible chain length of inter-PoP chaining. But the number of Core Clouds and Internet public Clouds is far smaller than the number of PoP clouds as such we can expect a clear limit on the spread of the inter-PoP links.

Having discussed in details the overlay algorithm and the implications of such system design we move on to the evaluation of a simulation of the proposed design.

6.5. Case study of enhanced streaming algorithm

6.5.1. Enhanced live streaming algorithm evaluation

In this section we provide some results based on simulation to support the usage of such novel cloud technology. The provided results validate the system construction as

described in this work, by constructing a solid cross-cloud overlay that minimizes latency and has limited overlay diameter or depth, as such providing enhanced service locality for live streaming. For these experiments we implemented a simulation of the cross-cloud

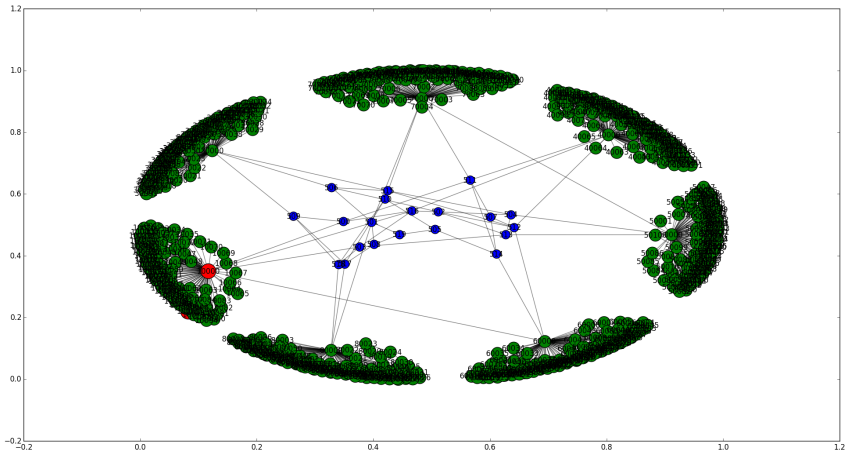

Figure 6.5 – Simulated multi-cloud infrastructure

distributed algorithm. In order to have more meaningful results the proximity measure used in this evaluation is number of hops. Such metric can be easily translated to real latency estimation based on latency measurements on these specific networks (Community Networks and ISP and Telephony access network) introduced in Section. 6.2.

A generalized view of the real access network infrastructure and used in these simulations is shown in Fig. 6.5. Based on the described model, we have a number of geo-distributed clouds that are grouped and interconnected in a "Wheel" graph with the respective Core Clouds as the center of the graph. Internet clouds are simulated by a connected random graph. The Core Clouds are then randomly connected in between themselves and the Internet Clouds graph. The composed graph as shown and previously discussed in the system model has at the core the Internet Clouds and Core Clouds and at the edges the PoP Clouds. In Fig. 6.5 the clusters represent the access networks while the random graph in the center represent the Internet Clouds. This simulation model provides a good approximation of the system model introduced in this work.

The first experiment is conducted by fixing the random graph representing the Internet Clouds and uniformly increasing the number of local micro-clouds. As we can see from Figure. 6.6 the growth of the micro-cloud fabric does not influence much both the conversion rate and the average distance to the source, further more the error between the overlay estimation and the optimal proximity based paths is minimal.

In this second experiment the dimension of the micro-cloud fabric is kept unchanged and the dimension of the public cloud back-end is increased. With this experiment we study the impact of the overhead of such growth in the overlay algorithm. As predicted the growth

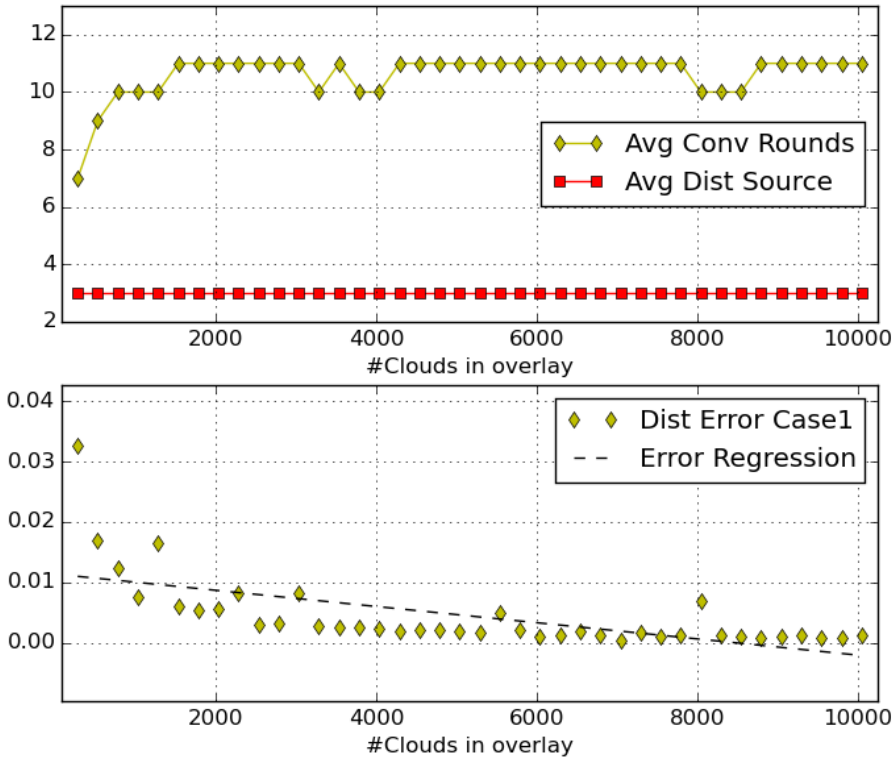

Figure 6.6 – Case1: Convergence for Scaling Micro-Clouds

of the Internet back-end impacts lightly the convergence rate and the distance to the source. In Fig.6.7 we can see that an addition of 400 public clouds impacts lightly the conversion rate and the distance to source. Nevertheless this number would be far smaller in reality as the global cloud providers are in a far smaller numbers. In general by having a small number of public providers and a large number of local clouds as in the first case Fig.6.6 the system can scale without performance penalty. This in itself provides also a good property of the system, as local growth provides better performance and very little management or performance penalty. In this second case as well the errors are really negligible and for this architecture the distributed algorithm produces a pretty good approximation of the optimal proximity aware path.

6.6. Conclusion and future work

To summarize in this work we presented a multi-cloud self-organized algorithm to optimize stream latency for live streaming on massive scale. The proposed algorithm

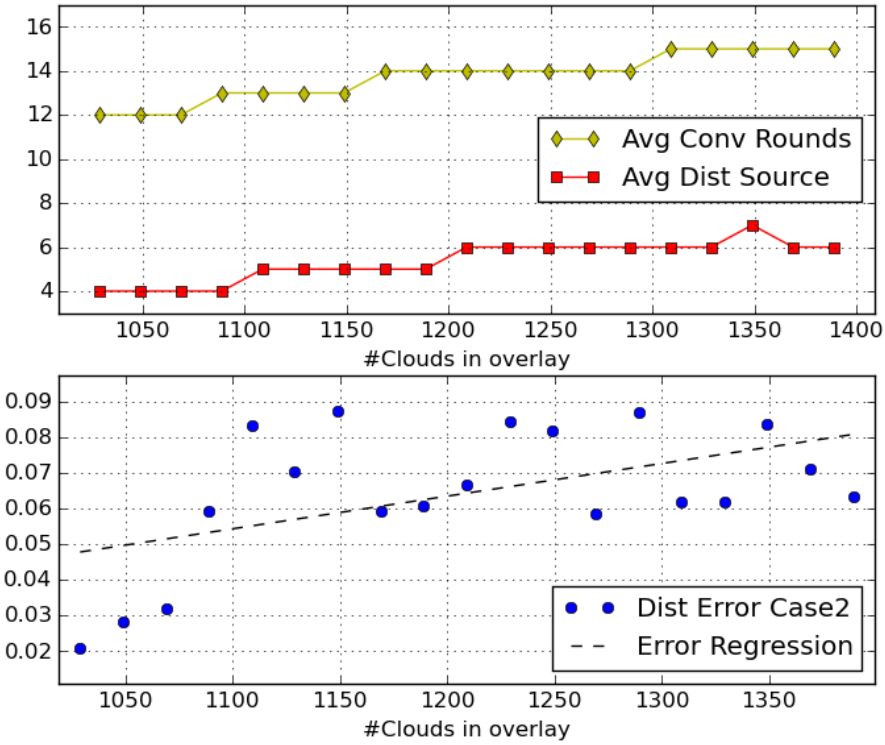

Figure 6.7 – Case2: Convergence for Scaling Public Cloud Fabric

builds an overlay that approximates the optimal proximity based solution up to 0.02 of standard error.

The clients actively request the stream from the closest cloud transparently and are insured persistent presence of the stream either through local or internet clouds. This architecture enables better support for client mobility and scales both vertically and horizontally providing a decoupling of the streaming backend from the client churn and providing a more stable backend for the service.

Future work will be focused on open problems toward the finalization of the proposed architecture, such as resource allocation in the clouds involved in the system, resource provisioning and price vs performance trade-offs toward a complete scalable solution.

Chapter 7

Media Quality-Centric Stream Allocation and Related Patterns for a WebRTC Cloud Architecture

7.1. Introduction

Previous work conducted in the scope of this thesis produced a network bandwidth allocation broker (arbitrating network bandwidth between endpoints of services sharing the same Network [Chapter.4]), a novel Cloud architecture that permits trade-offs between locality and service performance (Chapter.5) and a massively distributed self-managed gradient overlay for live video streaming (Chapter.6). Towards completition of these works we present in this chapter a real user-case scenario, of using real passively monitored user data are used to empirically analyze video bit rate and Cloud Forwarder. By monitoring the performance of video media streams we not only show that video simulcast always outperforms single layer video encoding, but also that minimum-based allocation strategy for streams into the Cloud Forwarder outperforms the other techniques given that no stream can be reallocated after the initial allocation.

Web Real-Time Communication (**WebRTC**) as the leading standard for real-time communications in the web, is seeing a rapid rise in adoption footprint. This standard provides an audio/video platform-agnostic communications framework for the Web, build-in right in the browser, and makes it a very interesting research scenario. The complex technology stack of a full implementation of the standard is vast and includes elements of various computational disciplines like: content delivery, audio/video processing, media transport and quality of experience control, for both P2P and Cloud relayed communications.

To the best of our knowledge, a study of the joint factors of Multi-Cloud distribution, media bit rate, as well as Cloud relayed resource loads is necessary. The contribution of this work is the analysis and exploitation of periodic patterns on *server workloads* and *media bit rate* derived from real test traffic and Cloud-based media back-end measurements, over an extended period of time and at scale. Additionally, a simple and effective load balancing scheme is discussed to fairly distribute big sessions over multiple servers by exploiting the discovered load patterns. A Cloud simulation environment was built to compare the performance of the proposed algorithm with other load allocation policies. Such work is crucial in designing resource allocation algorithms and media Service Level Objectives (**SLO**) spanning multiple Cloud entities. Based on our analysis, we discover strong periodical load patterns even though the nature of user interaction with the system is mostly not predetermined, with a variable rate of user churn. WebRTC[23, 10] is the html5 extension for real-time communications, enabling live media communications between two or more parties using standardised web technologies. WebRTC/RTCWEB is currently specified through three main aspects:

- WebRTC W3C standard API specification for use in web browsers [23].
- RTCWEB IETF standard recommendation for the set of protocols necessary for media communications for every connection [10].
- Webrtc reference software media stack (open source component of Chrome browser), implementing previous specifications [7].

WebRTC/RTCWEB are a set of standard recommendations conceived for delay non-tolerant applications where *interactive* real-time communication is necessary. One application of WebRTC/RTCWEB is multiparty audio/video conferences. A conference is a session where each participant, publishes his audio/video sources while simultaneously receiving audio and video streams from other participants. The API nature of WebRTC in web browsers makes it possible to easily go beyond basic conferencing use cases and allow applications to blend with media communications in ways that had not been possible before.

WebRTC clients are general purpose Web Browsers or devices that implement WebRTC/RTCWEB compatible standards. Common nomenclature for both is WebRTC Endpoint[1] (hence, both referred as such in the remainder of this work). To coordinate among them and/or with the cloud, WebRTC/RTCWEB endpoits require a messaging infrastructure as well. WebRTC/RTCWEB, though, specifically leaves messaging out of its definition, allowing freedom of choice, and focuses on the range of communication protocols and technology stacks that take care of real time communications for media (audio and video) and data signaling. WebRTC/RTCWEB stack intended for both: P2P and cloud-relayed communications. This work focuses on real-time media transmission leveraging WebRTC/RTCWEB and cloud-relayed architectures. An analysis of quality of such media architecture operation is of utmost importance for user experience and the overall performance of the system.

The protocol in charge to deliver media is the Real-Time Transport Protocol (**RTP**) and uses Real-Time Transport Control Protocol (**RTCP**) for quality control. RTP/RTCP[78] is a general purpose transport protocol that provides support for multi-homing. It is standardized to run over both lower level UDP and TCP protocols (although UDP is usually the rule for timeliness performance). RTP/RTCP, among other, adds support for media source identification, media mixers, media track synchronization facilities, quality of service feedback or media bundling and multiplexing. RTP is agnostic to specific codecs and can function as a transport for both video and audio stream formats. For example, in the framework of WebRTC we can encounter VP8, H.264 and VP9 video codecs, while for audio OPUS, ISAC, G.722 or G.711 are common as well.

