VRIJE UNIVERSITEIT

Coherence in Synchronous Shared Experiences

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op vrijdag 24 juni 2011 om 9.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Ishan Vaishnavi

geboren te Jammu Tawi, India

promotoren: prof.dr. D.C.A. Bulterman
prof.dr.ir. M.R. van Steen

promotiecommissie: prof. K. Nahrstedt
prof. K. Meyer-Patel
prof.dr.ir. H.E. Bal
prof.dr. R.D. van der Mei

Printed by: Ridderprint BV, Ridderkerk, The Netherlands

Contact: ishan.vaishnavi@gmail.com

ISBN 978-90-5335-418-6

To Akshay and Ritwik...

Acknowledgements

The year 2005 was one of great dilemma for me. I stood at the proverbial cross-roads of deciding what I wished to do with life. Long discussions with members of a very innocently named group "sister14", my hostel mates from IIT, followed. It turned out we all stood at the same point. All indecisive 20-something lads trying to figure what the answer to *life, universe and everything else* would be for each of us. As in the Holy Book[1] it was not the answer but the question that we were looking for. The question, as it turned out, was merely figuring out "What do you enjoy doing?" In my case this was research.

It, however, took a lot of time and discussions and practical experiments to figure the question out. For those long discussions over *chai* (and in most cases whiskey), I thank the members of sister14 and in particular, Masti, Khanna, Guru, Chacha, Manki, Bapu and Mulls. My gratitude to Aparna, for just being around and helping me identify my personality. And most importantly, the work I did with Dr. Greg Bollella in Real Time Java was the final nail that sealed the deal for research. Thank you Greg for introducing me to the research world.

Within months of making my decision, I found myself sitting at an office at CWI Amsterdam in presence of a friendly Spanish guy trying hard to make me feel welcome. Little was I to know then that this Spanish guy, more officially called Pablo, was going to form the main back bone of my entire time as a PhD student. Pablo was just always there, irrespective of the time and irrespective of the problem, for the entire time as a PhD student. I attribute more than half of what I now know in terms of research to

[1] Douglas Adams: Hitch Hiker's Guide to the Galaxy

Pablo. He also immensely helped in shaping my character to adapt to the needs of the research environment.

Parallel to Pablo my advisor Prof. Dick Bulterman was *almost* always around. Without his long term vision, direction and advice it would have been impossible to get so far. At various points during the PhD his help was invaluable in improving my ability to communicate ideas, both while presenting and in writing. I am also grateful to Dick for introducing me to most of the other researchers who could help directing me, in particular, Prof. Maarten van Steen, Prof. Klara Nahrstedt and Prof. Ketan M. Patel. Thanks to each one of them for guiding me at various stages of my PhD. I would also like to thank my review committee for their time and effort.

Professionally, the members of CWI's SEN-5 research group are some of the best I have seen. The experience and alertness of Jack both in technology as well as in table tennis and the long discussions with Bo and Kees on just about every topic under the sun kept my brain active on those dull rainy Dutch days. My gratitude goes to people working at CWI as well. Susanne, Nada and Mike for helping me with everything administrative. Aad, Maarten and Henk for their tremendous support in setting up all my experiments.

During thesis years, the support of friends is of utmost importance, especially, their ability to buy you beers when papers get rejected. Rodrigo and I shared the experience of going through our PhDs together over beers, pool, football games, table tennis and Fata Morgana *kip sates* sometimes joined by Pablo. Thanks to them for the beers and to Rodrigo for losing all those pool and table-tennis contests. On a personal front, Anna was there almost throughout my time as a PhD student. With her there was always something to look forward to after work. I got to taste the real Dutch way of life because of her. Without this, Amsterdam would not have been one of my favorite cities and my PhD experience not as pleasant. I personally hold her responsible for all inaccuracies in Dutch in the Nederlandse Samenvatting. Thanks for tolerating all the *kip gung paos* that you were and will be forced to eat.

Without my family I would have been nowhere close to beginning a PhD. I thank them for life in general. First and foremost my father who continues to inspire and motivate me even today. His unending dedication towards me when I was younger is unparalleled. It is under his ideals that I became as open-minded and adaptable to situations as I am today. He was of-course complimented every step of the way by my mother. She would always take my side in my arguments with my father and that helped me a great extent in believing in myself. She was also the other end of the horizon from the otherwise military discipline that my father could sometimes impose. To Akshay and Ritwik I dedicate this thesis. They are and probably will be the dearest people to me throughout my life. A big hug to *Naniji* for taking care of me during my days at IIT and for feeding me all that deliciously cooked food. I wouldn't have

survived outside India without learning to cook *tamatar oulu* from her. Lastly, my sister14 group is nothing short of a *Band of Brothers*. All these people are my identity in life.

I would also like to express my gratitude to the Dutch government, in particular NWO, for funding my work primarily under the Bricks consortium. My heart felt *dankjewels* to the people of Amsterdam who make it such a special city to live in and a special mention for the bicycle thief who made the experience genuine.

Contents

Nederlandse Samenvatting

De afgelopen decennia zijn er applicaties verschenen waardoor het mogelijk wordt om ervaringen met media play-out (*media stream*) met elkaar te delen. Dit kan middels tekst/audio of video communicatie (*communication stream*). Voorbeelden van dergelijke applicaties zijn gedistribueerde online spellen (*distributed games*), Yahoo Zync en recente producten zoals Boxee Box. Dit proefschrift analyseert de eisen die zogeheten *shared experiences* (gedeelde ervaringen) stellen met betrekking tot synchronisatie, alsmede de implicaties die deze eisen hebben voor de inrichting van het toekomstige Internet. In vergelijking met traditioneel onderzoek naar synchronisatie, vereist onderzoek naar de synchronisatie van deze *shared experiences* enkele additionele elementen: (i) ongehinderde communicatie tussen de gebruikers (ii) gedistribueerde media synchronisatie en (iii) synchronisatie voor mobiele gebruikers. Wanneer het gaat over de synchronisatie van *shared experiences* wordt in dit proefschrift gesproken over coherentie (*coherence*). De invloed van elk van deze additionele elementen op de inrichting van het Internet wordt onderzocht, alsmede de invloed van elk van deze additionele elementen op de shared experience applicatie.

Ten eerste, voor ongehinderde communicatie tussen gebruikers zijn kwaliteitsgaranties (ook wel garanties met betrekking tot *quality of service*, of QoS geheten) over het Internet nodig. De inrichting van het huidige Internet is echter gericht op schaalbaarheid en best mogelijke communicatie (*best effort*), en niet op QoS. De uitdaging is dan ook om tijd-gegarandeerde communicatie (*time-bounded delivery*) over het Internet te bieden en tevens haar gerichtheid op best mogelijke communicatie te behouden. Deze twee doeleinden zijn per definitie conflicterend. De huidige aanpak met betrekking tot de zogeheten *Differentiated service network architecture* (Diffserv), lost dit

probleem op middels de creatie van een virtueel *circuit-switched* netwerk voor *real-time* verkeer over het *packet-switched* Internet. Deze oplossing leidt echter tot het verlies van een aantal voordelen van *packet-switching*, wanneer toegepast op real-time applicaties. Specifiek, optimaal bandbreedte gebruik en schaalbaarheid raken verloren. Daarnaast behoort data van real-time applicaties tot het *inelastic* verkeer, dit in tegenstelling tot *elastic* data verkeer gebaseerd op TCP. Dit betekent dat real-time applicaties, over het algemeen, hun bandbreedte behoefte niet zo gemakkelijk aan kunnen passen aan variërende netwerk condities. Hierdoor ontstaat er behoefte aan een mechanisme om van te voren connecties te regelen, voor die gevallen waarin het netwerk fors belast dreigt te raken. Traditionele invalshoeken met betrekking tot connectiebeheer (*connection admitance*), in het bijzonder RSVP, worden niet in staat geacht schaalbaar genoeg te zijn om te kunnen functioneren in het huidige Internet. Het toegangsmechanisme moet namelijk ook passen binnen de aanpak van best mogelijke communicatie van het Internet. Dit proefschrift presenteert *Estimated service* (Estserv), een extensie van de Diffserv architectuur. Estserv brengt de mogelijkheid voor real-time transport over het Internet, middels het deadline-gebaseerd scheduling mechanisme. Experimenten met Estserv, tonen een verbeterd bandbreedte gebruik in vergelijking met Diffserv, terwijl tegelijkertijd de schaalbaarheid behouden blijft. Daarnaast wordt er een best mogelijke connectie toegangsmechanisme gepresenteerd binnen de Estserv architectuur. De werking van dit mechanisme wordt aangetoond door middel van verschillende experimenten.

Ten tweede dient een applicatie voor *shared experiences*, gedistribueerde media synchronisatie te waarborgen, gegeven het bestaan van ongehinderde communicatie. Dit is nodig om in een consistente context van media stromen te voorzien om zodoende de communicatie tussen gebruikers te faciliteren. Exacte synchronisatie is onmogelijk te bereiken. Het is echter wel mogelijk om een systeem te ontwerpen waarin de play-out van mediastromen binnen bepaalde grenzen blijft. De uitdaging hier is om acceptabele grenzen vast te stellen. Met andere woorden, op welk punt begint een gebrek aan synchronisatie in de mediastroom de ervaring van de gebruiker in een *shared experience* te beïnvloeden. De waarde van deze grens hangt van een aantal factoren af, waaronder de applicatie zelf, de gebruikers en het soort mediastroom. In dit proefschrift zijn er gebruikersexperimenten uitgevoerd om deze grens vast te stellen voor een specifieke applicatie. Om een dergelijk experiment uit te kunnen voeren is het echter noodzakelijk dat er een werkend systeem is dat de mediastromen kan synchroniseren. Dit proefschrift analyseert eerder werk met betrekking tot het synchroniseren van gedistribueerde online spellen applicaties. Deze applicaties waren de eerste synchrone *shared experiences* over het Internet. Richtlijnen over hoe dit eerdere werk toegepast kan worden op *shared experiences* in zijn algemeen worden in dit proefschrift besproken. Een voorbeeld implementatie wordt gegeven en de nauwkeurigheid

van de synchronisatie die bereikt wordt binnen deze implementatie is berekend. De implementatie is getest met gebruikers in Amsterdam en Seoul. De bevinding is dat de bereikte synchronisatie grens binnen de door gebruikers acceptabel geachte grens ligt.