Live Audio/Video Conferencing in WebRTC/RTCWEB is implemented to use RTP to deliver media to endpoints and servers. In middlebox/server based topologies[79], each endpoint publishes one or more RTP streams for each media stream, and subscribes to each of the RTP streams of the other participants in the session. Other typical mechanisms are also implemented as well by means of a backend, like STUN and TURN for Nat-Trasversal.

WebRTC/RTCWEB is supported natively by major web browsers (e.g. Chrome, Firefox and Edge) and provides a free real-time communication medium. A pure P2P implementation of the standard has many limitations. Some of these limitations are: i) The upload bandwidth (and CPU usage) a client needs when sending the stream grows proportionally with the number of clients receiving such stream. ii) users behind firewalls or NATs may be subject to severe network restrictions and as such P2P direct communication may be not possible. iii) the rate of connectivity failure grows with the

[1] Standardized in: https://tools.ietf.org/html/draft-ietf-rtcweb-overview-12

number of clients joining the session. iv) Additional operations like archiving (saving to a permanent storage) a session may be a challenge. These and other issues constitute a big problem for applications that need reliability, quality and cost effectiveness in common fixed or mobile networks with higher download and far lower upload bandwidths. Common middlebox/server topologies include using Multipoint Control Units (MCU) or

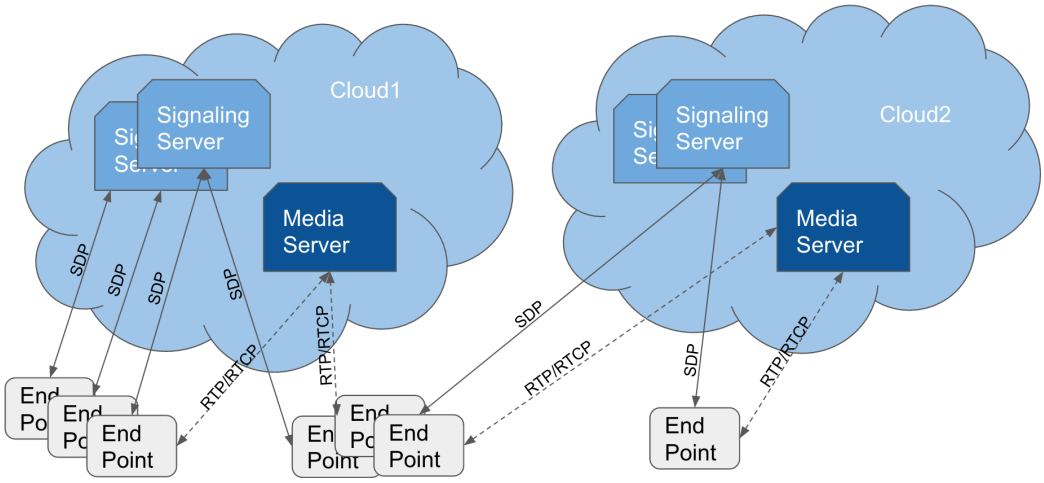

Figure 7.1 – System Overview

Selective Forwarding Unit (SFU). MCUs typically implement both software assisted multicast as well as media translation as needed, while SFUs selectively forward to each participant media (and control) packets in more or less sophisticate ways without transcoding operations. One of the most evolved forms of selective forwarding is the capacity to adapt media quality individually for every endpoint without conducting media translation when scalable/simulcast media encodings are used [80, 81]. In such a case, a sending endpoint (publisher) produce media streams composed of multiple qualities that can then be intelligently selected and forwarded by SFUs for each receiving endpoint (subscriber). Fig.7.1 shows the high level design of such an architecture.

In this work we will focus on examining media parameters and load profiles in the scenario where media operations use a Selective Forwarding Unit (SFU) as media relay (Fig. 7.1).

Network Quality of Service (QoS) is of utmost importance for these communication architectures. Also, one of the most important properties directly related to video quality is video bit-rate. For resilient real-time communications, bit-rate needs to adapt at every moment to the available resources in clients and network, and avoid dropping the communication. End-to-end rate-control in combination with RTP/RTCP takes care of it. In this work, we will focus on studying the impact of machine load in terms of streams/server towards rate-control and bit-rate received at the clients, as well as on the impact of rate-control with single layer encoded video if compared to simulcast[81].

In order to satisfy resource needs, enough SFU units need to be provisioned for all sessions in a communications cloud platform. This work assumes the definition of server

load to be the number of streams managed by each SFU, and since the servers are cloud machines within defined categories we can benchmark maximum allowed streams per server for each category. This metric appears to be more reliable than other resource metrics and can be easily sampled for each server. The allocation of streams directly impacts load factors as well as it can translate into media quality (such as bit-rate).

The motivation of this work is thus the study and definition of stream load patterns (per SFU), bit-rate and other system parameters. With these, one can devise automatic algorithms to predict resource needs and enforce Service Level Objective (SLO) limits. This can lead to improved user experience and service cost management. The observed patterns concern both: periodic repetitive server loads and the influence they have on media parameters (e.g. bitrate). Additionally, we present a comparative study between the use of simulcast (senders producing multiple qualities) and single stream adaptive approach where senders adapt bitrate to the worst bandwidth available to a receiver, all using Google's Congestion Control [43][42]).

7.2. Load Characterization

Load characterization is an impacting factor in devising resource allocation strategies for a distributed service. The scenario we are investigating is composed of a distributed software media multicast, real-time, delay non-tolerant in a subscribers/publishers cloud delivery system. The SFU is the multicast backend while the subscribers and publishers are the consumer and producer clients. Table. 7.1 shows the distribution of the load measurements taken on our test cloud as number of streams per server over $2min$ intervals. Publisher Streams have a mean of 37.31 streams/server while having a $25\% - 75\%$ percentile range of $32 - 51$ streams/server. The number of Subscriber streams, on the other hand, are centered at 85.52 streams/server and have a $25\% - 75\%$ percentile range of $24 - 125$. The number of streams/server without discriminating subscribers and publishers is centered around 122.83 and has a $25\% - 75\%$ percentile range range of $44 - 177$.

Table 7.1 – Data Distribution Load Test Cloud 2Min Interval

Property	Publisher Streams	Subscriber Streams	Streams
Count	190279	190279	190279
Mean	37.31	85.52	122.83
25%	32	24	44
75%	51	125	177
Max	159	868	980

A very interesting trait of such distribution is that the maximum number of streams reached per machine is orders of magnitude higher than the respective mean or 75% range. This aspect is very important as once the first stream is allocated to a server, the following streams associated to the same session need to be allocated on the same server. This study is built with the assumption that a session does not span multiple servers in order to gain

simplicity, while different sessions can be allocated in different server or clouds. In turn, big sessions are not allowed to be migrated to different servers, which makes resource scheduling a critical task for a cloud back-end. If multiple big sessions would be allocated on the same machine, that could cause such sessions to hit the machine stream capacity limit. Once such limit is reached, it would cause problems for streams joining the session as the SFU would not be able to handle the load.

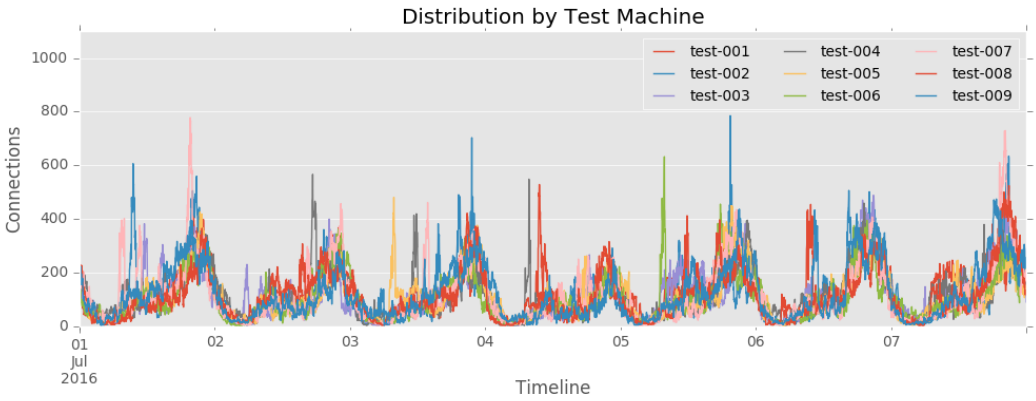

Figure 7.2 – Data Center/Server Load Distribution 7days period

We visualize in Fig. 7.2 the average count of streams/server as measured for one week worth of data (a subset of the dataset used for this resource allocation part). Visualization of the entire dataset would be tedious for longer periods. The load presented in Fig. 7.2 is sampled in 2min intervals as measured from server logs over our test data-center. In general we make the observation from the data that the there is a strong periodic load pattern. Further exploring such pattern, we examine the load distribution in the form of a lag plot (Fig. 7.3) where each lag unit represents a duration of $2min$.

The lag plot exposes a strongly auto-correlated sequence. Clustering of values around the diagonal represents a strong positive auto-correlation. Data-center centric load as such can be well approximated by an auto-regressive (or running-averages) model. The linear regression equation covering the lag observations is written in Eq. 7.1 with parameters $slope = 0.9974$ and $intercept = 2.8282$. Even though we are able to predict the total load going to a defined data-center, a resource allocation algorithm can still get into problems under the restrictions that once the first stream of a session is assigned to one server all other streams of that session will be assigned to the same server. As such, extra care needs to be taken in allocating sessions to servers, so that the load is spread fairly. This is also the purpose of Alg. 2 which tries to balance load on servers by allocating new streams of new sessions to underutilized servers.

$$Y_{t+1} = 0.9974 * Y_t + 2.8282 \tag{7.1}$$

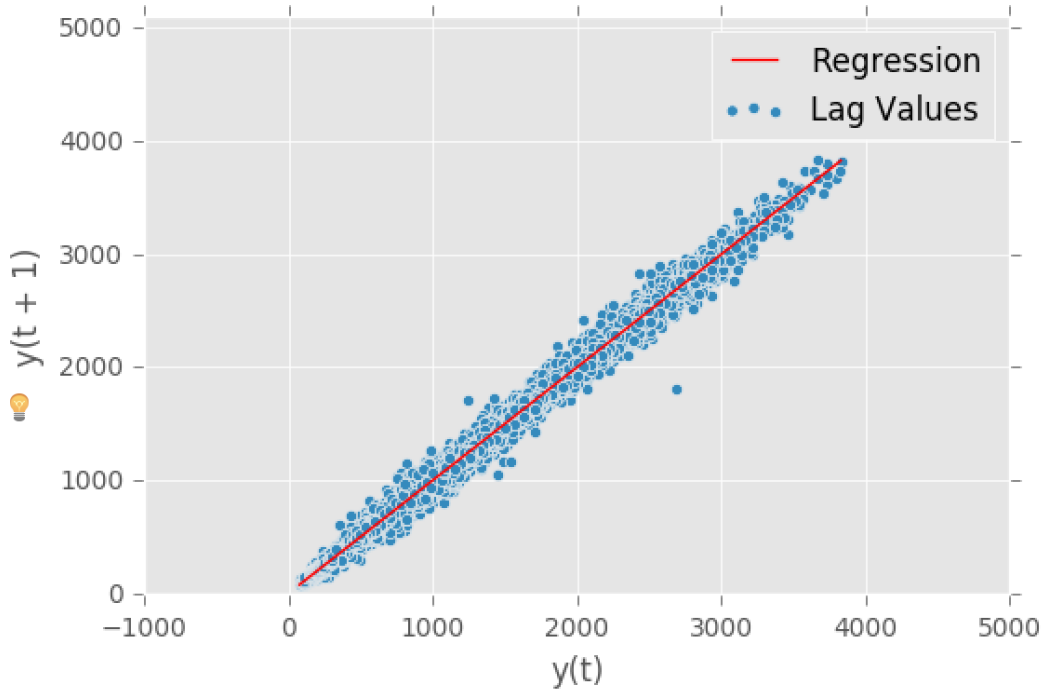

Figure 7.3 – Data Center total load lag plot 1month period

7.3. Load Balancing Algorithm

In this section we present a stream allocation algorithm that distributes as much as possible peak sessions without previous knowledge of session sizes. In a Platform as a Service deployment architecture, knowledge of individual users connecting to the platform is restricted and user anonymity must be guaranteed. Such restrictions in information handling introduce limitations in the stream allocation algorithms. In practice each stream gets a unique identification string chosen at random on creation, and as such no historical data can be aggregated for a defined end user of the platform. Predicting session size may be difficult given that minimal individual user information is available. The allocation policy for each stream should try to offload peaks to different servers as much as possible with only server load-centric historical information.

The server selection policy, part of the Minimal Load selection algorithms, presented in Alg. 2 allocates incoming streams to the machines with current minimal load in a determined time window. Load statistics normally lay within a time window that can be deduced from the data in order to take into account subsequent big sessions arrival time. Lines $2 - 4$ initialize the state of the selection algorithm, then in the following lines $(5 - 12)$ a subset of the currently allocated servers with minimal load are selected. The minimal set entries do not differ more than ***radius*** stream utilization from the absolute minimal server load in the system. To conclude, lines $13 - 17$ select either one of the target subset servers at random with min load cardinality, or allocates a new server to

Algorithm 2 Min Load Server Allocation Policy

```
1:  procedure SERVER_SELECTION(S_Loads, radius, max)
    // S_Loads[i], array of structures [(srv_id, load)...]
    // radius, Minimum set selection radius
    // max, Maximum allowed streams per server

2:      S_Loads ← sort_ascending(S_Loads)
3:      S_MinSet ← [ ]
4:      srv_selected ← Nil
5:      for (srv_id, load) in S_Loads do
6:          if load < max &&
7:         ((load − S_Loads[0].load) < radius) then
8:              S_MinSet.add(S_Loads[i])
9:          else
10:             break;
11:         end if
12:     end for
13:     if len(S_MinSet) == 0 then
14:         srv_selected ← allocate_new_server()
15:     else
16:         index ← rand_int(0, len(S_MinSet) − 1)
17:         srv_selected ← S_MinSet[index].srv_id
18:     end if
19:     return srv_selected
20: end procedure
```

expand the system. We will observe in Sec. 7.7 that the family of Minimal Server load algorithms performs better in terms of maximum server load during operations, through a series of simulation based on real usage trace data from our test data center.

Fig. 7.4 shows the distribution of the peak sessions for the servers of the test data center over a period of 2 days, we can observe that by applying the min load policy, we manage to distribute the big sessions over the different servers and thus protect the system from allocating consecutive big sessions on the same machines. This property is seen as a result of big sessions being rare and their inter-arrival time being sporadic.

The algorithm presented is a simple but efficient way to handle and distribute the load between servers in a fair way. Alas other parameters need to be taken into account before this algorithm is production ready. The decision to either start new servers when existing resources are not available could be changed to a mixed solution between admission control and resource allocation to keep resource cost at bay. Another limiting aspect of the algorithm is that if the rate of incoming subsequent big sessions is lower that the rate at which users join the session, multiple big sessions would end up on the same server. Those sessions on the same server would exhaust the resources available on the server and provide quality issues. The randomization part of Alg.2 (lines $16 - 17$) tries to tackle

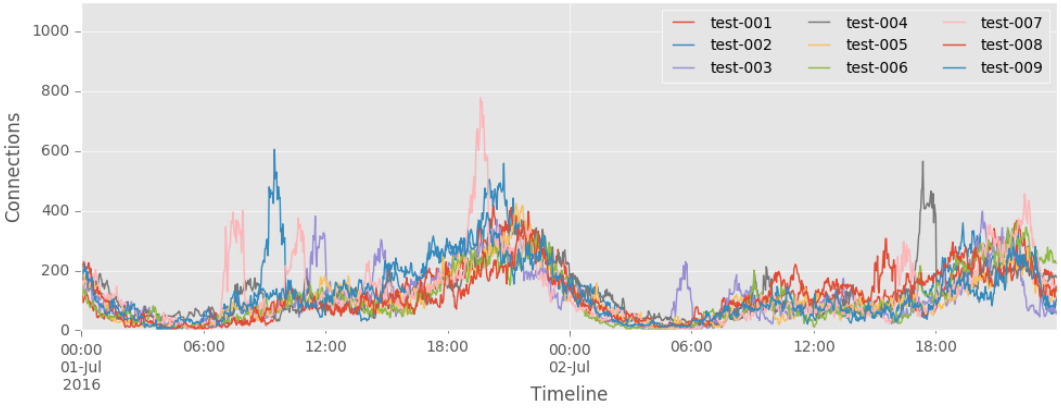

Figure 7.4 – Max Loads in test servers in 2min Interval

exactly this problem by allocating streams on different servers randomly within the subset of servers with minimal load. In practice, one can assume that the rate of incoming big sessions is far lower than the rate at which users join existing sessions. Under these assumptions, the algorithm even though simple, performs quite well. Later in this work (Sec. 7.7) we will further examine other allocation algorithms and compare their performance with Alg.2 through a simulated environment based on real session traces.

7.4. Media Bit Rate Analysis

7.4.1. Bit Rate Distribution

Given a media codec, bit rate is a key quality metric as it has a direct relation to resolution, quality and frequency of video frames being encoded at the endpoints. As such, we separate our analysis into the following video resolutions QVGA(320x240), VGA(640x480) and 720p(1280x720), all at 30 $frames/second$. The codec used for the analysis is VP8 (MtI[2] in WebRTC/RTCWEB endpoints). From previous measurements and also domain knowledge, we can consider that for each of these resolutions, and a wanted frame rate of 30 $frames/second$, a good enough average video bit rate for single layer streaming (only one video resolution encoded at the source), in a conferencing use case environment, can be, respectively: QVGA (300kbs), VGA (500kbs), 720p (1.2Mbps).

The first thing we examine is the distribution of bit rate values for both publishers and subscribers. As shown in Fig. 7.5 **VGA** and **QVGA** comply with the expected behaviour and exhibit normal distributions centered around $317.9Kbps$ and $492Kbps$ respectively. These two distribution are well centered and have an acceptable standard deviation of 90.7 and 130.8 respectively. In the case of the **720p** resolution, a different behaviour is exhibited in which the bit rate profile is distributed over a longer range of values and only a portion of the values is within the upper acceptable range of bit rates. Based on the data gathered, we conclude that such behaviour is mostly due to still commonly limited upload bandwidths

[2]Mandatory to Implement

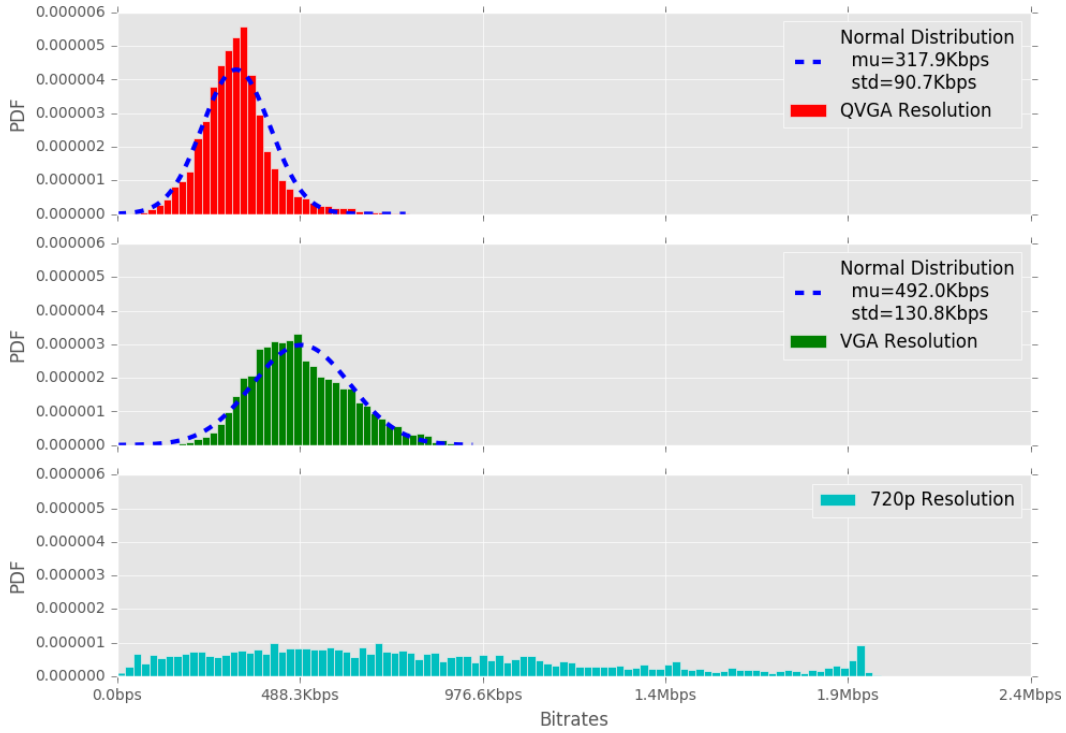

Figure 7.5 – Subscribers Probability Distribution Functions

of clients access networks (i.e. ADSL, 3G), together with it's effect on WebRTC software rate-control behavior[7]. That restricts the subscribers' statistically received bit rate and quality of experience.

The distribution of outgoing bit rates for Publishers in the 720p resolution shows the same distribution seen for Subscribers. If we compare the distributions for 720p Subscribers in Fig. 7.5, and Publishers in Fig. 7.6 it is clear that the limiting factor that causes low level bit rates is actually upload bandwidth. The publishers cannot keep up with the needed bit rate to stream 720p video, congestion is generated limited by the underlying network infrastructure, and rate-control algorithms keep bit rate lower than desired in order to avoid congestion.

7.4.2. Bit rate and Load correlation

From data, we can discern a clear pattern of decreasing bit rate with the increase in number of streams allocated in a data center. Such pattern provides a good limiting factor for QoS specific optimizations. Knowing this behaviour in advance, for a given bandwidth and server CPU capacities, we can limit the number of streams per machine in order to guarantee a minimum bit rate range based on resources. This can effectively lead to allocation algorithms that are QoS aware and try to optimize video quality on the platform.

We present in Fig. 7.7 temporal samples of average bit rates sampled over the machines of our test data center for a period of 9 Days in 2min client samples, in order to

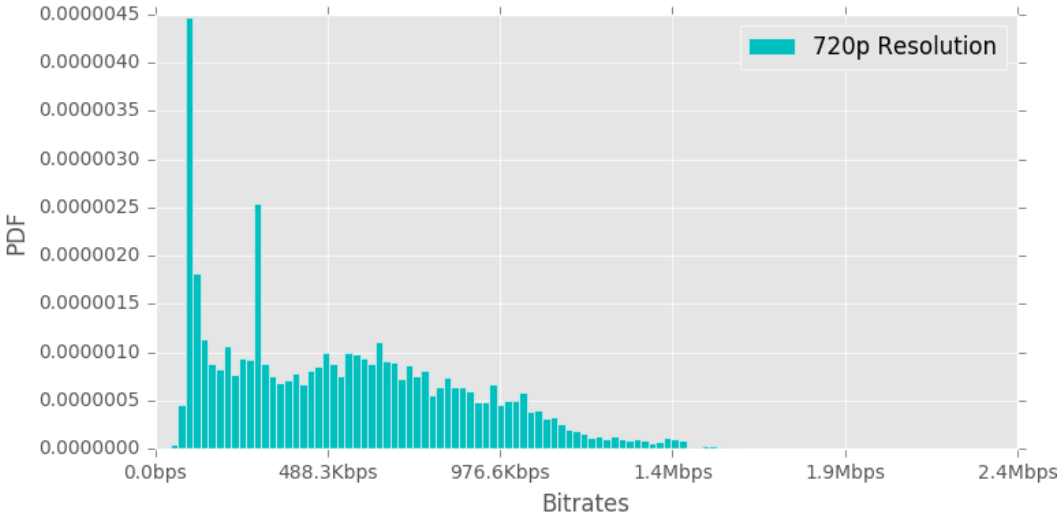

Figure 7.6 – 720p Publisher Probability Distribution Functions

try and find a correlation between load and impact on bit rate quality. We calculate the *Pearson* correlation coefficient and p-values in order to see if there is some linear correlation between load and bit rate quality. On these measurements, servers nerver hit their limit capacity, and as such, machine overload does not impact the results. *Pearson* coefficients in case of **QVGA** and **720p** are both 0 and show no correlation, although in the case of **VGA** we have a coefficient of -0.48 with p-values 0 which mean that there is a correlation between the two. Dwelling deeper into such correlation potential, we expand our study of bit rate in order to account for session sizes and also number of subscribers/publisher. The following section will introduce data for simulcast streaming, explaining better the drop in bit rates we have seen in correlation with the number of streams/server.

7.5. Non/Simulcast impact on quality

In case of non-simulcast video streams, given that only a single layer encoded stream is available, the publisher shall adapt bit rate to match an estimate of the worst subscriber bandwidth or some lower limit called **Minimal Bit rate**. On the other hand, simulcast publishers send all the available video qualities directly without limiting or adapting bit rates (as long as upload bandwidth allows). In the case of simulcast, the SFU chooses for every subscriber which quality it shall receive, matching at best, the available bandwidth. From such definitions, we can see that as more clients are subscribing to the same stream simulcast provides more flexibility, and permits to keep average bit rate more stable, avoiding the worst subscriber effect of pure packet relaying topologies, as *bad* subscribers typically are few.