Een derde eis voor coherentie is dat de *shared experience* zich synchroon met de gebruiker moet verplaatsen. Synchrone verplaatsing betekent dat de staat van de *shared experience* op de originele locatie consistent dient te zijn met de staat van de *shared experience* op de doellocatie. Dit werk onderscheidt het verschil tussen zogeheten *state-full* en *stateless* stromen. Specifiek, communicatiestromen kunnen gezien worden als stateless (bijvoorbeeld, gesprekken via Skype hebben geen *state*), terwijl media stromen state-full kunnen zijn (bijvoorbeeld, de *state* van een film is de laatst afgespeelde scene). Een traditionele aanpak van verplaatsing van *shared experiences* neemt deze verschillen niet in overweging. Daarnaast zal de traditionele aanpak het verzoek tot verplaatsing van de *shared experience* in individuele media verplaatsing opbreken. Doordat de *shared experience* opgebroken is, moet dezelfde verplaatsingsactie steeds opnieuw ondernomen worden voor iedere afzonderlijke media stroom. Deze aanpak wordt dan ook gekenmerkt door herhaling in de *signalling plane* en is daarmee inefficiënt. Andere nadelen, zoals de onmogelijkheid om de optimale gebruikerservaring te garanderen, worden eveneens besproken in dit proefschrift. Dit werk presenteert een mechanisme dat de status van de state-full media items in de *shared experience* opslaat. Dit gebeurt gezamelijk met de andere (stateless) items. Vervolgens wordt de opgeslagen data getransporteerd en herstart de *shared experience* op de doellocatie. Hierbij wordt de opgeslagen status opnieuw toegepast op de state-full media. Deze manier heeft als voordeel dat de presentatie heronderhandeld wordt op de doellocatie, waardoor een optimale gebruikerservaring geboden kan worden. De opgeslagen data, gezamenlijk met de *shared experience*, wordt nu slechts eenmaal verplaatst, waardoor herhaling in de *signalling plane* voorkomen wordt.

Als laatste, gaat een aantal oplossingen, gepresenteerd in dit proefschrift, uit van de assumptie van variërende niveaus van tijd synchronisatie. Het algemene protocol voor tijd synchronisatie op het Internet is het Network Time Protocol (NTP). Onontkoombaar is de vraag of de assumpties wat betreft tijd synchronisatie, die in dit proefschrift gemaakt zijn, valide zijn. Dit proefschrift analyseert de bestaande NTP netwerken, waarbij een aanpak, genaamd spidering, gebruikt wordt. Spidering is feitelijk een zogeheten *breadth first* zoektechniek, waarbij de actuele status van iedere knoop onderzocht wordt. Zodra het gehele netwerk gescand is, kan een beeld gevormd worden over de algemene status van het NTP netwerk. Onderzoeken, die in dit proefschrift gepresenteerd worden, tonen aan dat meer dan 98 procent van de knopen in het NTP netwerk binnen 128ms van elkaar opereren. Ondanks het feit dat deze waarde acceptabel is voor oplossingen voor gebruikersmobiliteit en media synchronisatie, dient er meer werk gedaan te worden om het vereiste niveau van tijd synchronisatie voor een niet-gemodificeerde

versie van Estserv te realiseren. Dit proefschrift presenteert daarom een oplossing om Estserv in een on-gesynchroniseerde Internet omgeving toe te passen.

Concluderend, dit proefschrift beargumenteert dat het ontwerp van het huidige Internet aangepast dient te worden, zodat synchrone *shared experiences* op een efficiënte manier gefaciliteerd kunnen worden. Meer efficiënte QoS technieken, zoals Estserv, die niet in strijd zijn met de basis principes van Internetcommunicatie, moeten ontwikkeld worden vanuit het huidige Internet. Daarnaast moeten alle ontwerpen van het Internet zich verantwoorden over de manier waarop zij beheerd worden, zowel administratief als economisch. Tijd synchronisatie mechanismen zouden inherent moeten zijn aan het ontwerp van het toekomstige Internet. Hierdoor kunnen technieken, zoals synchrone gebruikersmobiliteit en gedistribueerde media synchronisatie, gefaciliteerd worden. Tot slot, worden er een aantal richtingen voor vervolgonderzoek besproken die logisch voortvloeien uit dit proefschrift.

CHAPTER 1

Introduction

The convergence of synchronous communication, media broadcast and playback mechanisms on the Internet can create innovative forms of *shared experiences*. The preliminary steps towards such innovative shared experiences can already be seen in services, such as Yahoo Zync. In Zync two users situated at different locations can watch a YouTube video together while chatting. The play out of the videos at both ends is synchronized. In this way Zync integrates instant messaging with video playback, enabling users to share their experiences about the video. A generic scenario, which identifies with such shared experiences, consists of the following two features: primarily, the users are connected to each other via an IP based network over which they can synchronously communicate. This communication can be done via applications, such as text messaging, audio or video conferencing or in the future 3D immersion [103]. The data sent over the network by these applications is named *communication stream*. Secondly, the users consume on demand or broadcast media presentations, which can originate from different sources depending on their service providers. Each of these media presentations can be composed of a *primary media stream* and multiple *secondary media streams*. The play out of the primary media streams must be temporally related across the participants, while the secondary streams may or may not have a temporal relation with their respective primary stream. Figure 1.1 presents a network abstracted, application level view of this scenario.

This combination of various communication streams over the Internet, with differ-

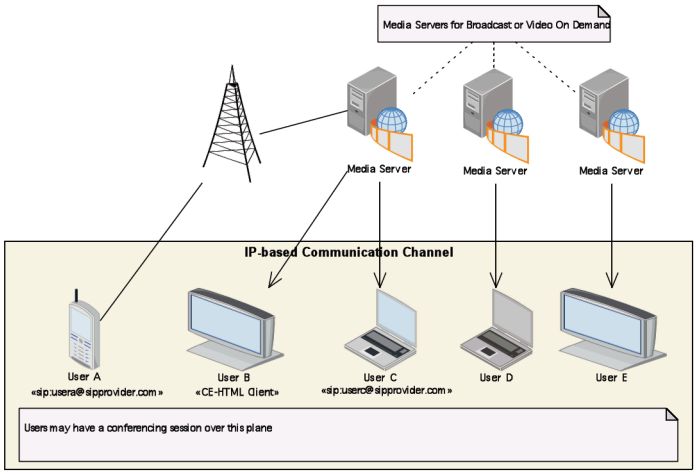


Figure 1.1: Scenario as seen by the application

ent types of on demand or broadcast media presentations can be used to create innovative shared experiences as shown in Figure 1.2. The figure demonstrates how newer shared experience applications can be created by the combination of synchronous communication with media streaming. Thus, using this definition for shared experiences, as demonstrated in Figure 1.2, distributed games become the first shared experiences to have existed over the Internet. However, to realize these yet to appear shared experiences it is important that appropriate synchronization mechanisms exist. Compared to traditional synchronization research, synchronization of shared experiences requires additional elements. This work therefore adopts the term *coherence* when referring to synchronization of these shared experiences.

Traditional media synchronization research refers only to orchestrating the play out of within (*intra-media*) or across (*inter-media*) streams. Synchronization across streams can further be classified on the basis of a number of factors, such as media sources as shown in Figure 1.3. In addition to orchestrating media play out of the primary media streams across users, coherence also refers to the unhindered interaction of users over the communication stream, in addition to the seamless motion of user presentations to other devices or locations. If users cannot interact with each other then there is little use of a distributed synchronization mechanism. It is in fact unhindered user interaction that creates the requirement of distributed media synchronization. Fur-

ther, the ubiquitous connectivity of newer devices at all times to the Internet will mean that these sessions need to seamlessly move and adapt from one device to another and from one network to another as the user moves.

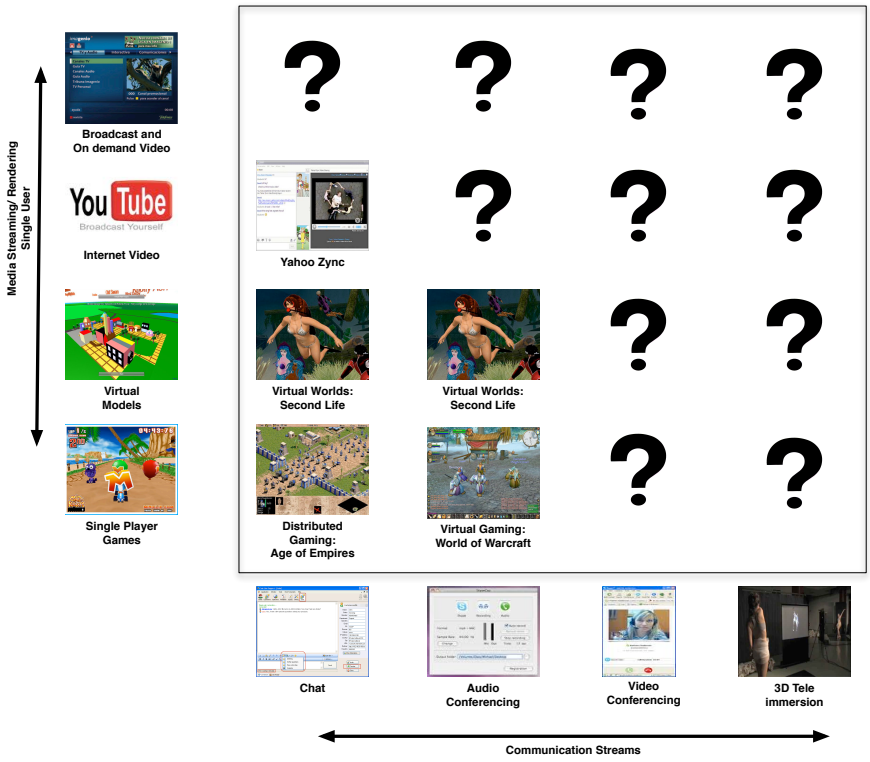


Figure 1.2: Existing and yet to appear shared experiences

1.1 Research Area

The three identified elements of coherence impose a number of requirements on the existing Internet: Firstly, the element of unhindered inter-user interaction places a requirement of time-bounded deliveries of the shared experience stream. Secondly, the element of orchestration of play out of the primary media stream across locations places

a requirement of the existence of a distributed media play-out synchronization on the Internet. Lastly, the element of user mobility requires that when users change their location they must be able to restart their presentations optimized to the new location and in the same state as they left it. It is a widely held opinion that the current Internet cannot support all these demands efficiently. This is demonstrated by various projects that have recently started to appear on designing the new Internet: NetSE[1] is the USA, in Japan AKARI[2], Program Future Internet[3] in Korea and a proposal of future Internet architectures [93] by the European Commission. This leads to the question:

Main Question: *What is the design for the future Internet that can provide efficient technical support to coherence in new shared experiences, while sustaining the scalability and maintainability of the existing network?*

The designs conceived in the projects mentioned above are broadly split into two categories: *revolutionary or clean slate* architectures and *evolutionary* architectures. It is a widely held opinion that no successful approach can be completely clean slate owing predominantly to economic factors. This work, thus, also adopts an evolutionary approach to future Internet design.