The results of this section are derived from client side measurement using our test SFU-based cloud over a period of 14Days. We sample data for different distributed test machines

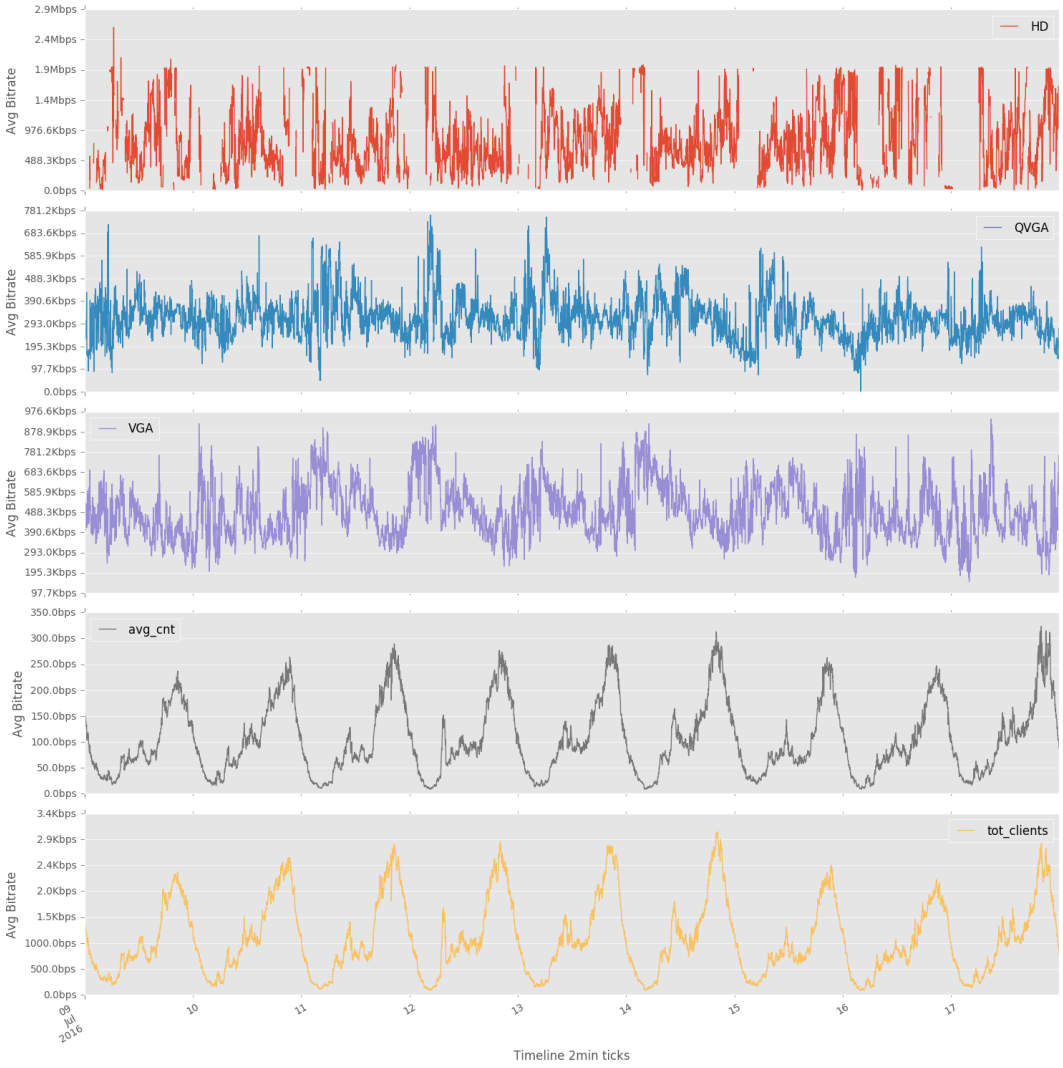

Figure 7.7 – Bit rate and Respective Loads over time

in order to avoid bias on data due to geolocation. Bit rate estimations should not be affected by physical allocation of cloud resources as they are sampled on all connections at the same time. We focus here on VGA quality for simplicity.

Inspecting the impact of topology on the average bit rate, we see that with an increase of #subscribers/publisher non-simulcast adapts the bit rate to match either the minimum bit rate limit of the system or to the lowest performing subscriber; thus, all subscribers being penalized. This behaviour is not optimal for sessions with a high number of subscribers. In Fig. 7.8, we observe that in general bit rates for simulcast (on the same platform) are equal or better than non-simulcast. They are, at most, comparable just for low number of subscribers/streams (at around 2 subscribers). As commented, the more the number of subscribers, the more the bit rate profile falls for non-simulcast, giving rise to

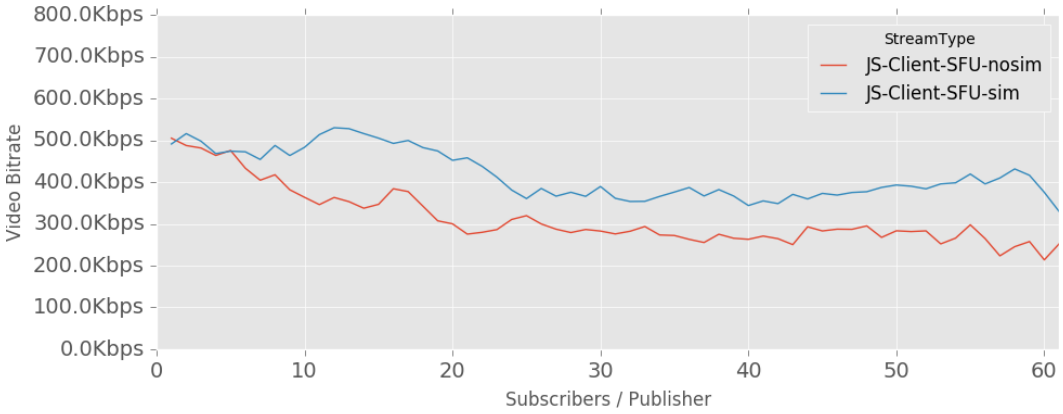

Figure 7.8 – Average Bit rate for #Subscribers/Publisher

the the patterns shown in Section. 7.4. This newly discovered pattern justifies the Pearson correlation seen in the decrease of VGA bit rate as load on the machine increases. With the increase of load we have higher number of sessions per machine and as the number of #subscribers/#publishers increases on those sessions the average bit rate falls further to accommodate the lower performing subscribers up to a minimum bit rate threshold. As such it is not only the load in itself that leads to a lower bit rate but the topology of the sessions as the presence of sessions with more than 3 #subscribers/#publisher already start impacting bit rate in a non-simulcast session.

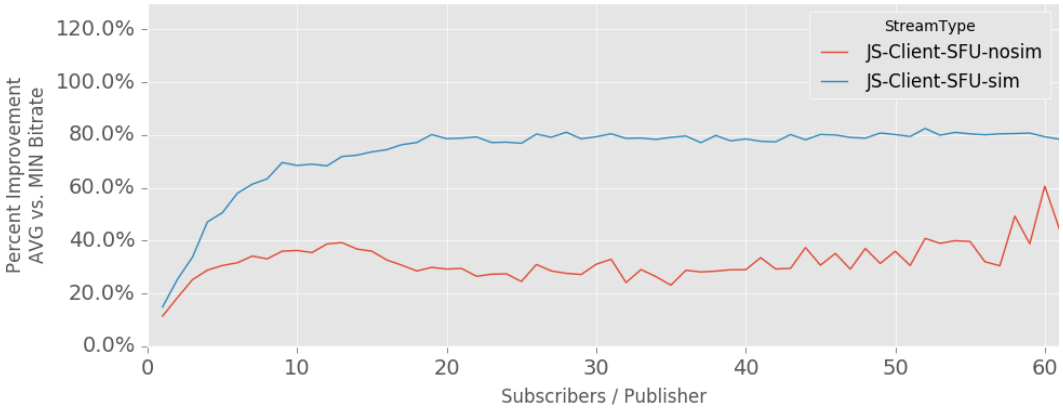

Figure 7.9 – Percent improvement AVG over MIN bit rate for #Subs/Pub

A good factor to further examine the impact of simulcast and non-simulcast publishers on the quality of the streams, is the relative difference of average set of bit rates to the minimum bit rate perceived by the worst of the subscribers from the same publisher. This parameter would present the relative perceptual improvement of the average bit rate over the minimum. In general simulcast outperforms non-simulcast by 2 times and also provide more stable bit rates as the number of subscribers increases. This behaviour is captured in Fig. 7.9, where we have the percent improvement over the number of #subscriber/publisher.

Simulcast performance can be tempting as a way to increase stability and scalability, but for low number of subscribers, like the case of One-to-One sessions, encoding and transmitting all video qualities may be a waste of client and network resources. In the case of mobile devices, simulcast may be beyond the possibilities of some hardware models, as it can easily take a 30% more CPU resources than the non-simulcast scenario for the same nominal resolution. This is only a small limitation at the moment of writing, as with the projected growth of mobile device CPUs, and with the inclusion of video encoder/decoder in the hardware architecture it will lead to more efficient encoding/decoding.

On a One-to-One user session the non-simulcast would not only need lower CPU usage but also would use the available bandwidth between the two users more efficiently as only one resolution layer is sent from the publisher. In WebRTC / RTCWEB, by means of Session Description Protocol (SDP) renegotiation, it is specified that it should be possible for endpoints to switch between simulcast and non-simulcast. However, some platforms may not support switching video transmission mode, thus knowing when to enable or disable simulcast becomes a very important decision or trade-off point. This further motivates our study of bit rate patterns and the results provided in this section can be used exactly to tackle this problem and drive trade-off decisions.

7.6. Monitoring and System Architecture

The testbed used to gather the data used in this work, is built in a Cloud environment that was constructed from a micro deployment of a full Cloud relayed WebRTC infrastructure. We allocate a number of bare-metal machines instances in a Cloud provider on which we deploy Service API, Control & Signaling Services and Media Selective Forwarding Units infrastructure. The data is sampled from both the server-side and the clients sides. WebRTC endpoints are software based components running a Javascript compatible client. All servers were allocated in the same data center and provided seamless access to the machines while each one of the sessions is allocated by a load balancer based on the actual load of servers. In our experimental setup, sessions are not permitted to span multiple servers in order to limit the spread of stream latency as well as increase system stability. The components of our monitoring systems and the testbed setup are shown in Fig. 7.1 where the relationship between the various components are presented. Our testbed incorporates the following components supporting a full WebRTC production level deployment:

1. **Service API** is the entry point to the infrastructure and is a HTTPS based application that functions as a resource selection service and authentication.

2. **Control & Signaling Services** Is a software component implementing HTTP message communication and is used to exchange messages based on Session Description Protocol (SDP) protocol specification, which is used to negotiate stream parameters between WebRTC endpoints. Each server is configured to have one Signaling service instance.

3. **Media Selective Forwarding Units** This software component is present in all of the servers allocated and handles media stream forwarding for the sessions that are allocated in each of the machines.

The **Service API** is a HTTPS based server which is in charge of allocating sessions and streams based on machine loads. In our test bed, there is only one centralized instance of the allocator service api as this is only used to decide where to allocate streams the first time that the request for a stream is made. In having such one time only usage nature, it is guaranteed that this system component is not a bottleneck for the use-case analyzed in this work. Service API is only responsible to direct client to the Signaling and SFU servers, and does not transport media traffic, or is subject to high load in any case.

Service API Units Service	x1
Signaling Units Service	x9
Selective Forwarding Units Service	x9
Total Servers	x9

Table 7.2 – Testbed Service Count

In order to setup endpoint capabilities a **Signaling Service** is needed that takes care of configuring endpoints with compatible features. In general in WebRTC the signaling component is not a media transport protocol, and as such is used only to setup the stream parameters and does not transport any kind of media. For this work, endpoints implement a websocket-based signaling protocol that is used to exchange SDP compliant messages and to setup the media protocols and capabilities for the actual audio/video communication. WebRTC as a standard does not define specific protocols for signaling on purpose. The choice for signaling is a non-functional API of the architecture which can depend on the application and/or use-case. The application nature of endpoints make a standard for signalling the less and less necessary now-a-days. Through using this signaling service, the endpoints negotiate a common set of features that are supported by them all and enables direct or relayed media inter-communications, and keep the status of the call updated.

At last the **Selective Forwarding Service** component is loaded in each of the servers of our setup and is responsible for implementing audio/video communication relay units. Media communications are implemented using the RTP/RTCP protocol and can accept streams of custom video resolutions based on VP8 video codec. Supported audio codec is Opus which is the Mandatory to Implement choice for production ready WebRTC environments.

The exact number of resources used in conducting this study are listed in Tab. 7.2. We have a total number of 9 bare-metal machines hosted into a Cloud resource providers, on each one of the machines we have deployed both a **Signaling Service** component as well as a **Selective Forwarding** media component. On one of the allocated machines we have allocated one instance of the **Service API** responsible for allocating sessions to servers.

7.7. Load balancing evaluation and experimentation

Further experimentation has been conducted by comparing the performance of various allocation algorithms, and in turn verify the efficiency of the proposed algorithm (Alg. 2). To drive the comparative analysis of such stream allocation policies, a dataset of traces of real WebRTC sessions was extracted from our test cloud. These traces were used as simulation events for the resource allocation algorithm of a simulated cloud environment. The various load allocation techniques are built so that no user information outside of stream parameters provided in the WebRTC[78] standard are used.

7.7.1. Experiment Setup

Experiments on the performance of the various stream allocation policies were conducted in a simulated cloud environment, using task parameters to mimic the real *Test Cloud* environment. The simulation environment was implemented by extending the CloudSim [17][82] platform with a custom package to handle WebRTC streams. Normal operations of the CloudSim framework support batch processing tasks, we have coded in our extension package the ability to process WebRTC streams defined by the following parameters: ***Stream Identifier***, ***Session Identifier***, ***Stream Start Time***, ***Stream Duration***, ***Allocated Server Identifier***. The extended simulation environment supports the creation of multiple data centers, each one with an arbitrary number of Hosts, on which a single Virtual Machine (**VM**) is allocated. The package permits the configuration of the number of Hosts/VMs and also an upper limit on the number of streams that can be processed without SLO violations. During simulation run-time each VM can process in parallel as much streams as configured by the maximum streams parameter. Once the simulation is running if the number of streams surpasses the maximum configured number per VM, the additional streams will be executed in a time-share manner with the existing streams on the machine. In case that such streams are running in a time-shared fashion, the completion time of these streams would be delayed causing a violation of the SLOs. This aspect of the simulation permits to directly detect the performance of the stream allocation strategies, not only in terms of number of streams processed by the server, but also as reduced quality streams that would not meet real-time requirements.

As CloudSim is a Java based simulation environment in which the simulation entities are costructed in active Java Threads, extra care need to be given to the choice of active components allocated for each real processing unit in the evaluation machine. The server in which the simulation is executed has 8 CPU cores which are used to execute 9 Data Center/Vms thus the number of active components is not much larger than the number of cores. To further verify that the simulation of the sessions is not impacted by the specs of the server we provide a comparison of scheduling the original data on simulation and verify each tasks start/finish times. In Section.7.7.2 we compare such data in order to establish a baseline for the simulated environment.

For our experiments we have sampled one week of real system traces from our **Test Data Center**. We have selected a period with data concerning both normal operational loads and a particular day with a sudden spike in load behaviour. This data ensures that

we have a general view of the implemented algorithms for both spiked data and normal operations. Our simulation environment is made up of: **1 Data Center**, **9 hosts**, **9 VMs** and **196 325 WebRTC Streams**. First we examine an algorithm that mimics the same stream allocation algorithm as the original data, which is also our baseline for testing the simulated environment. By comparing this algorithm to the original data we prove that the simulation is valid and calculates the same load profiles as the real **Test Data Center**. Other algorithms were implemented based on: **Static Threshold**, *Minimal Load*, and as well as *Rotational* algorithms such as Round Robin variants. Each one of the implemented algorithms is compared to the others and the baseline. As previously seen and observed in

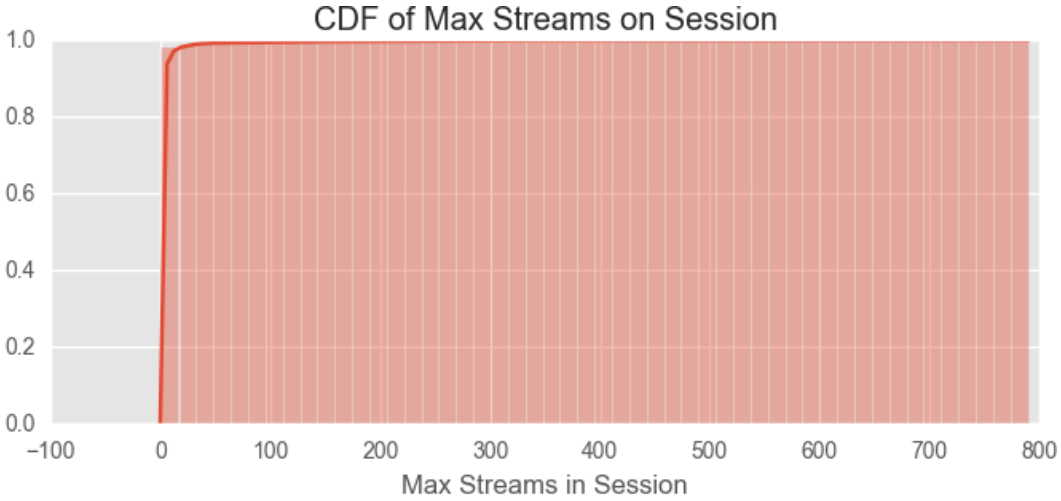

Figure 7.10 – Session size distribution

Fig. 7.10, the distribution of session sizes follows a tail distribution in which the majority of the sessions have a small compact size while there is a long tail of big sessions which needs to be handled. The distribution shows that high spikes in session sizes or sessions with very high number of streams are rare and exceptional events. This makes it very unlikely for such sessions to have similar inter-arrival times and lower the impact of concurrent big sessions to the system. Such distribution explains the choice of algorithms we use in the comparative analysis. The fact that the majority of the sessions are limited in size up to 10 streams per session leads make a clear case for Round Robin algorithms as the session sizes are mostly homogeneous with only a small part of the sessions having sizes of up to 300-400 streams. These observations are also verified by the evaluation of the algorithms, where the RR algorithm performs quite close to the min-load algorithm which gives the best results in terms of maximum load reached on the servers.

7.7.2. Establishing a Baseline

Verification of the simulated environment was provided through the implementation of the **Preset Allocation** algorithm. The algorithm reads the input data and schedules the tasks to be processed at the same machine as the real system, with the same defined upper

limit of **1 100** streams/machine and same **Stream Start Time**. The implemented baseline algorithm mimics the same configuration of the original system data sampled from the **Test Data Center**. We define in the context of this work the concept of SLO violation as streams which duration, or completion time is higher than the original duration of the stream.

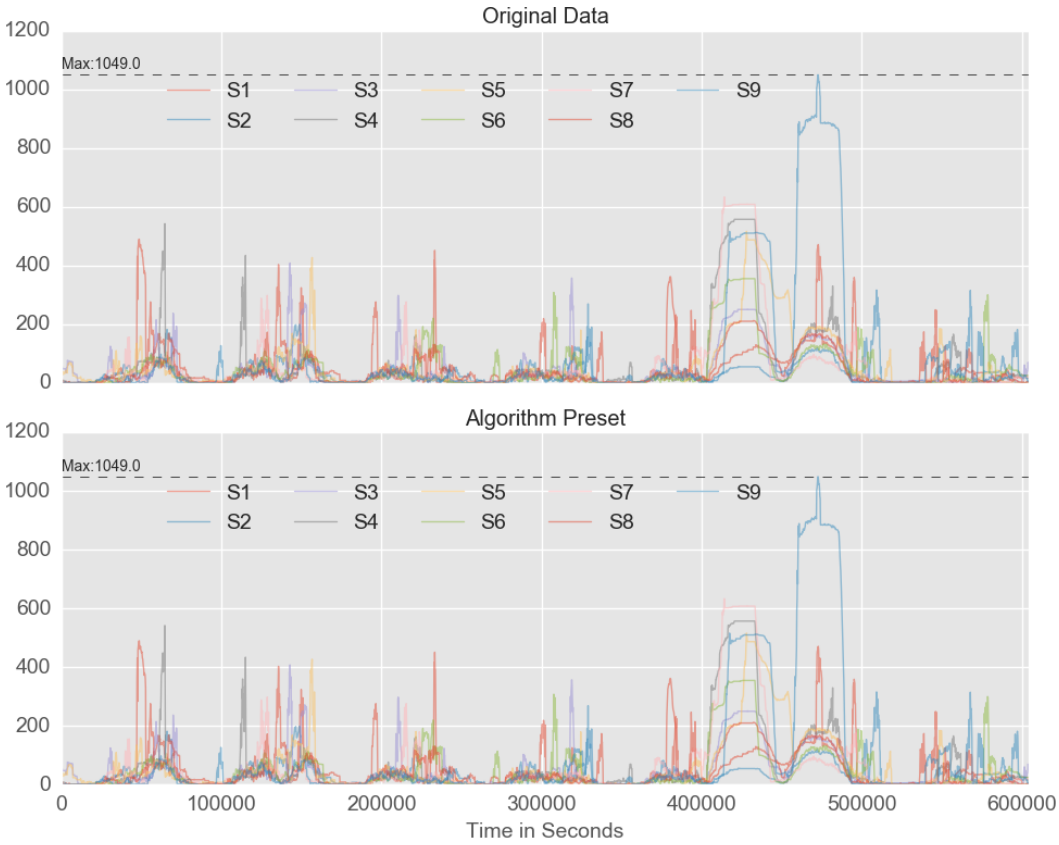

Figure 7.11 – Comparison of Original Data with Simulation

The simulation finished with no violations of SLO and both **Stream Start Time** as well as **Stream Duration** are identical to that of the original data. The load profiles for both the original data and the simulation are presented in Fig.7.11, these profiles are totally identical. This data verifies our simulation environment and provides strong evidence of a stable and repeatable way for conducting further experiments.

7.7.3. The Minimum Load Algorithms

In order to try and minimize the maximum utilization in terms of streams in our servers, and lower the impact of peaks in form of big sessions, we experimented with minimum stream load algorithms. All the experiments were conducted by using the same parameters of our **Test Data Center** environment as the setup previously described when establishing our baseline (Sec. 7.7.2).

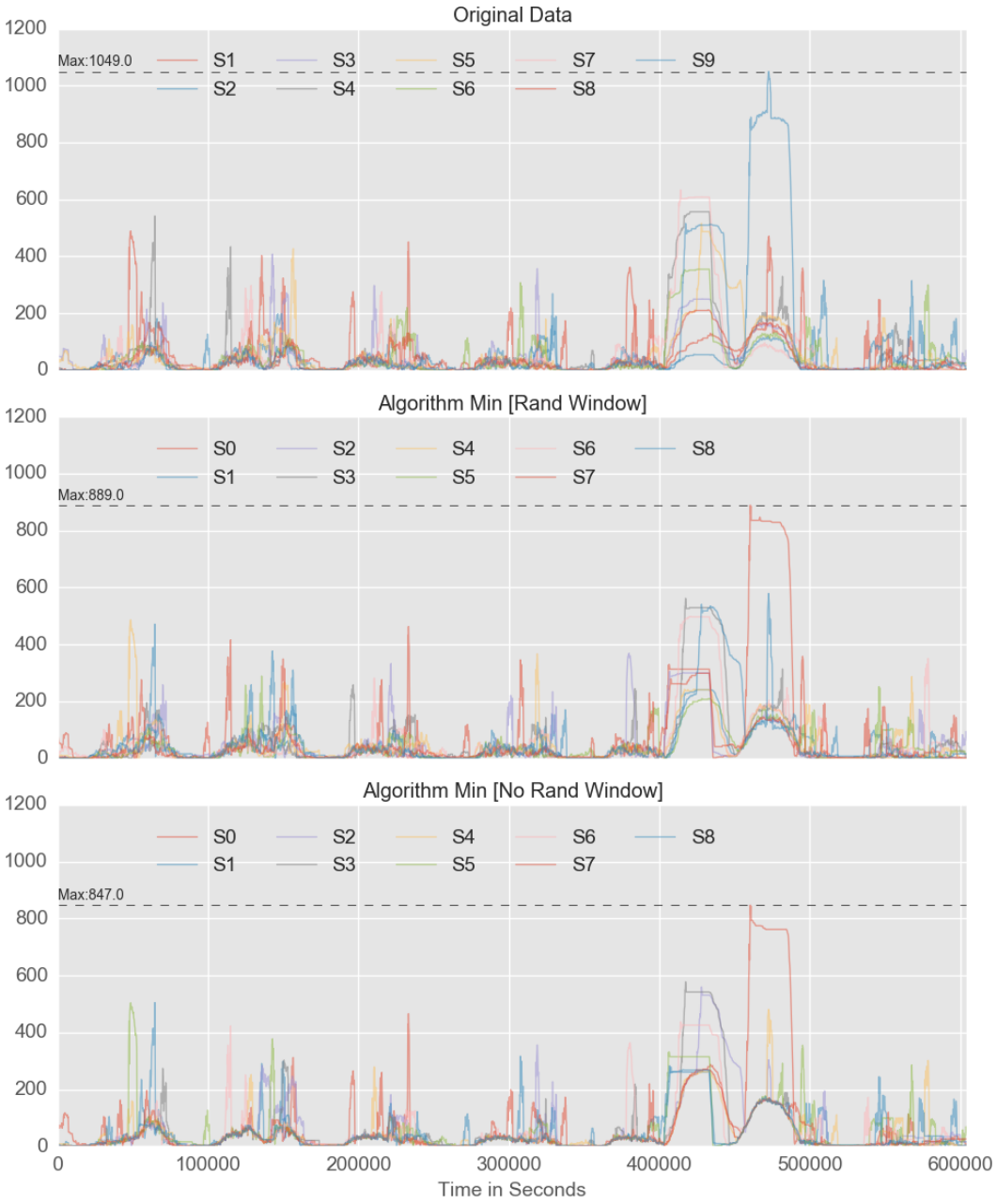

Figure 7.12 – Comparison with Minimum Load Algorithm

The first experiment provided is though Alg. 2 with a *threshold of 30* streams resource occupation difference for the minimal load set. Within the minimal load set, the selection of server where to allocate the next streams is done through randomly selecting one server candidate. We observe from our experiments that this algorithm already lowers the maximum observed utilization seen on all servers by 15%. This policy implements a

minimal load algorithm with randomized selection windows and the results are presented in Fig. 7.12. Although this algorithm performs quite good, we can still improve over it by using a purely Minimal load selection algorithm. The algorithm plainly chooses for each new stream the server with minimal load without randomizing a set of minimal load servers. The results presented in Fig. 7.12 last subplot shows an additional improvement over the maximum load reached for all servers in the baseline, scored at 19% lower than the max utilization of the original data. Both algorithms provide a noticeable improvement over the other algorithms and thus were selected to tackle the problem of stream allocation.

7.7.4. Round Robin and Static Threshold

One other family of algorithms that were implemented and evaluated were rotation based algorithms implemented both as a per stream rotation and a threshold based rotation. The stream based rotational algorithm performs better than the original data and that the threshold rotation algorithms, but under-performs as compared to the minimal load algorithms. We observed that the maximum peak reached for all servers on the **Round Robin** stream allocation algorithm scores a maximum peak value of **925** and as such doesn't generate any SLO violations (Fig. 7.13). In these experiment results peak utilization do not reach the maximum number of streams per machine. Such numbers hold also due to the fact that in general a low number of #subs/#pubs dominates session sizes. The incoming rate of big sessions (more than **50** streams) is very rare but still impacts the overall performance. Clearly the impact of these rare big sessions is seen in the class of algorithms based on Static Threshold.