In the next section the main question is split into a number of sub questions based on the previously identified requirements on the Internet, namely: time-bounded delivery, distributed media synchronization and user mobility. The questions related to these areas form the main focus of this thesis. Other complementary issues which are not directly addressed in this work are presented in section 1.2.5. The contributions of this work towards these sub questions are highlighted in Section 1.3. This is followed by the organization of this thesis is Section 1.4.

1.2 Research Questions

The previous section enlisted three main requirements that coherence in shared experience would impose on the Future Internet: time-bounded delivery, distributed media synchronization and user mobility. In addition distributed media synchronization will internally require time synchronization. The key research questions emerging from each of the three requirements, further supplemented by questions due to the requirement of time synchronization, are presented in the following subsections.

[1] www.geni.net/netse_about.html
[2] akari-project.nict.go.jp/eng/overview.htm
[3] mmlab.snu.ac.kr/fiw2007/presentations/architecture_tschoi.pdf

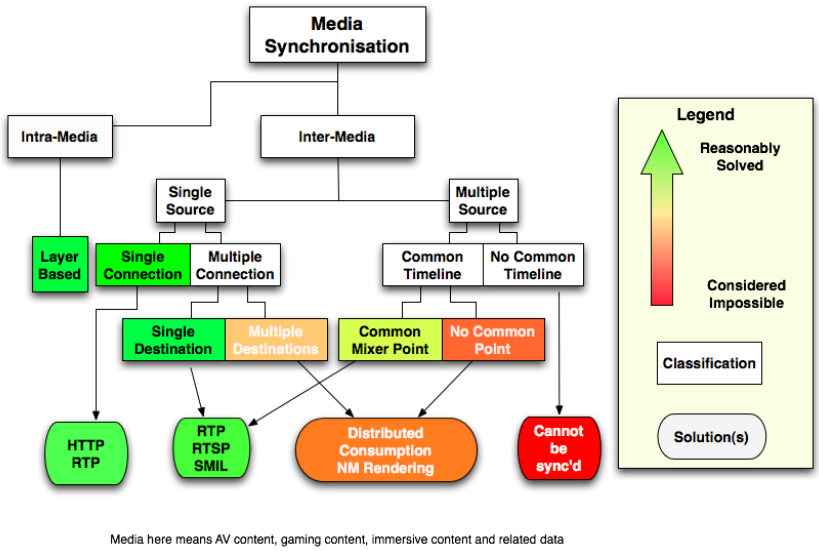


Figure 1.3: Overview of media synchronization research

1.2.1 Time-Bounded Delivery

The communication streams such as conferencing streams require time-bounded delivery, with bounds of the order of $150ms$, for coherence in shared experiences. The future Internet must stay best effort and therefore these bounds cannot be guaranteed. However, the architecture can be made more efficient. Time-bounded delivery, also referred to as real-time delivery, falls in the traditional field of multimedia delivery research: quality of service (QoS). The current state of the art in quality of service architectures, differentiated services [36](Diffserv), provides service class differentiation of packets based on the required quality of service. For time-bounded delivery its *expedited forwarding*(EF) class provides the highest priority over all other service classes, with the assumption of over-provisioning. This thesis looks at the efficiency, in terms of bandwidth utilization, of this architecture and examines how it can be extended to the future Internet. The research questions are:

Research Question 1.1 *How efficient is the Diffserv network towards communication streams, such as video conferencing, in networks that are not overprovisioned?*

This assumption of over-provisioning cannot always be practically ensured. This is especially true across cross-domain links and in the face of bandwidth heavy communication applications, such as video conferencing and 3D immersion [69]. Thus for practical usability the performance of all QoS architectures must be studied in links which are not overprovisioned. In fact the assumption of sufficient and overprovisioned links in the design of Diffserv serves in effect as a deterrent to its acceptance. This is because QoS mechanisms are unimportant to administrators of overprovisioned networks and non-overprovisioned networks violate the assumption. The in between sweet spot is hard to find in practical networks. Further, newer applications such as high definition video conferencing and 3D tele-immersion use *large and varying* amounts of bandwidth. Over provisioning becomes an unreasonable assumption for these large bandwidths. It is therefore important to examine how Diffserv performs over under-provisioned links.

Research Question 1.2 *How can Diffserv-EF be extended to ensure efficient behavior of the future Internet in links that are not overprovisioned while maintaining scalability provided by the Diffsev architecture?*

Efficient behavior here means that if, say, 5Mbps of bandwidth is available to an under-provisioned link then all of that 5Mbps is actually utilized to send data to achieve the best possible quality for the communication stream in the shared experience. The loss of throughput due to congestion control, flow control, reliable delivery and application level bandwidth adaptation mechanisms is well understood [22, 67]. These mechanisms are however, the lifeline keeping the Internet running today and cannot be wished away. The idea is to try to provide a QoS infrastructure within the Diffserv architecture, such that these mechanisms do not result in underutilization of the available bandwidth.

Research Question 1.3 *Is it possible to provide connection admittance control in the future architecture, while still maintaining scalability provided by the Diffsev architecture?*

Efficient utilization, however, is not enough to ensure a good quality of experience to users. For example: 500 video conferencing connections through a 5Mbps link result in 0.01Mbps per-connection. Even though the entire capacity of the link may be utilized, the user experience of the quality of communication stream will not be satisfactory. Prior to Diffserv, Integrated services architecture [8](Intserv) was proposed to ensure end-to-end time-bounded delivery. Due to scalability and economic issues [3]

Intserv has not been widely accepted. The Intserv architecture also provided a connection admittance mechanism, which was based on each router being successfully able to calculate a specific time for the delivery of data packets of the requested connection. Such an architecture cannot be used in the future Internet since it is not best effort. Best effort delivery is considered to be the single most salient feature responsible for the scalability and therefore the success of Internet today. In an evolutionary approach, thus, at no point must a router have to create persistent memory structures or queues on a per-connection basis. Thus an innovative approach is required, which automatically informs the requesting client application that a connection is infeasible.

1.2.2 Distributed Media Synchronization

Media synchronization is important for coherence. Research in this area has so far primarily focused on synchronization of streams to a single client. This includes both intra-media synchronization and certain types of intermedia synchronization. Intra-media synchronization refers to the temporally correct play out of parts of a stream itself, such as the play out of frames of a video. Inter-media synchronization to a single client refers to the correct play out of related streams, such as lip synchronization [57] between audio and video stream of a movie. Distributed synchronization, where media streams across multiple clients need to be orchestrated are less studied, in particular in cases where no assumption on the media sources is made. This is shown in Figure1.3. This thesis examines how newer shared experiences can readapt the synchronization algorithms that have existed in older shared experiences, in particular distributed gaming Figure 1.2. The questions are:

Research Question 1.4 *What levels of distributed play out synchronization does a distributed media system need to achieve?*

Studying distributed media synchronization for future shared experiences as in Figure 1.2 is incomplete, without first knowing the required level of synchronization according to user perception. This question has not previously been studied in research. Currently, as a rule of thumb of 150 - 200ms is assumed. This value emerges from lip synchronization [57] and communication industry research.

Research Question 1.5 *Can event synchronization, as used in gaming, be extended to synchronize user actions and to achieve distributed media play out synchronization in synchronous shared experiences?*

User actions such as “pause”, “play” or “jump to scene” should be executed across all participants in a synchronized manner. Figure 1.2 shows that shared experiences

include distributed gaming. Distributed games over the Internet have existed and been successful for about little over a decade. Just as all future shared experiences, distributed games too require distributed synchronous play out of the game. In the gaming community this is termed *event synchronization*. The concept of event synchronization is that user actions (events) performed at one location of the game must execute simultaneously at all other participating users for the game to be fair. It should then be possible to use the same algorithms, as used in games for event synchronization, to synchronize user actions in future shared experiences. Further, by classifying play-out position update packets as periodic pseudo user events it should also be possible to achieve distributed media play out synchronization using the same algorithms.

1.2.3 User Mobility

Another aspect of providing coherent shared experiences is synchronous user mobility. User mobility refers to the capability of the system to move the user's presentation in a synchronized manner between two locations. Here synchronization means that the users can retrieve their media presentations in the same state as they left them, and adapted to the new context. It is different from session mobility in which the user may stay in the same environment and move only one session of the presentation to another device. Traditionally user mobility is achieved as a composite functionality of the comprising sessions' mobility as in [14], using *SIP-refer* [82]. This is inefficient in the signaling plane since it involves the repeated execution of almost identical session mobility requests. The question then arises:

Research Question 1.6 *Can user mobility be more efficient in the signaling plane, than being considered as a collection of individual session mobility request?*

This SIP based session mobility mechanism is intended for communication streams so that the call need not be dropped. However, for media presentation mobility more coordination between devices is required to transfer state information, which does not exist in conferencing-like real-time applications. Furthermore, this mechanism is inefficient in the control plane as the presentations in the future shared experiences become more complex. Each and every session needs to be renegotiated to the new user context and transformed accordingly. Thus, a new mechanism for user mobility is required.

1.2.4 Time Synchronization

Distributed media synchronization mechanism assumes time synchronization. The question then arises:

Research Question 1.7 *How accurate are the clocks synchronized using NTP?*

As mentioned above time-bounded delivery can never be guaranteed in best effort future networks. In such cases validation of play out position update packets requires that the different participating nodes speak of the same time-stamps. It is then important to examine the efficiency of the existing solution for time synchronization in the Internet, Network Time Protocol [60]. Previous similar surveys, conducted about two decades ago on the performance of NTP [61], [68] found that over 95% of the nodes in the NTP network are within $128ms$ of each other. The surveys also found that level 1 servers are heavily loaded with in some cases up to 30000 computers connecting to a single server. The aim of this survey is to update the time synchronization accuracy results for the current network. Further, a number of existing devices may not have the capability to run NTP, we also examine what other alternatives exist in such cases.