The class of *Static Threshold* algorithms exhibits very big spikes in streams that do not only surpass the thresholds but as well the maximum number of streams that the machine can handle, creating SLO violations. We implemented a *Round Robin* variant, in which servers are not rotated until the resource occupation matches a given threshold. This technique has benefits as the servers needed to handle the load are in average less than the allocated number of machines. Even though we have a clear benefit in number of machines being used, the peaks reached during operations show an unacceptable side effect by producing a high number of SLO violations in terms of server overbooking. We show the result of such experiments in Fig. 7.14, where peak utilization of the servers reaches a values of **1 484**.

The other implemented algorithm of the threshold family is the static threshold with minimal load utilization policy selection. After the threshold utilization for a server is reached the next stream is allocated to the server with minimal load. Both these algorithm are fragile and prospect to be impacted by big sessions assigned to a server that has reached near threshold utilization. As previously discussed in this case the peak utilization of such server reaches utilization factors of **134,9**%, which is unacceptable for real production scenarios.

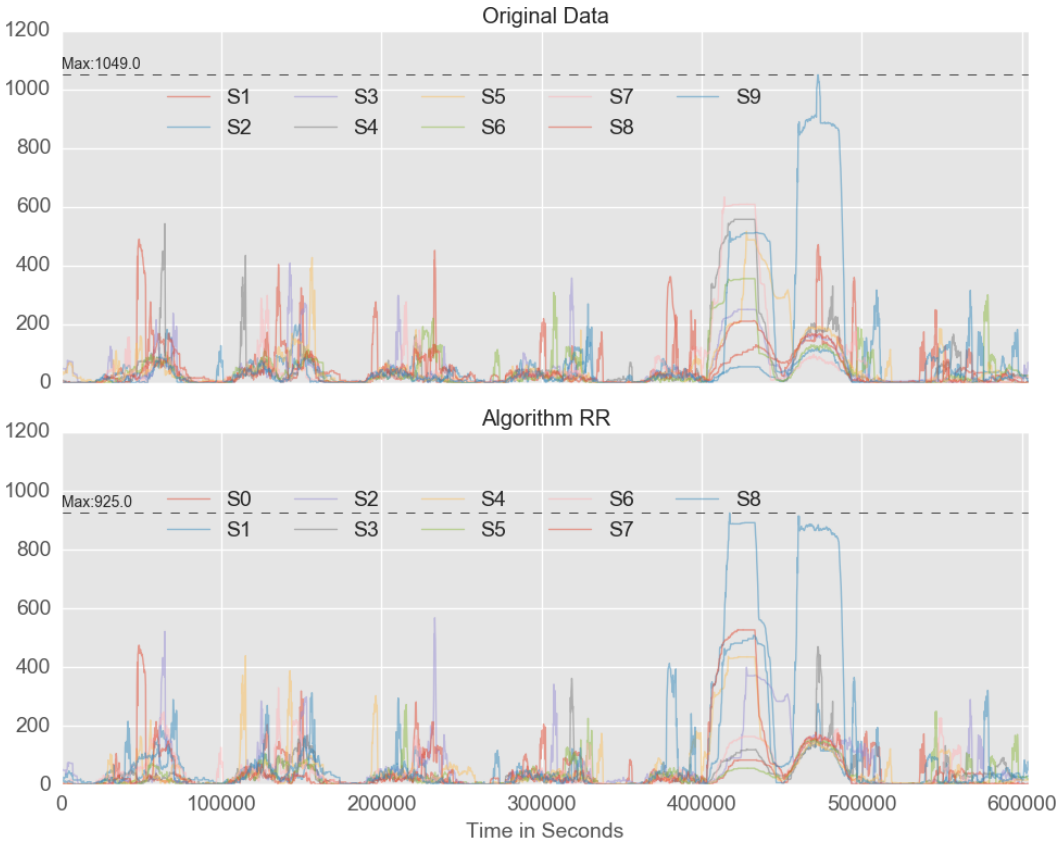

Figure 7.13 – Comparison with Round Robin Algorithm

7.8. Conclusions and Future Work

Building on these previous state-of-the-art, we provide insight into load patterns on real large scale WebRTC production systems. Media performance is characterized through video bit-rate profiles for a SFU-based cloud supporting both simulcast and single source rate controlled video streams. Furthermore, we provide an extended study of how session topology load in the form of subscribers/publisher influence media bit-rates. We have shown from real client measurements that video simulcast provides benefits as big as 2 times higher than the minimum perceived bit-rate in the non-simulcast case. Constructing on the discovered load and media patterns, we have provided a session allocation algorithm to fairly distribute big sessions among machines.

Different algorithms were evaluated in order to verify that the selected *Minimal Load* algorithm really behaves better in terms of peak utilization reached as compared to *Round Robin* and *Static Threshold* algorithms. All the algorithms that were discussed need minimal information on distributing the load and guarantee user anonymity toward the platform. This is an important aspect of running a Platform as a Service, and also in terms of insurances that can be given to both the users and the software implementers. We observe that the minimal load algorithms perform better than the rest of the observed

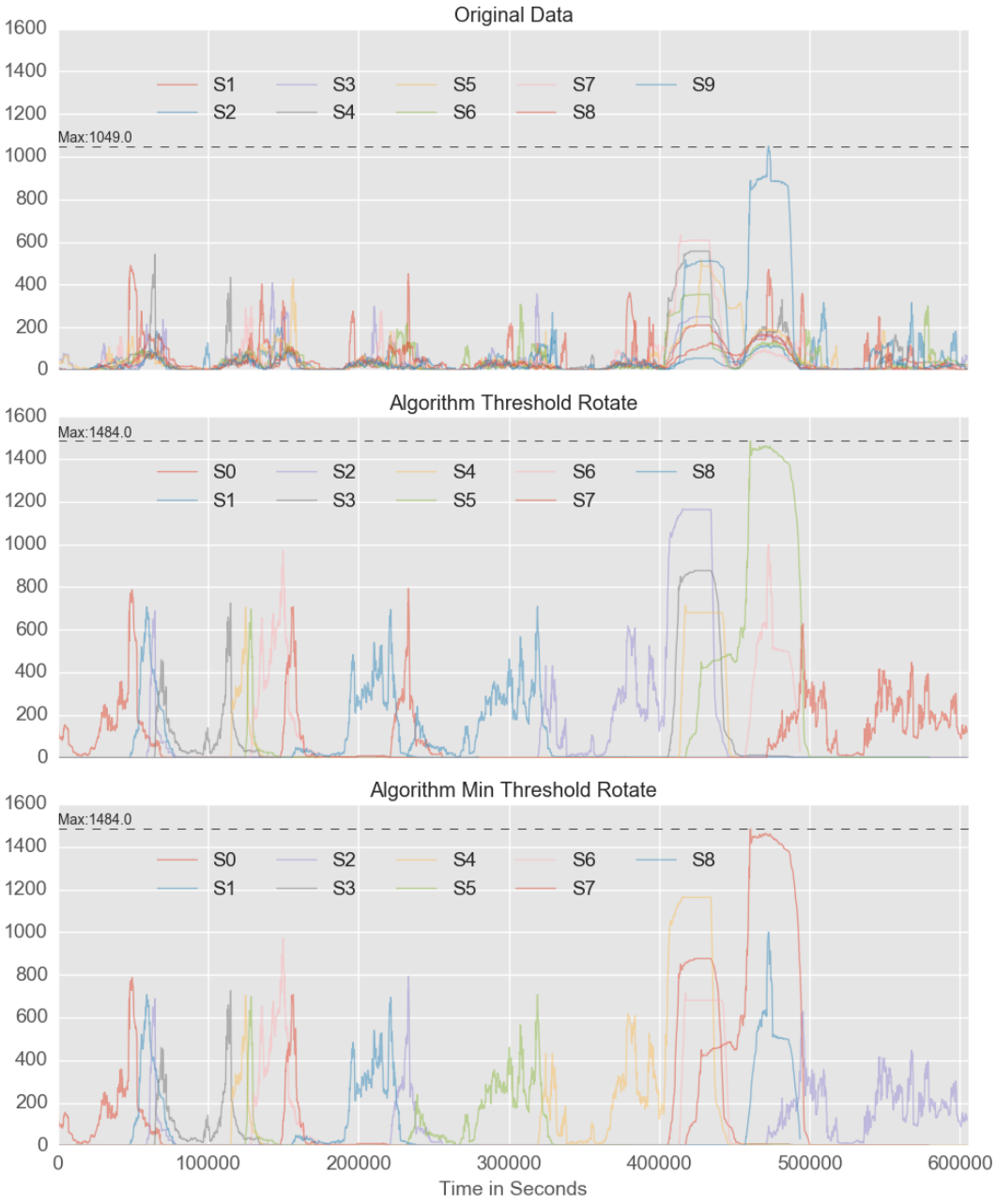

Figure 7.14 – Comparison with Static Threshold Algorithm Variants

algorithms and optimize peak utilization by exhibiting peaks of **19**% lower than the real allocation and up to **50**% lower than the static threshold algorithms.

Future work includes providing load allocation algorithms for multi-cloud deployment SFUs to optimize either cloud resources or user latencies. Further work can be conducted to investigate the alternate model in which sessions can span multiple SFUs and servers.

This new system bypasses restrictions of single server session allocation treated here, but introduces new challenges on trunking and added latency to the stream. The introduced challenges are in the form of system stability, as each failure in bridging inter-connectivity can lead to heavy network partitioning.

Chapter 8

Conclusions and Future Work

8.1. Conclusions

This work provides results in both managing and optimizing distributed media stream processing applications through Hybrid Cloud and Network architectures. We tackle various problems related to these kind of delay no-tolerant applications in terms of Network based limitations such as latency/locality, bit rate/bandwidth, Cloud stream forwarding load. Each one of the contributions of the work were peer reviewed in conference publications and the derived implementations were evaluated in real world deployments.

A core problem addressed in this work is related with whether network bandwidth can be management at the Cloud service endpoints to impact system stability in terms of bit rate/bandwidth, toward optimizing service performance. Bit rate is an important factor in real-time media streaming applications as it directly dictates quality on both audio and video. To tackle such issue we provide a Network Bandwidth Manager for Cloud services that share the same network infrastructure. The provided controller operates on the end-points of Cloud services and uses a off-line calculated predictive model for the planing phase of a MAPE controller. Our evaluation of the proposed controller was conducted by analysing the operations of a Cloud Object storage system with arbitrating between two types of traffic the user-centric workload and system-centric workload. **The proposed model leads to a gain of at least a factor of 2 in system stability as compared to the un-controlled system and provides a good way to trade-off bandwidth between Cloud services**. This model ensures that network becomes a first class resource of the Cloud model, and provides stronger performance SLOs. During operations using this technique bandwidth can be mediated between client real-time streams and system non-functional concerns, such as accounting etc... Thus by slowing down ingestion of non-function data we can enhance the performance of the functional part of a real-time media stream processing service.

The Access Networks (edge, core network) of ISP, Carriers and Community Networks have no general purpose cloud infrastructure, while Internet Resource Providers provide on demand Cloud resources. We find an opportunity for the unification of the resources inside

an AN and outside in order to provide a unified cloud offer through Cloud Federation and provide service mobility toward the users in order to optimize locality. **Our proposal is a Multi-Cloud novel Cloud architecture merged inside the Network infrastructure, that enables services to be placed as close as 1 hop away from the user.** This novel architecture permits to optimize locality and to scale the applications by trading-off with service performance. Compared to previous work done in the field, our approach provides a hierarchical federation model, which permits to trade between locality (the further closer to the edges, you deploy cloud services in the hierarchy, the closer you get to the users) and service performance (the further away services move from the edges, the number and performance of available resources is increased). We conduct a structural comparison of the proposed architecture to existing cloud architectures of comparable scale.

Further building upon the presented hybrid Cloud and Network architecture with the objective to demonstrate further latency improvements to known real-time media stream processing applications. We present a self-organized large scale Cloud live streaming service which with minimal management overhead closely approximates the optimal solution up to a error of order 0,02 and provides structural improvements as compared to previous state-of-the-art. Such self-organised overlay makes is resilient to cloud failures and shows a quick convergence toward a stable state. **The algorithm builds a Inter-Cloud stream forwarders communication backbone through a distributed overlay, which is proven through simulation results to converge in range of 6-15 communication rounds.** These results were calculated by varying the size of both edge clouds and Internet Public Clouds in ranges of thousands of Cloud providers, which is orders of magnitudes higher than the number of real cloud providers in real world scenarios.

Quality of media in a real-time media stream processing applications is an important factor in evaluating the performance of the system. In this work we focus on whether Cloud based stream forwarders load does impact media stream quality. Additionally we focus on investigating if Cloud based stream forwarders load can be balanced in order to enhance stream media quality. We conduct an analysis of a production grade WebRTC/RTCWEB system in order to find patterns correlating Cloud backend workloads and stream quality. **We present empirical results that show an improvement in video media bit rate when using simulcast video encoding (all desired video qualities encoded in parallel on the same stream) over congestion controlled single layer encoding (with a dynamic bit rate and one single quality encoded at any given time) as session size increases.** We observe further load patterns and devise a stream allocation scheme for sessions not spanning multiple servers. **Our approach provides improvements of maximum peak utilization per machine from 19% to 50% in comparison with the other algorithms that were analyzed**. The limitation of having one server only restricts the number of viable algorithms, but future work will be focused on trunking and multi-server sessions in order to overcome such restriction.