1.2.5 Other Issues

This section presents other related issues of future shared experiences, which are, for various reasons, not handled in this thesis.

Reliable Delivery of the primary and secondary media streams is required for the shared experiences to be meaningful. Transmission control protocol, TCP, is the mainstay of the Internet since its inception. TCP provides best effort, yet reliable, end-to-end transfer of data. TCP is well studied in literature, has stable implementations in all operating systems and provides reliable delivery. Thus, TCP will continue to be a part of evolutionary future Internet architectures. While this reliable delivery is important it is not critical to provide coherence to shared experiences. Instead, distributed synchronization mechanisms have to account for buffer underflow, overflow or loss in delivery conditions of the primary and secondary streams.

Causality is the knowledge of the correct chronological order of actions. It is required by shared experiences to maintain correct ordering over the distributed events and is very important for coherence. In a distributed system causality is trivial to achieve if absolute time synchronization can be achieved. However, due to the phenomenon of time dilation we know that absolute time synchronization is a theoretical impossibility. This of course implies that absolute causality cannot be achieved.

A good approximation is however possible. Each physical system is limited within certain discrete blocks of time within which the correct order of events is imperceptible or unimportant, including the human brain. Mathematically, if events P_1 and P_2 happen in between time t_1 and $t_1 + \delta$ then, for each perceptual system there exists a ψ, such that $\forall \delta < \psi$, the correct chronological ordering over P_1 and P_2 cannot be seen by the system. Thus, if P_1 and P_2 happen within ψ we can say that P_1 and P_2 happened at the same time. If a distributed system can be time synchronized to within $\gamma << \psi$ then an approximate order over these events can be achieved by assuming all events that

happen within ψs of each other to have happened together. This principle is behind the design of time triggered architectures [53] where all nodes in a distributed system will send out a status report/ position update every ψs intervals at the same time. The values of the system in between these updates is calculated using dead-reckoning and conflicts need to be resolved using the application logic at higher layers. The assumption is that the systems are synchronized to within $\gamma << \psi$. All events that have occurred between consecutive events are said to have occurred at the same time. Causality is thus well understood in literature and the principle of time-triggered architectures is used in systems from air traffic radars to embedded system design and thus not addressed by this work.

Synchronous relations of secondary streams with the primary stream needs to be expressed and executed accordingly. These synchronization relations can be expressed in descriptive languages, such as SMIL[4] or NCL [86] and executed using compatible players, such as Ambulant[5]. These relationships affect the coherence of a single user more than the overall coherence of the shared experience, and thus are considered out of the scope of this thesis.

1.3 Contributions

In the last section we identified the seven main questions this thesis addresses. Questions 1.1, 1.2, 1.3 relate to time-bounded delivery of the communication stream and are handled in Chapter 3. To answer Question 1.1 this thesis in Chapter 3 examines the bandwidth efficiency of real-time transmission in Diffserv in experiments over a replicated edge router setup, using a video conferencing application. The experiments show that the real-time links using Diffserv can underutilize available provisioned bandwidth. The future Internet must then ensure efficient behavior of the network even when not overprovisioned. Which in essence is Question 1.2. Chapter 3 further presents Estimated Service, an adaption of the deadline-based scheduling mechanism to multi-hop networks that extends the differentiated networks framework. Using this deadline-based scheduling mechanism we demonstrate an improved bandwidth utilization, while maintaining end-to-end delays, over under-provisioned Diffserv Expedited Forwarding (EF) [19] links in an Estserv network over the traditional Diffserv architecture. The experiments are performed using traffic performance generation and measurement tools as well as the video conferencing applications over a controlled network. This work is then extended to the current Internet with similar results. Lastly, regarding Question 1.3, a best effort mechanism for connection admittance using the

[4] http://www.w3.org/AudioVideo
[5] http://www.ambulantplayer.org/

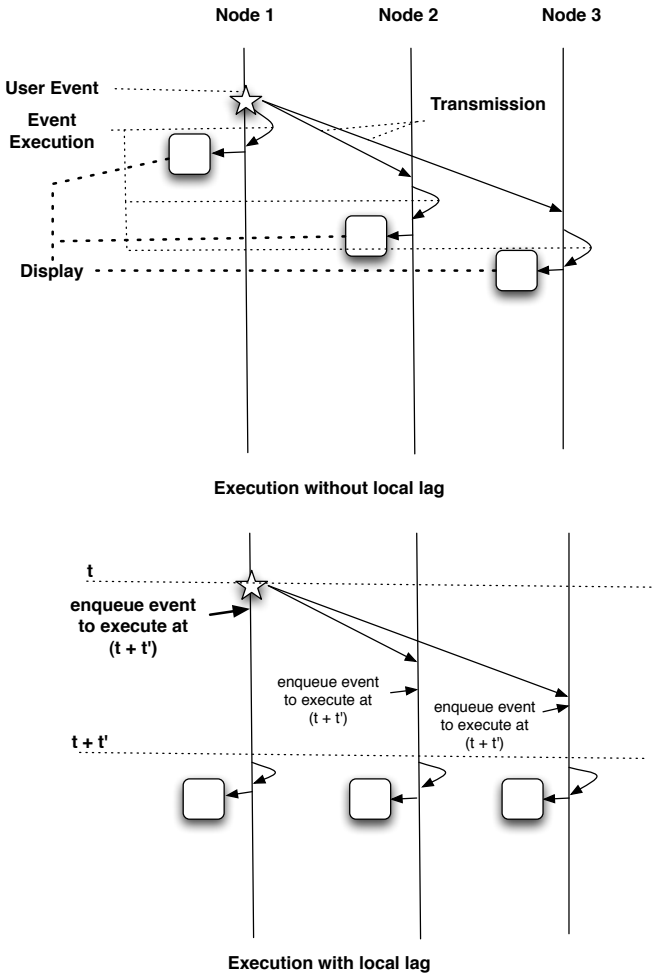


Figure 1.4: Figure demonstrating the concept of local lag algorithm

estimation mechanism is presented. The work presented in Chapter 3 consists of extracts from the following publications:

- Ishan Vaishnavi, P.S. Cesar, Dick C. A. Bulterman, Oliver Friedrich, *From IPTV*

Services to Shared Experiences: Challenges in Architecture Design. Proceedings of the IEEE Conference on Multimedia and Expo, in IWITMA 2010, Singapore.

- Ishan Vaishnavi, Dick C.A. Bulterman. *Estimate and serve: scheduling soft real-time packets for delay sensitive media applications on the Internet.* June 2009, NOSSDAV '09: Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video.

The next two questions 1.4, 1.5 refer to distributed media play out and are handled in Chapter 4. Responding to Question 1.4 requires that some implementation of a synchronization algorithm exists, which in essence is Question 1.5. Thus in Chapter 4 a temporary and static mechanism for achieving media synchronization in a given environment is described. This system is validated to be accurate within 150ms. Thus using this system with regards to Question 1.4, Chapter 4 reports on results of user tests conducted to determine the extent of user tolerance to out of sync videos while audio conferencing or text chatting with each other using this implementation. With regards to Question 1.5 the problem with implementing event synchronization is that in a distributed system events will always take a finite time to travel from one location to another, and therefore cannot be perfectly simultaneous. During this elapsed time various actions may occur at the other end, which may conflict with the original events. For distributed games this problem has been addressed by a technique called local lag and time warp [59]. Figure 1.4 demonstrates this technique. When the user performs an action at one end, it is not executed simultaneously at his end. Instead each event is associated with a global execution time stamp in the future $t_{now} + t_{est}$, where t_{est} is the worst-case network delay estimate to other participating nodes. This technique tries to ensure that almost all nodes receive the event before being executed. The same concept can be extended to other forms of user interaction with distributed media. In particular the work demonstrates how in distributed media presentations user actions such as "pause", "play" or "jump to scene" can be executed across all participants in a synchronized manner. Further, by classifying play-out position update packets as periodic events, Chapter 4 shows how distributed media synchronization can be achieved using this event synchronization infrastructure. An implementation of this synchronization algorithm is presented. Chapter 4 contains extracts from the following documents

- I. Vaishnavi, , P. Cesar, O. Friedrich, S. Gunkel and D. Geerts. *From IPTV to Synchronous Shared Experiences: Challenges in Design: Distributed Media Synchronization.* Accepted for publication in Elsevier Journal of Signal Processing: Image Communication, 2011.

- D. Geerts, I. Vaishnavi, R. Merkuria, P. Cesar, and O. van Deventer. *Are we*

in Sync? Synchronization Requirements for Watching Online Video Together. Accepted for publication at CHI 2011, Vancouver.

Work done in Chapter 4 also contributed to IETF draft:

- H. Stokking, M. van Deventer, O. Niamut, F. Walraven, R. van Brandenburg, I. Vaishnavi, F. Boronat, M. Montagud. *RTCP-XR block type for inter-destination media synchronization.* IETF Draft, 2011

Both Estimated service and the synchronization algorithm assume clock synchronization. As mentioned in Section 1.2.5, to align with the principles of time triggered architectures [53] the upper bounds on the errors in time synchronization, γ, must be much less than the required perception, ψ. We know that for time-bounded delivery the value of ψ is $150ms$ [46], while for distributed media synchronization ψ is measured by the user tests. The accuracy of time synchronization must be guaranteed to be within $20ms$. This raises Question 1.7 which is handled in Chapter 6. This thesis in Chapter 6 surveys the health of today's NTP network similar to the surveys [61], [63]. These surveys were performed about twenty years ago. Since then a lot has changed in the network and even with NTP. Thus a more recent survey is beneficial. The results of our survey demonstrate that 60% of the nodes in the NTP network are within $20ms$ of each other. This implies that the Internet of today can be synchronized to a reasonably high level of accuracy provided the nodes are properly administered.

Lastly with regards to Question 1.6 in Chapter 5, recognizing the one to one association of presentations with users leads us to propose a *presentation layer mechanisms* for user mobility. The idea is to store user's current presentation state in a descriptive language, such as SMIL, on the service provider's server when the user indicates he wants to move from one location to another. At the new location this session is retrieved, and various optimization decisions on device, network, user context are performed. Then the last saved state is retrieved from the stored file and applied to this new connection, thereby restoring the old session. In this way the presentation is already optimized to the new context while preserving synchronous user mobility.