8.2. Future Work

Future work can be focused on further improving micro-cloud based, Cloud architectures in various ways as well as shift the focus on WebRTC/RTCWEB cloud backed SFUs and MCUs. As more of the Cloud market is further concentrating into a number of big players such as Google, Microsoft, and Amazon but no limited to them. Also community cloud is emerging as a highly distributed Cloud architecture based on voluntary donating or micro-economy based Cloud deployment. Almost all of these providers insure distributed Cloud environments with locality areas spanning all over the world. The reach of data center coverage increases and gets closer and closer to the user, there is still a need for more efficient Inter-Cloud federation algorithms both for Hybrid Cloud, Cloud Federations as well as Multi-Clouds.

As this work is focused on Inter-Cloud architectures and application improvements, the continuation of it can be directed on further analysing WebRTC/RTCWEB applications for real-time audio/video conferencing backed by Cloud backbones. This type of applications provides a good research area in which the Cloud feature should adapt itself to the ever changing network conditions and topologies in order to ensure stream delivery in a delay non-tolerant manner. Continuing with these kind of real-time applications of particular interest is the ability to continuously and dynamically scale the Cloud backbone in order to minimize cost and optimize stream quality metrics. The impact of Cloud resources in terms of VM costs and Bandwidth allocated, starts getting prohibitory with large increase in client workload. Managing such growth in an optimize manner lowers by much the cost of running a Platform as a service based on web real-time communications.

Further contributions can be given to improve stream resource allocation algorithms introduced in Chapter. 7. The work done so far considers monolithic allocation algorithm in which the first stream of the session dictates which machine the following streams should end to. Such limitation can be overcome by using trunking between SFUs, which means that a virtual sender and receiver are built to forward the streams between servers. Trunking permits to offload the needed work to handle a session between SFUs but introduces limitation in the aspect of increased latency and system resiliency. An increasing the number of hops for the signal to reach the desired destinations leads to increased latency, that can influence stream quality. Since for each server there would be a trunk from the original server and a number of clients producing and consuming content, we have a potential case of network partitioning in which if the Inter-Server link breaks, the clients associated with that server would be isolated from the rest of the servers. Further research into minimizing latency and increasing the level of fault resiliency of the system would help improve such load balancing schemes.

Another road ahead into increasing stream quality for WebRTC/RTCWEB sessions based on Cloud backends is the discovery of novel congestion control algorithms. The congestion control algorithm helps estimating network bandwidth and keeps the network from being overflowed thus ruining stream quality. The GCC congestion control algorithm works well and is resilient to TCP connections sharing the network with UDP WebRTC traffic. Even though this is a good feature it is sometimes too aggressive and as such on low bandwidth sources could lead to congestion due to continuously trying to probe network

for more bandwidth. New algorithms need to be discovered that perform well in all cases including Mobile, Web or Hardware clients.

Appendix A

Building a Cloud Simulator for WebRTC workloads

A.1. CloudSim Introduction

CloudSim is a simulation framework which implements the basic components of a Cloud environment in a discrete event simulator model. The components we are intersted in and presented in CloudSim are the following:

1. **Cloudlet** a Cloud task to be executed in the cloud environment.
2. **VM** an abstraction representing a virtual machine, which executes Cloudlets.
3. **Host** a model which represents a physical host in a data center in which VMs are executed.
4. **Data Center** the abstraction of a data center which contains multiple Hosts.
5. **Data Center Broker** is the main scheduling entity which receives a series of Cloudlets as input and schedules their execution in different datacenters.

CloudSim is modeled in a layered design as shown in Fig. A.1 with code divided into **User Code** and **CloudSim Code**. Users of the framework usually need to extend the components in the **User Code** in order to implement novel Cloud models and Scenarios. The code implementing such scenarios can be directly evaluated through the run-time of the framework. Each software component implemented in CloudSim is called a *Simulation Entity* and implements a message passing model. Simulation Entities have a main cycle that is executed each time the event scheduler wakes the component up to process events waiting to be processed. Each message is queued and delivered at the appropriated component to be processed at the required simulation time.

The simulation engine, implements a discrete event simulator model which uses the common Simulation Entity interface to run the simulation. On top of the run-time components of the framework we find a **Network Layer** that implements network

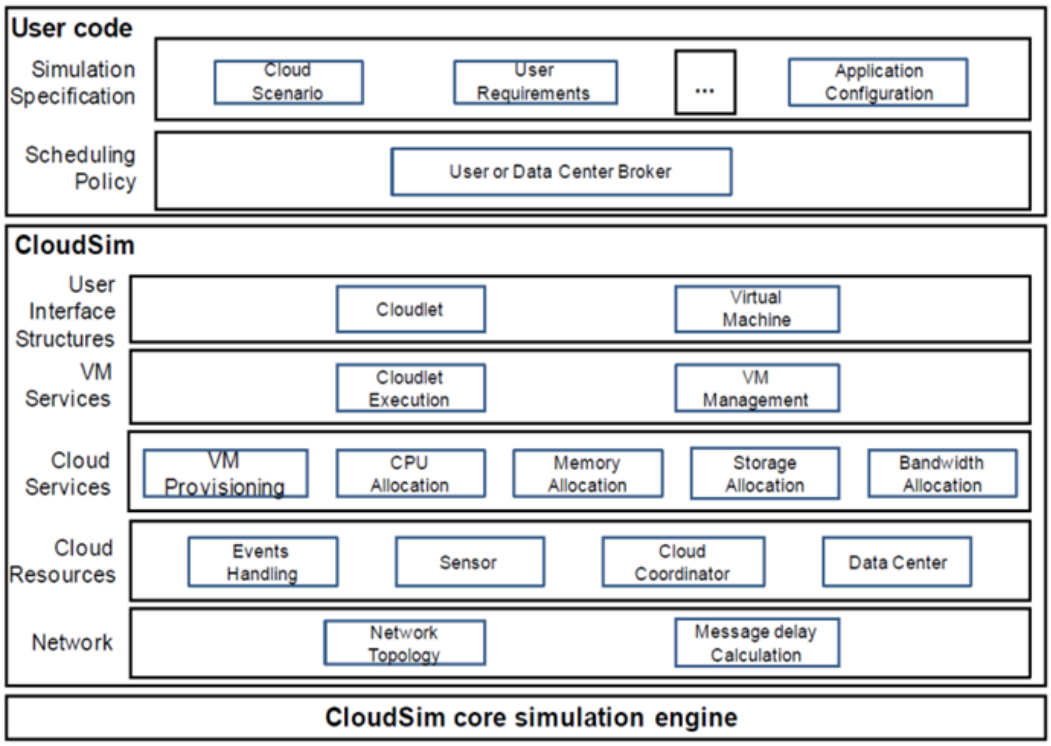

Figure A.1 – CloudSim architectural overview [8]

properties in which the simulation scenarios can model various topologies and delay profiles. Applications targeting network studies can thus model the desired properties of the underlying inter-connectivity in details.

Cloud Resources layer implements tha basic resource level reppresantation of real hardware resources such as Cloud Coordinator, Data Center etc... In order to permit experimentation with various Cloud models and scheduling the **Cloud Services** layer exposes configurable and extend-able allocation and provisioning policies of resources. **User Interface Structures** and **VM Services**, implement the cloud tasks that would executed during the simulation (Cloudlet) as well as VMs that execute such tasks. All the processing resources in the framework use a measure of Millions Instructions Per Second (MIPS) in order to simulate CPU micro-instructions execution length or capabilities. Thus each component when active calculates how many MIPS can be allocated and how many resources can be executed in a time or space shared fashion.

The main entity that users needs to extend in order to run simulations of dynamically allocating and scheduling Cloud resources is the Data Center Broker which takes care of allocating VMs to datacenters as well as scheduling Cloudlets into such VMs. The scheduling of such resources during the execution of the simulation can either use predefined CPU, Host and VM scheduling policies based on space and time sharing, or custom allocation policies that the user can provide.

The default Data Center Broker policy receives a set of CloudLets and schedules them at the different available Cloud resources independent of the task inter-arrival time. This model does not suit our intended WebRTC environment because the inter-arrival time of each WebRTC stream is important at dictating resource allocation. The framework is not able to schedule the streams at different inter-arrival times as the Data Center Broker is not an active component as such it is only able to schedule once all the tasks and patiently wait for all responses.

A.2. WebRTC Stream Simulator

In this section we describe in general lines the extensions conducted to the framework in order to simulate the execution of WebRTC streams and Cloud Forwarders that forward the streams between end-points. We implement streams and VMs based on WebRTC dynamics by using CloudSim primitives and extending them with the necessary properties. We represent each stream with a model in which each stream has the following run-time properties:

1. **Stream Arrival Time** The time in which the stream started to send data.
2. **Session Identifier** The identifier of a session to which the stream belongs to.
3. **Stream Identifier** The identification of the stream, a combination of the stream origin and destination identifiers.
4. **Stream Duration** The duration of the stream.
5. **Original Host** The original host on which the stream was allocated.

For such simulation environment to be able to conform with the requirements of a stream based Cloud forwarder, we implement a **WebRrtcCloudlet** which extends the Cloudlet and includes the additional properties listed above. This extended Cloudlet needs to transform the stream based model into a batch processing one, that can be executed by the CloudSim simulation engine. The extended WebRrtcCloudLet has a stream arrival time, a session identifier a duration in terms of MI, and a original destination VM. Every host created for the WebRTC simulation admits only 1 VM as described by the limits presented in Equations A.1 to A.6.

As we need a way to simulate the duration of the stream in the existing MIPS based VM, CPU and Host execution unit. We represent the duration of the stream in multiples of the length of units of MIPS that one virtual CPU can execute as MI per second. As such the resulting time is exactly the same as the duration of the stream. Each VM in turn is created with the same number of CPUs as the limit of streams that a VM can execute without contention in each time frame, thus if contention is present multiple streams would need to share the same CPU and thus the two streams would have longer duration that the original data. The longer duration of the stream is a violation of the Service Level Objective as it

would mean that the users in that stream were not satisfying the delay limits for a stream to be real-time.

Equations. A.1 to A.6 represent exactly the described configuration of resource capabilities. In our evaluations we selected a base MIPS value is 1 MIPS and as such it means that each VM has $n_{streams}$ CPUS, each of them can execute 1MIPS, while a task with duration t has a length in the order of $t * 1MIPS$.

$$u \ = \ Base\ unit\ of\ MIPS \tag{A.1}$$

$$n_{streams} \ = \ Limit \ \ of\ Streams\ per\ machine \tag{A.2}$$

$$WebRTC_{stream\_duration} = duration_{seconds} * u \tag{A.3}$$

$$VM_{cpus} = n_{streams} \tag{A.4}$$

$$CPU_{mips} = u \tag{A.5}$$

$$Host_{mips} = n_{streams} * u \tag{A.6}$$

A.3. Cloud Broker Algorithm

The Cloud Broker algorithm is based on an active component, which aligns itself with each arrival time boundary of tasks to be processed and schedules the Cloudlets that coincide in that inter-arrival boundary. After scheduling the necessary streams the broker waits for tasks terminations and as well schedules itself to be executed in the next time boundry that has at least one task to schedule.

Algorithm 3 Datacenter Scheduler

1: **procedure** SCHEDULER($allocator, Cloudlets_{left}, Cloudlets_{submitted}$)
// $allocator$ is a allocation policy wich given a broker and a cloudlet decides the destination VM

2: $Cl_{submitted} \leftarrow [\,]$
3: **for** $cloudlet$ **in** $cloudLetsInNextBoundry()$ **do**
4: $vm \leftarrow allocator.getVm(this, cloudlet)$
5: $send(vm, tag.CLOUDLET\_SUBMIT, cloudlet)$
6: $Cloudlets_{submitted}.add(cloudlet)$
7: $Cl_{submitted}.add(cloudlet)$
8: **end for**
9: $Cloudlets_{left}.removeAll(Cl_{submitted})$
10: **if** $cloudletList.size() != 0$ **then**
11: $schedule(this, getNextArrivalTime() - lastArrivalTime, scheduler)$
12: **end if**
13: **end procedure**

Algorithm. 3 presents a time based discrete event scheduler, in which Line.3 selects all the CloudLets arriving in the next inter-arrival time boundary. The next time boundary is selected in **cloudLetsInNextBoundry()** method by first ordering in ascending order of inter-arrival time all streams left to be scheduled and selecting the head of the list that share the sime inter-arrival time. In Lines.4-7 an allocator algorithm is used to decide which **vm** to use in executing the CloudLet. Line.9 removes the submitted streams from the queue of streams that are still to be scheduled. To terminate the algorithm, Lines.10-12 check if more streams exist that need to be scheduled and schedules the execution of the scheduler at the next inter-arrival time boundary.

The allocators that were implemented for this work to evaluate the execution of stream forwarding by Cloud SFUs in WebRTC range from host selection as in the original data, Static Threshold, Minimal Load, Round Robin to Randomized models. Each one of the allocation algorithms implements checks based on session identifier for the stream. Allocation of successive streams that belong to the same session are directed to the same host where the first stream of the session was allocated and bypasses any allocation policy.

Bibliography

[1] Ericsson Summer Report 2016. https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf. accessed: January 2017.