This method helps in re-negotiating all the QoS and synchronization requirements presented above at the new location independent of the previous location, which could not be done with session mobility. This improves performance, does not require additional implementation overhead and provides better user experience. The chapter is an overview of the following publications:

- I. Vaishnavi, P. S. Cesar, A. J. Jansen, B. Gao, Dick C. A. Bulterman. *A presentation layer mechanism for multimedia playback mobility in service oriented architectures.* December 2008, MUM '08: Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia

- Pablo Cesar, Ishan Vaishnavi, Ralf Kernchen, Stefan Meissner, Cristian Hesselman, Matthieu Boussard, Antonietta Spedalieri, Dick C.A. Bulterman, Bo Gao.*Multimedia adaptation in ubiquitous environments: benefits of structured multimedia documents.* September 2008, DocEng '08: Proceeding of the eighth ACM symposium on Document engineering

- R. Kernchen, K. Moessner, P. Cesar, I. Vaishnavi, S. Meissner, M. Boussard, C. Hesselman, *Intelligent Multimedia Presentation Delivery in Ubiquitous Multi-Device Scenarios.* IEEE MultiMedia (IEEE MM), 17(2), April-June, 2010.

There are three other supporting contributions required for the completeness of this thesis. Firstly, in Chapter 2 the thesis explains a generic architecture for shared experiences in detail. Second, as a supporting contribution to Chapter 3, an efficient bandwidth measurement methodology is reported in Appendix A. Third, as a supporting contribution to Chapter 4, a distributed time synchronization algorithm is presented in Appendix B. This algorithm, called neighbourCast, handles cross-domain time synchronization cases for shared experiences where one or more of the user is unable to access an NTP server. These peripheral issues are also presented in the following publications:

- Ishan Vaishnavi, P.S. Cesar, Dick C. A. Bulterman, Oliver Friedrich, *From IPTV Services to Shared Experiences: Challenges in Architecture Design.* Accepted for Publication in IWITMA 2010, Singapore.

- Ishan Vaishnavi, Ahsan Arefin, Dick C. A. Bulterman, Klara Nahrstedt, Raoul Rivas, *Eureka: A Methodology for Measuring bandwidth usage of networked games, environments and applications.* Proceedings of International Conference on Multimedia and Expo 2010, Singapore.

- Ishan Vaishnavi, Dick Bulterman, Pablo Cesar, Bo Gao. *Media Presentation synchronization for Non-monolithic Rendering Architectures.* December 2007, ISMW '07: Proceedings of the Ninth IEEE International Symposium on Multimedia Workshops.

In conclusion, this work argues that the design of the existing Internet has to evolve further to facilitate the acceptance of synchronous shared experiences in an efficient manner. More efficient QoS techniques, such as Estimated service presented in this thesis, that do not violate the best-effort principles should be evaluated and accordingly deployed on the existing Internet. Further, all designs for the Internet must account for the way it is managed, both, administratively and economically. Time synchronization mechanisms should be inherent to the design of the future Internet to facilitate synchronous user mobility and distributed media synchronization mechanisms.

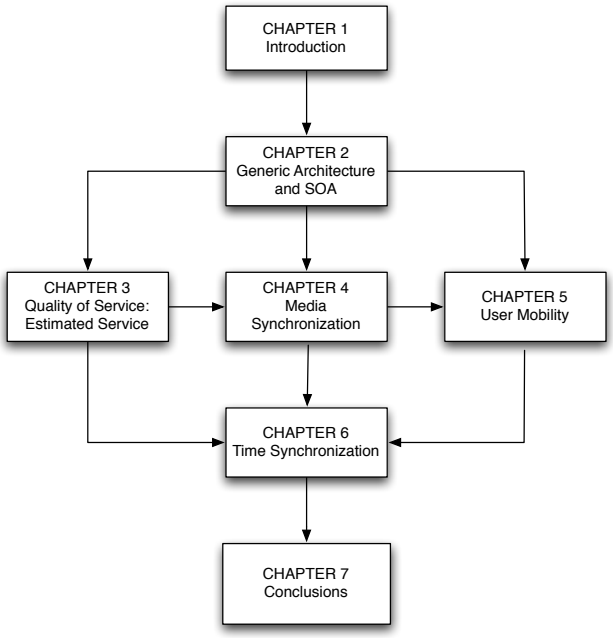


Figure 1.5: Organization of the thesis

1.4 Organization

Figure 1.5 shows the overview of the organization of the chapters. The next Chapter presents the generic architecture of synchronous shared experiences. Therein high level components are identified and their responsibilities, in terms of the questions presented in this chapter, are specified. We identify four main topics: quality of service, distributed media synchronization, user mobility and time synchronization. As can be seen in Figure 1.5, Chapters 3 through 6 deal precisely with these with these four topics, individually. Each of these four chapters answers the respective questions assigned to it. A brief summary section is presented at the end of every chapter to highlight its contributions, with regards to the questions identified in this chapter. Chapter 7 summarizes all the results of this work, presents future opportunities in the area of synchronous shared experiences, and finally presents some personal concluding remarks.

CHAPTER 2

Generic Architecture and State of the Art[1]

This chapter presents a conceived generic architecture for synchronous shared experiences used in this thesis. A synchronous shared experience for the purposes of this work is defined as a media experience shared between geographically distributed users communicating with each other synchronously. Application and network level views for synchronous shared experiences are represented in Figure 2.1. The figure goes a step further than the integration of the user's home network to the public Internet as presented in [66]. This chapter defines an architecture that is used throughout this the-

[1] Part of the work done in this chapter is published in

- I. Vaishnavi, P. Cesar, D. C. A. Bulterman, and O. Friedrich. *From iptv services to shared experiences: Challenges in architecture design*. Proceedings of IEEE Conference on Multimedia and Expo, 2010.
- R. Kernchen, K. Moessner, P. Cesar, I. Vaishnavi, S. Meissner, M. Boussard, C. Hesselman, *Intelligent Multimedia Presentation Delivery in Ubiquitous multiDevice Scenarios.* IEEE Journal of MultiMedia, 17(2), April-June, 2010.
- C.Hesselman, Daniele Abbadessa, Wouter van der Beek, Daniel Gorgen, Keir Shepherd, Sander Smit, Mark Gulbahar, Ishan Vaishnavi, Josip Zoric, Dietwig Lowet, Robert de Groote, John O Connell, Oliver Friedrich. *Sharing Enriched Multimedia Experiences across Heterogeneous Network Infrastructures.* Published in IEEE Communications Magazine, Volume 48 Issue 6, June 2010 .

sis to help achieving the requirements for *coherent* synchronous shared experiences, namely: an efficient quality of service over the communication stream, distributed synchronization of the media experience, the capability of users to move from one device to another, and time synchronization. Components within this architecture are identified with various responsibilities towards achieving coherence. To further clarify these responsibilities, this chapter reviews the current state of the art in each of the requirement areas. It identifies relevant shortcomings that lead to the questions of this thesis.

The challenges to achieve *coherence* can be highlighted using Figure 2.1. The figure firstly draws attention to the communication streams, which travel through various domains across the Internet and require meeting the 150ms end-to-end deadline. Therefore these domains must agree on common QoS semantics such as those provided by the differentiated service architecture [36]. Further, even though the service and content providers for the TV service may be different than the one for mobile service, they still need to achieve distributed media synchronization over the relevant media streams. Lastly, to give the user a feeling of continuity, when the user leaves one domain and joins the other his presentation must move between these domains in a synchronous manner. An example of such a move can be from a cable network to a 3G domain. Lastly, a time-synchronized Internet is important for these operations.

In this chapter our client architecture is presented in Section 2.1 that helps meet these challenges. The section explains the functionality of the various important components of our architecture. The relevant portions of this architecture will be re-referenced later throughout this thesis. The Section 2.2 through 2.2 then identify the state of the art of the various requirements for coherence and how they influence the architecture. Section 2.6 then presents a brief summary of the chapter.

2.1 Generic Architecture

Our generic architecture for shared experiences is shown in Figure 2.2. It includes a number of servers (session, presence, and media servers). Just two cross-domain clients are shown for clarity. A number of needed blocks have deliberately been skipped (e.g., conditional access) in order to better scope our contribution. Each client includes a number of major components: the parent renderer, the session manager, the media (primary/secondary) content renderer, the shared experience renderer and the synchronization agent. Two of the software enablers, the synchronization agent and the parent renderer, must be implemented in-house. The rest are plug-able and can be provided by the appropriate (other) service providers, as far as they conform to the interface required by the parent renderer and the synch agent. As part of this work one such

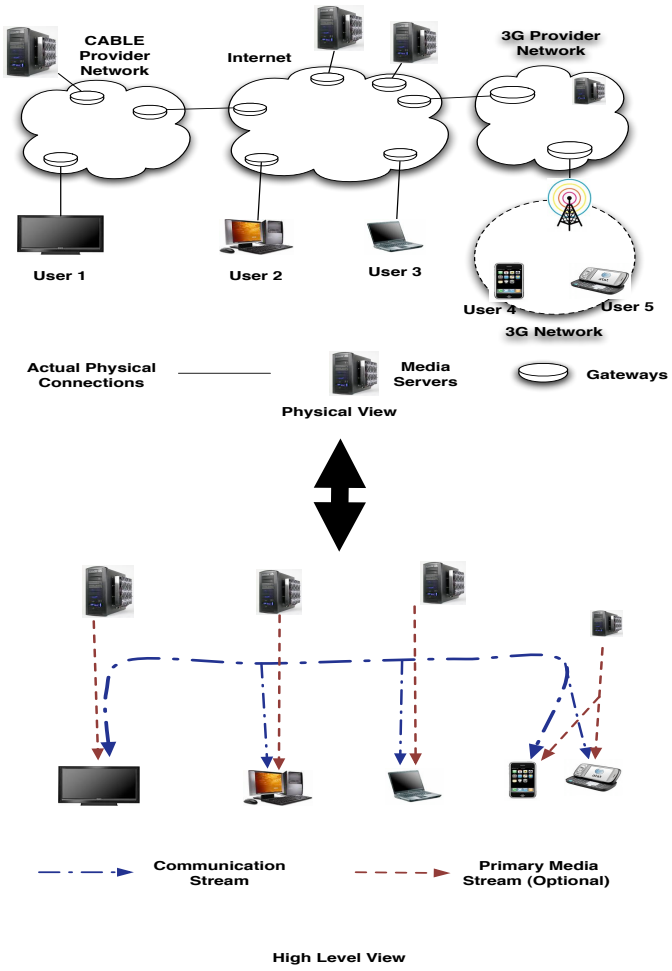


Figure 2.1: Network and application level views

interface was defined within the iNEM4u project[2]. Each of these software components has a certain set of responsibilities as discussed in the following paragraphs.

[2]www.inem4u.eu

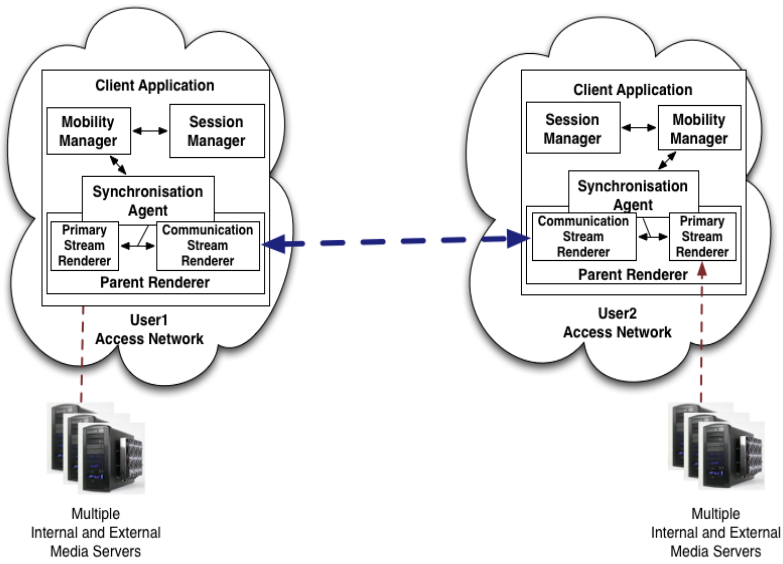


Figure 2.2: A generic architecture for shared synchronous experiences

The parent renderer, in Figure 2.2, is the overall composite renderer that manages and provides user interface/interaction. It is, firstly, responsible for the final dynamic layout of all the media elements and communication streams that compose the shared experience. This may depend on a number of factors, such as screen size, and resolution depth. The parent renderer is also responsible for automatically handling media priorities. For example, prioritizing an incoming call over an existing video playback. The parent renderer may use initiatives in *structured content modeling* such as SMIL[3] and MPEG-4 [73] to express these complex intermedia relationships. Structured content modeling provides a high-level set of constructs for describing the temporal and spatial relationship of the elements, allowing for efficient content control and adaptation mechanisms. A recent further extension provided in [49] was the concept of a state of presentation. While this is not directly relevant to the work done in this thesis, it is nonetheless utilized later in Chapter 5 for synchronized user mobility.

The session manager's role is more than that just of a presence client. Besides presence, it maintains the other media content and shared session details. It must also be

[3]http://www.w3.org/AudioVideo

cross-domain identifiable, i.e. accessible throughout a generic API providing interfaces for multiple protocols as SIP. It also acts as mediator between users from different network domains (e.g. managed Telco IPTV and Over-The-Top Portals) and allows them to share contents transparently of their access network or use various video-streaming standards. Sessions for shared experiences extend the basic multimedia session concepts towards a multiuser, multicontent approach. One such solution is the iSession [34], which provides a logical representation of interactive multimedia sessions that span across multiple domains. An iSession contains all the information that is required by a client to connect to an existing session and to replay an archived session. It includes information on the services and the content that are consumed and on the users who participate in the session. It also contains metadata information that may be required for synchronization of media streams across domains and layout information to create a common experience at the parent renderer across devices. While the session description is domain-independent, clients use domain-specific technology to establish connections to the content sources that are described in the session description.

The media content renderer is a plug-able component such as an IPTV service client, an RTSP client or Internet Video client. It is responsible for handling the primary media stream in accordance with the synchronization agent's control. The shared experience renderer is responsible for rendering the shared session stream and is a plug-able component provided by the shared session service, such as, the Skype client for video/audio conferencing.

The synchronization agent is responsible for a distributed synchronous play-out of the media stream in accordance with the communication stream. This may involve maintaining time synchronization, synchronized play-out of the media content across the network and provide synchronous user mobility in cooperation with the session manager. It is the component responsible for *coherence* of the shared experience.

Further, each component is responsible for maintaining the QoS semantics of respective data. For example, the synchronization agent must mark the synchronization position updates presented later in Chapter 4 as control, while the shared experience renderer must mark its data packets as requiring real-time transmission, and the media renderer must mark its data for reliable delivery.

Coherence over this architecture requires: (1) Efficiency while providing QoS to the communication channel, the media play-out and the control messages, (2) Synchronous play out of the shared multimedia content and (3) restoration of user content in case of user or device mobility. The following sections study each of these requirements and how they relate to the architecture in more detail. The shortcomings in the relevant state of the art are presented to highlight the questions raised in Chapter 1. This thesis will address the questions raised in each one of these requirements in their respective chapters, culminating in an overall discussion in Chapter 7.

2.2 Multimedia Quality

Research in media quality has broadly been divided into two distinct and partly exclusive fields: *network based quality of service (QoS) mechanisms* and *bandwidth adaptation mechanisms*. While network QoS research has focused on providing end-to-end service guarantees, bandwidth adaptation mechanisms have focused on achieving the best possible performance under the given network circumstances. This section focuses on the work done in the network research. We argue that one of the major reasons in the lack of wide scale acceptance of network based QoS solutions stem from the fact that interoperability between bandwidth adaptation mechanisms and network mechanisms. This is because each treats the other as a black box and their interoperation has not been taken into consideration in either of their designs.

2.2.1 Quality of Service

The QoS guarantees required by our architecture can be split into three different categories. The QoS guarantees required by the (1) communication stream, (2) media content and the (3) control flow including synchronization control messages presented later in Chapter 4. Each has different requirements.

For *communication streams*, the QoS guarantees depend significantly on the shared experience itself. While conferencing-like applications require immediate time-bounded delivery, chat-like applications require reliable causal delivery. In this thesis we focus on conferencing-like applications, which include audio conferencing, video conferencing and 3D tele-immersion [69]. The idea of conferencing-like distributed shared experiences is to be able to feel like you are in the same physical space. This means that every action that a user makes at one end must appear immediate to the other users. From a network designer's perspective this implies immediate delivery of actions. However, how fast is immediate? A value of 150ms is used in communication in the telecom industry and is considered appropriate here [46]. These values have been verified by user studies such as [21].

In the early 90s the integrated services architecture was designed to achieve end-to-end delay predictability. Intserv [8] aims to achieve hard RT characteristics over the Internet traffic by restricting excess traffic and policing the accepted traffic. Intserv has not been widely deployed due to economic, maintenance and scalability issues [3]. An understanding of Intserv concepts, in particular connection admittance, is important for the purposes of this thesis. Intserv recommended the resource reservation protocol, (RSVP) [99] to reserve resources along an optimal path before the connection could begin. If it is not possible to reserve these required resources along the complete path then the connection is said to have been rejected by the network. Thus each new connection is required to pass through a connection admittance round before it was allowed

to transmit data onto the network.

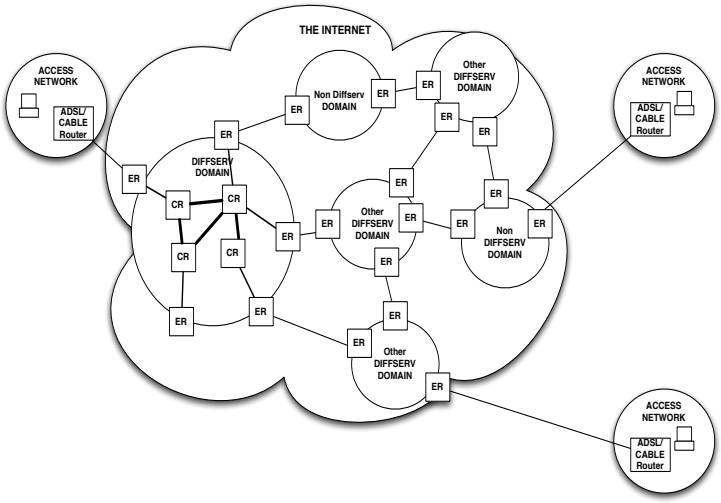


Figure 2.3: Internet architecture: ER = Edge Router, CR = Core Router

The current standard for providing these QoS guarantees is the Differentiated Service networks architecture [36]. Differentiated networks work by classifying packets into behavior classes, known as *per hop behaviors*. The classification is based both on business as well as technical models. From a business perspective, this classification helps ISPs charge users (or other ISPs) based on the type of service they require while on the technical side it splits the networks into distinct unrelated parts that are easier to manage. This is explained in what follows.

The current Internet is split into domains show in Figure 2.3. Typically each of these domains is maintained by a company or a public institution and is connected to other domains or access networks via edge routers (ER). The wireless/ADSL network at our homes is an access network. This classification, while making maintenance easier, also creates a number of business and technical issues. From a business viewpoint, domains that are connected to each other, need to negotiate service level agreements (SLA). Thus, for example, a user, say Caroline, has a service level agreement with an ISP to provide (say a) 5 Mbps connection to her home. The system administrators at the ISP will configure the edge router in their domain to police the incoming traffic to ensure Caroline does not use more than 5Mbps. Further, the ISP will negotiate a SLA

with other domains they are connected to, based on estimates on aggregates from other existing or expected users. These estimates may of course be incorrect or financially infeasible for the ISP. Thus, even if the networks continue to become faster at the core router (CR Figure 2.3) many business driven bottlenecks will continue to exist at edge routers (ER). In general, it is these ERs which determine the end-to-end bandwidth availability for users.