[2] Openstack. Openstack cloud software. http://www.openstack.org/software/, February 2014.

[3] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. Interconnected Cloud Computing Environments. *ACM Computing Surveys*, 47(1):1–47, may 2014.

[4] A. Gupta and R. K. Jha. A survey of 5g network: Architecture and emerging technologies. *IEEE Access*, 3:1206–1232, 2015.

[5] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, 13(18):1587–1611, 2013.

[6] Microsoft Azure. Azure media services rtmp support and live encoders. https://azure.microsoft.com/en-us/blog/azure-media-services-rtmp-support-and-live-encoders, December 2016.

[7] The webrtc project. https://webrtc.org/.

[8] SaaS CloudSim. Cloudsim general overview. http://cse564.wikispaces.asu.edu/1, December 2016.

[9] Nikolay Grozev and Rajkumar Buyya. Inter-cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 44(3):369–390, 2014.

[10] The Internet Engineering Task Force. Real-time communication in web-browsers. https://datatracker.ietf.org/wg/rtcweb/documents/, November 2016.

[11] David W Chadwick, Kristy Siu, Craig Lee, Yann Fouillat, and Damien Germonville. Adding federated identity management to openstack. *Journal of Grid Computing*, pages 1–25, 2013.

[12] Javi Jimenez et al. Supporting Cloud Deployment in the Guifi.net Community Network. In *5th Global Information Infrastructure and Networking Symposium (GIIS 2013)*, Trento, Italy, October 2013.

[13] Bart Braem et al. A case for research with and on community networks. *ACM SIGCOMM Computer Communication Review*, 43(3):68–73, July 2013.

[14] Community networks testbed for the future internet. `http://confine-project.eu/`.

[15] Roger Baig, Jim Dowling, Pau Escrich, Felix Freitag, Agusti Moll, Leandro Navarro, Ermanno Pietrosemoli, Roger Pueyo, Vladimir Vlassov, Marco Zennaro, and Roc Meseguer. Deploying Clouds in the Guifi Community Network. *14th IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2015)*, May 2015.

[16] The Mobile Broadband Standard. 3gpp specification. `ftp://ftp.3gpp.org/specs/2016-12`, December 2016.

[17] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[18] Ken Pepple. *Deploying OpenStack*. O'Reilly Media, 2011.

[19] Openstack swift's documentation. `http://docs.openstack.org/developer/swift/`.

[20] The National Institute of Standards and Technology. The nist definition of cloud computing. `http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf`, December 2016.

[21] IEEE advancing technology for humanity. Cloud service and deployment model. `http://cloudcomputing.ieee.org/images/files/education/studygroup/Cloud_Service_and_Deployment_Models.pdf`, December 2016.

[22] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, Oct 2009.

[23] World Wide Web Consortium. Webrtc 1.0: Real-time communication between browsers. `https://www.w3.org/TR/webrtc/`, November 2016.

[24] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[25] Alan Shieh, Srikanth Kandula, Albert Greenberg, Changhoon Kim, and Bikas Saha. Sharing the data center network. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 309–322, Berkeley, CA, USA, 2011. USENIX Association.

[26] Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazières, Balaji Prabhakar, Albert Greenberg, and Changhoon Kim. Eyeq: Practical network performance isolation at the edge. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 297–311, Lombard, IL, 2013. USENIX.

[27] Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: Sharing the network in cloud computing. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 22:1–22:6, New York, NY, USA, 2011. ACM.

[28] Q. Duan. Modeling and performance analysis on network virtualization for composite network-cloud service provisioning. In *2011 IEEE World Congress on Services*, pages 548–555, July 2011.

[29] Dijiang Huang et al. Mobile cloud computing. *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter*, 6(10):27–31, 2011.

[30] K A Khan, Wang Qi, C Grecos, Luo Chunbo, Wang Xinheng, Q Wang, C Grecos, C B Luo, and X H Wang. MeshCloud: Integrated cloudlet and wireless mesh network for real-time applications. *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*, pages 317–320, 2013.

[31] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massoulié, Christophe Diot, and Pablo Rodriguez. Greening the internet with nano data centers. *Proceedings of the 5th international conference on Emerging networking experiments and technologies CoNEXT 09*, page 37, 2009.

[32] Amir H. Payberah, Fatemeh Rahimian, Seif Haridi, and Jim Dowling. Sepidar: Incentivized market-based P2P live-streaming on the Gradient overlay network. *Proceedings - 2010 IEEE International Symposium on Multimedia, ISM 2010*, pages 1–8, 2010.

[33] Amir H. Payberah, Jim Dowling, Fatemeh Rahimian, and Seif Haridi. GradienTv: Market-based P2P live media streaming on the gradient overlay. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6115 LNCS:212–225, 2010.

[34] Amir H. Payberah, Jim Dowling, and Seif Haridi. GLive: The Gradient overlay as a market maker for mesh-based P2P live streaming. *Proceedings - 2011 10th International Symposium on Parallel and Distributed Computing, ISPDC 2011*, 00(00):153–162, 2011.

[35] Xiaofei Liao, Hai Jin, Yunhao Liu, Lionel M Ni, and Dafu Deng. Anysee: Peer-to-peer live streaming. In *INFOCOM*, volume 25, pages 1–10. Citeseer, 2006.

[36] S. Asaduzzaman, Ying Qiao, and G. Bochmann. Cliquestream: An efficient and fault-resilient live streaming network on a clustered peer-to-peer overlay. In *Peer-to-Peer Computing , 2008. P2P '08. Eighth International Conference on*, pages 269–278, Sept 2008.

[37] Kunwoo Park, Sangheon Pack, and Ted "Taekyoung" Kwon. An adaptive peer-to-peer live streaming system with incentives for resilience. *Computer Networks*, 54(8):1316–1327, 2010.

[38] L. Kontothanassis, Ramesh Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky. A transport layer for live streaming in a content delivery network. *Proceedings of the IEEE*, 92(9):1408–1419, 2004.

[39] Yu Liu, Hao Yin, Guangxi Zhu, and Xuening Liu. Peer-assisted content delivery network for live streaming: Architecture and practice. *Proceedings of the 2008 IEEE International Conference on Networking, Architecture, and Storage - IEEE NAS 2008*, pages 149–150, 2008.

[40] Thinh Nguyen Kim, Seil Jeon, and Younghan Kim. A CDN-P2P hybrid architecture with content/location awareness for live streaming service networks. *Proceedings of the International Symposium on Consumer Electronics, ISCE*, pages 438–441, 2011.

[41] Zhenyun Zhuang and Chun Guo. Optimizing CDN infrastructure for live streaming with constrained server chaining. *Proceedings - 9th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2011*, pages 183–188, 2011.

[42] V. Singh, A. Abello Lozano, and J. Ott. Performance analysis of receive-side real-time congestion control for webrtc. In *2013 20th International Packet Video Workshop*, pages 1–8, Dec 2013.

[43] L. De Cicco, G. Carlucci, and S. Mascolo. Understanding the dynamic behaviour of the google congestion control for rtcweb. In *2013 20th International Packet Video Workshop*, pages 1–8, Dec 2013.

[44] S. Islam, M. Welzl, D. Hayes, and S. Gjessing. Managing real-time media flows through a flow state exchange. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 112–120, April 2016.

[45] D. T. Nguyen, K. K. Nguyen, S. Khazri, and M. Cheriet. Real-time optimized nfv architecture for internetworking webrtc and ims. In *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pages 81–88, Sept 2016.

[46] Kundan Singh and Venkatesh Krishnaswamy. Building communicating web applications leveraging endpoints and cloud resource service. *IEEE International Conference on Cloud Computing, CLOUD*, pages 486–493, 2013.

[47] Ying Liu, V. Xhagjika, V. Vlassov, and A. Al Shishtawy. Bwman: Bandwidth manager for elastic services in the cloud. In *Parallel and Distributed Processing with Applications (ISPA), 2014 IEEE International Symposium on*, pages 217–224, Aug 2014.

[48] Harold C. Lim, Shivnath Babu, and Jeffrey S. Chase. Automated control for elastic storage. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 1–10, New York, NY, USA, 2010. ACM.

[49] Beth Trushkowsky, Peter Bodík, Armando Fox, Michael J. Franklin, Michael I. Jordan, and David A. Patterson. The scads director: Scaling a distributed storage system under stringent performance requirements. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST'11, pages 12–12, Berkeley, CA, USA, 2011. USENIX Association.

[50] Ahmad Al-Shishtawy and Vladimir Vlassov. Elastman: Autonomic elasticity manager for cloud-based key-value stores. In *Proceedings of the 22Nd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '13, pages 115–116, New York, NY, USA, 2013. ACM.

[51] Bart Braem, Chris Blondia, Christoph Barz, Henning Rogge, Felix Freitag, Leandro Navarro, Joseph Bonicioli, Stavros Papathanasiou, Pau Escrich, Roger Baig Viñas, Aaron L. Kaplan, Axel Neumann, Ivan Vilata i Balaguer, Blaine Tatum, and Malcolm Matson. A case for research with and on community networks. *SIGCOMM Comput. Commun. Rev.*, 43(3):68–73, July 2013.

[52] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[53] Avinash Lakshman and Prashant Malik. Cassandra: A decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 44(2):35–40, April 2010.

[54] Roshan Sumbaly, Jay Kreps, Lei Gao, Alex Feinberg, Chinmay Soman, and Sam Shah. Serving large-scale batch computed data with project voldemort. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST'12, pages 18–18, Berkeley, CA, USA, 2012. USENIX Association.

[55] Openstack cloud software. `http://www.openstack.org/`.

[56] Yi Wang, Arnab Nandi, and Gagan Agrawal. Saga: Array storage as a db with support for structural aggregations. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, SSDBM '14, pages 9:1–9:12, New York, NY, USA, 2014. ACM.

[57] Peter Bodík, Rean Griffith, Charles Sutton, Armando Fox, Michael Jordan, and David Patterson. Statistical machine learning makes automatic control practical for internet datacenters. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[58] IBM Corp. *An architectural blueprint for autonomic computing*. IBM Corp., 2004.

[59] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 143–154, New York, NY, USA, 2010. ACM.

[60] Scaling media storage at wikimedia with swift. `http://blog.wikimedia.org/2012/02/09/scaling-media-storage-at-wikimedia-with-swift/`.

[61] Stephen Hemminger et al. Network emulation with netem. In *Linux Conf Au*, pages 18–23. Citeseer, 2005.

[62] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[63] OpenNebula. Opennebula flexible enterprise cloud made simple. `http://www.opennebula.org/`, February 2014.

[64] Apache CloudStack. Apache cloudstack open source cloud computing. `http://cloudstack.apache.org/`, February 2014.

[65] Ying Liu, V. Xhagjika, V. Vlassov, and A. Al Shishtawy. Bwman: Bandwidth manager for elastic services in the cloud. In *Parallel and Distributed Processing with Applications (ISPA), 2014 IEEE International Symposium on*, pages 217–224, Aug 2014.

[66] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4–1, 2009.

[67] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[68] Atefeh Khosravi, Saurabh Kumar Garg, and Rajkumar Buyya. Energy and carbon-efficient placement of virtual machines in distributed cloud data centers. In *Euro-Par 2013 Parallel Processing*, pages 317–328. Springer, 2013.

[69] Sergiu Nedevschi, Sylvia Ratnasamy, Jitendra Padhye, and Intel Reserch. Hot data centers vs. cool peers. In *HotPower*, 2008.

[70] Parmy Olson. WhatsApp Hits 600 Million Active Users, Founder Says. *Forbes, retrieved October*, 2014.

[71] Cisco Forecast. Cisco Visual Networking Index: Forecast and Methodology, 2014-2019. http://www.cisco.com/c/en/us/solutions/collateral/service\discretionary{-}{}{}provider/ip\discretionary{-}{}{}ngn\discretionary{-}{}{}ip\discretionary{-}{}{}next\discretionary{-}{}{}generation\discretionary{-}{}{}network/white_paper_c11\discretionary{-}{}{}481360.html, December 2014.

[72] Periscope. Live streaming company. http://www.periscope.com/, December 2015.

[73] Skype. Live communications. http://www.skype.com/, December 2015.

[74] M G Saslow. Effects of components of displacement-step stimuli upon latency for saccadic eye movement. *Journal of the Optical Society of America*, 57(8):1024–1029, 1967.

[75] Dana Petcu. Consuming Resources and Services from Multiple Clouds. *Journal of Grid Computing*, 12(2):321–345, January 2014.

[76] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. Interconnected Cloud Computing Environments. *ACM Computing Surveys*, 47(1):1–47, May 2014.

[77] V. Xhagjika, V. Vlassov, M. Molin, and S. Toma. Structured cloud federation for carrier and isp infrastructure. In *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, pages 20–26, Oct 2014.

[78] Network Working Group. Rtp: A transport protocol for real-time applications. https://tools.ietf.org/html/rfc3550, July 2003.

[79] Internet Engineering Task Force (IETF). Rtp topologies. https://tools.ietf.org/html/rfc7667, November 2015.

[80] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 17(9):1103–1120, September 2007.

[81] The Internet Engineering Task Force. Using simulcast in sdp and rtp sessions draft-ietf-mmusic-sdp-simulcast-06. https://tools.ietf.org/html/draft-ietf-mmusic-sdp-simulcast-06, October 2016.

[82] R. Buyya, R. Ranjan, and R. N. Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *2009 International Conference on High Performance Computing Simulation*, pages 1–11, June 2009.