In our architecture there are other characteristics of data besides bandwidth, in particular, delay and reliability of delivery. The current standard to provide appropriate types of service and priority to appropriate types of traffic is the Differentiated service (Diffserv) architecture. Diffserv recommends a number of *per hop behaviors* (PHBs) based on the required type of service. Each ISP may define its own PHBs within its Internet domain. Based on the bandwidth-PHB combination(s) in the SLA, the price between domains is negotiated. The edge routers do this re-mapping of PHBs when a packet crosses domains. The PHBs are marked in the IP header's *type of service (TOS)* field. Diffserv related IETF standards define a number of standard PHBs: expedited forwarding (EF [19]) for real-time delivery and assured forwarding (AF [18]). Af further consists of various subclasses. This makes it easier for various domains to come to a universal mapping from a PHB in one domain to another, so that the end users may get consistent service. Packets within the network are prioritized based on these PHBs, EF having strict priority. In practice, however, EF class cannot be given absolute priority since users today are more tolerant to errors in their real-time Skype conferencing sessions as compared to their best effort e-mail service not working. Thus most implementations use a sort of weighted fair queuing between the various per-hop behaviors [64]. This implies that the entire channel outgoing bandwidth is not available to the EF channel at all times and certain packets may need to be queued.

However, to ensure end-to-end predictability, the Diffserv-EF standard recommends short queue lengths for the EF class. This is done under the assumption of correct provisioning. There are two problems with this assumption in practice: a) it is difficult to correctly provision networks and b) users, especially in multimedia, do not always transmit at a constant bandwidth. Thus for practical applications, analysis of how the architecture behaves under non-overprovisioned conditions is also important. We formulated this as Question 1.1. As a reply to this question, Chapter 3 shows that Diffserv-EF in non-overprovisioned networks leads to inefficient utilization of available bandwidth, leading to Question 1.2. Chapter 3 proposes Estimated service networks which utilizes the time bounds on the data it delivers to realize a more efficient bandwidth utilization. Lastly, as already pointed out in Chapter 1, efficient bandwidth utilization is not enough, connection admittance, addressed as Question 1.3, is also required to maintain a minimum quality level of the communication streams.

Connection admittance is an important aspect for real-time multimedia communi-

cation. Unlike downloading a file, media-based communication is worthless if a certain lower limit of bandwidth availability cannot be met. Thus scalability and fairness of a connection alone are not enough for real-time communication. A minimum bandwidth availability "guarantee" for existing connections is also required. Chapter 3 proposes a scalable method to achieve connection admittance which is somewhat similar to RSVP [99], but does not require any router to maintain any connection-based data structures.

The *media content* itself requires reliable as well as time-bounded delivery. The bounds in time here are larger than those of a conferencing stream and depend on other factors, such as, buffering. Diffserv architectures provide a reliable delivery group (*Assured Forwarding Class*) with subclasses with various parameters, which include low delay. This is suitable for the low delay assured forwarding the primary media stream requires.

Finally, control flow semantics are required to assure prioritized delivery of *control packets* such as the synchronization packets. These semantics can also be achieved using Diffserv based classification by giving control flow priority over all other classes. The control flow class needs to be overprovisioned and since this bandwidth requirement for control packets is limited. The real-time class expedited forwarding comes next and is used for the shared experience traffic. Lastly, the media content can be sent via the assured forwarding with low delay class.

There is a considerable body of concern on the acceptance of QoS solutions in practice [3], including Diffserv. A number of papers argue that it is due to economic factors [9, 17], while some others cite technical scalability and maintenance issues and even security [83] for ISPs. However, in recent years QoS, especially Diffserv-EF is beginning to be accepted: routers such as Cisco NBAR and solutions such as Cisco AutoQOS [15] can prioritize and mark packets from VoIP solutions as EF. Diffserv is also used in smaller networks and specific VoIP solutions [17]. Thus in the design of future networks Diffserv will play an important role.

2.2.2 Bandwidth Adaptation

Bandwidth adaptation is required in our architecture for the communication streams to adapt to existing network conditions. Work in bandwidth adaptation can be broadly classified based on the level in the OSI stack it belongs to: application layer or transport layer. Application layer mechanisms have the advantage of knowing semantic details of the application absent at the transport layer. Here we look at some important work that will be referred to during the course of this thesis.

Transport layer bandwidth mechanisms mostly evolve around congestion control [1]. Congestion control mechanisms are a central feature to the stability of the Internet. While reliable delivery was the main focus of the initial TCP design, congestion control

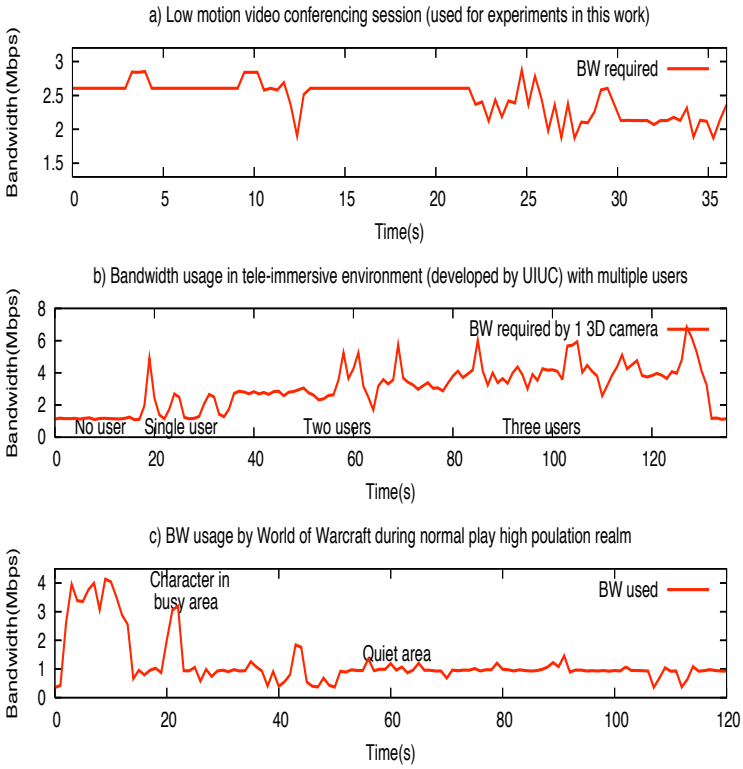


Figure 2.4: Bandwidth requirements for various real-time multimedia applications.

was retrofitted in 1988 [97]. This was appropriate as TCP flows were the major share of the Internet traffic. In recent years UDP traffic due to applications, such as IP telephony and gaming, is beginning to claim more and more share of the bandwidth.

Newer protocols have been designed and implemented to fit UDP with congestion

control, to improve performance for traffic classes, which prefer timeliness to reliability (real-time traffic). A prime example is the Datagram Congestion Control Protocol (DCCP) [22]. However, DCCP has changed very little in terms of congestion control itself. The same TCP models Additive Increase Multiplicative Decrease (AIMD) and TCP Friendly Rate Control (TFRC) have been re-adopted as CCID2 and CCID3, respectively. As the authors of DCCP in [22] themselves acknowledge, these congestion control models were originally designed for best effort traffic and have issues with respect to real-time packet delivery. These issues include inability to handle changes in bandwidth requirements due to codec artifacts, inability to start up rapidly after idle periods, and inability to utilize the best available bandwidth even though it may already be provisioned.

Furthermore, the forthcoming class of applications such as, video-conferencing and tele-immersion [69] require real-time transmission as well as large and significantly *varying* amounts bandwidth as shown in Figure 2.4. This creates two issues in the current design of congestion control mechanisms. Firstly, the assumption that the situation of traffic from one instant to the next will not change significantly is now incorrect. Figure 2.4 shows the bandwidth requirement of a video conferencing session and tele-immersive session with a single 3D camera. The graph shows that there can be bursts in the data that last for very small instants. Owing to these variations in data (or bursts) the network traffic condition may vary significantly from one time instant to the next.

While congestion control mechanisms are important to the work presented in Chapter 3, they are not the focus. Thus Chapter 3 will only briefly look at the performance of congestion control mechanisms, in particular DCCP, with the current QoS architecture Diffserv.

In the application layer work has been done to ensure best performance under the available bandwidth conditions. This includes work, such as scalable video codec [71], progressive meshing [65], bandwidth reduction techniques for tele-immersion [103], and frame skipping [38] amongst others. These solutions provide a prioritization scheme over the data. For example, it is more important to send I frames than B/P frames. Thus, under low bandwidth conditions the higher priority data alone is sent to conserve bandwidth. However, these approaches make similar assumptions about the behavior of the network as congestion control, such as inexistence of short-lived bursts. Further, their interactivity with QoS mechanisms has not been studied and is inefficient, as this thesis will demonstrate inefficient.

A major part of the inefficiency is because there is no way the application can communicate this prioritization to the network. Thus, a network error has to be first communicated back to the application, which then reacts to it. They may therefore adapt to congestion situations that may no longer be present, thereby causing an under-utilization of the available provisioned bandwidth. Chapter 7 hints at how the architec-

ture recommended in Chapter 3 provides an infrastructure for the application layers to communicate their data preferences to the network.

2.3 Distributed Synchronization

Distributed synchronization of the primary media streams is required in the architecture in order to provide common ground during synchronous social experiences [98]. Traditionally, research on media synchronization is divided into two subtopics: intra-media, intermedia [75]. The first one, intra-media, refers to the correct temporal display of a single stream of media items (e.g., audio and video). The second one, intermedia, refers to the synchronization between several, possibly multiplexed media streams (including multicast) and the dependencies on the devices buffer lengths and the network jitter, among other issues. In recent years a third form of synchronization event-based has appeared. Event-based synchronization refers to the delivery of events across multiple nodes in a way that consistency of the media application is maintained. A subset of event-based synchronization is application-to-media synchronization, which allows (interactive) applications to be delivered in sync with the piece of content they refer to by adding triggers into the broadcast video stream which are parsed at the end user. In this specific case the event is not a user event, but a media event, whose timing can be known in advance (e.g., a commercial within a television program) or unknown (e.g., a goal by a football team). Table 2.1 shows the different use cases of media synchronization, together with a representative example, and its relevance for this thesis.

Table 2.1: Types of synchronization

Synchronization type	Example	Relevance to this work
Intra-stream	Video rendering	Not relevant
Interstream	Multiplexed stream audio and video in the same stream (one rendering component and one source)	Not relevant
	Multisource video (IPTV) and subtitles (internet portal)	Relevant
	Group synchronous video watching across different locations	Relevant
Event-based	Distributed gaming: playing across different locations	Partially relevant
	Social communication: audio chat, while watching television	Relevant

Figure 2.5 complements Table 2.1 and demonstrates also the level of research in each of these subfields. From the figure it becomes clear why it is important to address the topics of group synchronization from multiple-sourced media streams.

Intra-stream synchronization refers to maintaining timing relationships within a particular stream. For example, for a 30 fps video this means that the renderer must display every next frame within 0.033s. The inherent difficulty in doing so is due to (1) network delays and loss of ordering and (2) the fact that hardware is typically

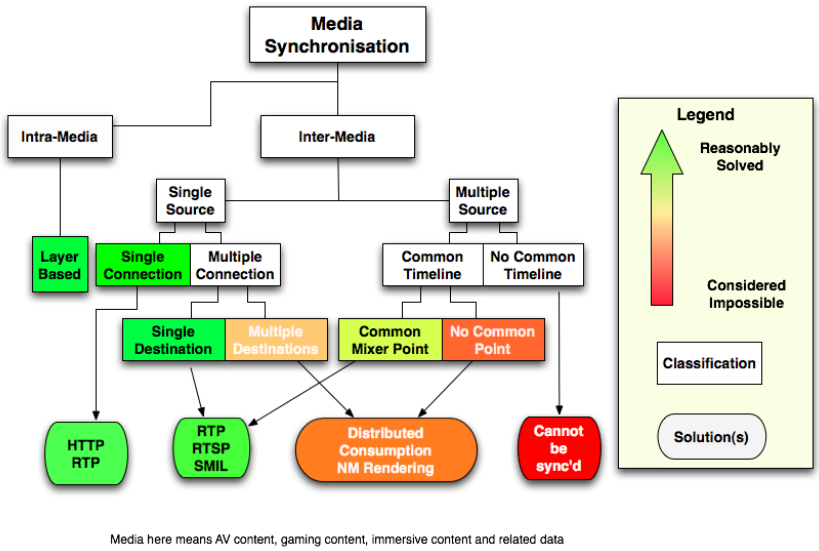


Figure 2.5: Figure demonstrating level of research effort in each of the topics under media synchronization

quantized, i.e. most rendering hardware devices handle chunks of data which may or may not align correctly with a frame size of the media item. This issue can be resolved by buffering and providing play-out feedback [74] to the sender to adjust send rate accordingly. The issue then resolves to making sure that problems of underflow or overflow do not occur, which is in the domain of QoS issues. Barring certain specific cases intra-stream synchronization is no longer considered a major research topic, as shown in Figure 2.5.

Interstream synchronization refers to synchronization between two separate streams of media, for example, audio-video lip synchronization [57]. In the case of interstream synchronization, we can identify three subcases:

- multiplexed streams (e.g., audio and video in the same transport stream)
- group synchronization (e.g., synchronized viewing across multiple destinations), also known as InterDestination Media Synchronization(IDMS [89])
- multisource synchronization (i.e., distinct per-stream-item transport streams)

The first case is handled at the multiplexing codec container level, and its similarity with intra-stream synchronization makes it less relevant for this thesis. The other two use cases impose a number of challenges that fit within the scope of our scenario.

Group synchronization refers to a shared viewing experience of peers in different locations. For example, to provide video playback to a number of friends in different locations: one using a mobile phone and another one using the IPTV connection. A number of synchronization algorithms exist. These can be divided into three types (1) master-slave, (2) synchronization maestro, and (3) distributed control. In the master-slave scheme [44] the master receiver multicasts its play-out position to the slave receivers which have to adjust accordingly. The *maestro* scheme [39] on the other hand is a more client-server based approach in which each receiver reports its play-out status to a central synchronization server or *maestro*. The maestro then calculates appropriate adjustments for each player and sends these a control message. In the distributed scheme [43], all receivers exchange messages to perform the job of the maestro in a more distributed fashion. A comparison between the distributed and the centralized scheme is presented in [41].

Multisource synchronization is a subset of group synchronization, see Figure 2.5. multisource synchronization considers the case in which media elements are sourced from different servers and possibly from different networks. Some examples include the provisioning of subtitles from an Internet portal synchronously with video served from an IPTV provider. Other examples might be the synchronization of a video broadcast from a mobile phone with Google maps that indicate the location of the video. Multisource synchronization has not been well studied so far, primarily because the infrastructure and the scenarios for doing so were missing. However, as can be seen in Figure 2.1, users need not belong to the same service provider and may stream the same video content via different providers. In such a case group media synchronization schemes, which assume a central node in the media delivery path, would not work. The realization for the need of such an algorithm is slowly but steadily dawning on the research community. As an example, while ETSI TISPAN introduced IPTV services in Release 1 of its NGN specifications [54], it neglected explicit media stream synchronization aspects. With Release 2 specifications, the need for synchronization of IPTV media/content was recognized. Recently, a standard for synchronizing shared IPTV services using the RTCP-XR block is being defined at the IETF [89] as a parallel development with contributions from this thesis and the TISPAN NGN specification [54].

Finally, the last class of synchronization is called event synchronization. There are a number of relevant scenarios such as distributed gaming, instant messaging solutions and VCR-like distributed control execution while watching streaming video. Event synchronization can be considered as a subclass of group synchronization. The basic

problem arises from the differences in event delivery time (jitter) seen by participating nodes. This leads to inconsistent copies across the nodes playing the same media. In this area [59, 104] proposed an algorithm called *local lag with time-warp* to handle this inconsistency in distributed games. The general approach is to estimate the worst delay in the network for a participating node and to enforce this delay on all participating nodes, including itself. If an inconsistency is found at a participating node then the events are rolled back to the last known consistent state and played fast forward to reach the current time. A similar concept based on the local lag mechanism is used in synchronizing distributed games, called bucket synchronization, is presented in [7]. In this thesis we will adapt this algorithm to media playback in our architecture, phrased as Question 1.5. We then use this algorithm to study user tolerance to differences in synchronization levels in a synchronized social video watching environment, thereby addressing Question 1.4.

2.3.1 Media Synchronization Across Domains

One of the challenges addressed by this thesis is to provide a converged view on synchronization across several network domains (e.g., Web and mobile). Such synchronization is not restricted to content consumption (i.e., intermedia) but also includes interactivity between users (i.e., event based). Moreover, we consider both the playback/media on demand model as well as real-time broadcast.

The target domains of this thesis (i.e. the Internet, IMS, 3G/Mobile) use different media delivery carriers:

- RTP streaming as used in the Internet in various media and communication applications, including mobile applications.
- Broadcast streaming used by IPTV (DVB-IP, also using RTP but with a different adaptation layer compared to native RTP)
- HTTP streaming used in the home domain and for Video on Demand (VoD) delivery on the Internet (e.g. progressive HTTP streaming)

RTP is appropriate for both interactive (human-to-human) communication (for example, VoIP applications) and non-interactive media delivery such as streaming (for example, VoD using RTSP, or interactive voice or video responders using SIP or H.248). RTP is tailored to be resilient to common QoS degradations of the Internet, using packet-loss concealment procedures that can be specialized per codec. RTP, in conjunction with its peer control protocol (RTCP) also provides built-in interstream time synchronization mechanisms as long as all the streams pass through a common RTP

mixer. In this respect RTP alone is not sufficient for this work, since we include synchronizing streams originating from entirely different sources.

HTTP streaming is suitable for non-interactive non-real-time media delivery, such as file sharing. However, it has met considerable success in single end user video-streaming as well. Classical examples are seen on the Internet, the most popular of which is YouTube.

2.4 User Mobility

As can be seen in Figure 2.1 end users in synchronous shared experiences are increasingly surrounded by varying devices which connect to the Internet using different technologies and service providers. In such a situation users may want to move their media from one device to another for various reasons identified in [58]. These reasons include cost effectiveness, better user experience, and physical movement of the user amongst others. This is called *session mobility*. Furthermore acceptance of newer standards in structured content, such as NCL [86] and SMIL[4], presentation will lead to more complex media which may be composed of many individual media sessions. In such cases it is important to talk of *user mobility* separately from *session mobility*.

Session mobility is an ongoing research problem. Previous research [58] has proposed two parallel solutions: the network-centric and the device-centric approach. The network-centric approach delegates the responsibility of transferring the presentation to the network, thus, the user may explicitly or implicitly initiate a transfer call. The advantage of this approach is that it would work with the broad variety of user devices currently in the market. The device centric transfer on the other hand requires the target or the originating device to discover the other device and then *push or pull* the session. Both these approaches, however, do not take into account the type and the complexity of the media presentation. While these architectures would work for live media, more coordination between the devices would be required for media playback. Historically it has been assumed that synchronization would later be solved by the underlying network layer. This, however, reduces the user experience of the presentation during and some time after transfer since the presentation is not synchronized then.

In [52], the authors present how MPEG-21 digital items can be used to realize session mobility between two devices and how it can adapt the session to different devices according to their capabilities. In their method, they use two kinds of digital items: CDI (content digital items) to carry the content and XDI (context digital items) to carry the context information. However, it is not straightforward using digital items to express complex intermedia temporal and spatial relationships.

[4] http://www.w3.org/AudioVideo

The issues of user mobility for multiple related session (distributed rendering) was solved by [14] as an aggregate of multiple session mobility requests. The authors proposed a SIP-based mechanism by extending the user agent to split a session over multiple devices. Specifically, they proposed a SIP extension header "Mobility" to improve the call transfer mechanism and make it transparent to the remote party. To solve the problem that the user needs to terminate all devices separately when a session was split over multiple devices, they proposed the concept of "Association" in the corresponding note. The association record contains the set of all call-IDs that belong to the same session. As an answer to Question 1.7, this thesis presents a much simpler approach to solve the same problem by delegating the presentation consistency responsibilities to the media player on the target device in Chapter 5. Chapter 5 argues that user mobility must be handled at the presentation layer and not at the sessions layer, and presents some advantages of doing so.

Papers [81, 79] look at transferring media sessions (typically RTP streams) for live conferencing systems. The authors in [82] present in detail media session transfer to various devices based on the Session Initiation Protocol (SIP) for signaling and Session Description Protocol (SDP) for session description. Transferring sessions in live media sessions as mentioned before, in Section 2.2.2 does not require detail state information of the presentation as it is live and incorporates limited user interactions. This is different in media playback, especially in complex presentation, where alternative media encoding may be available at the streaming server, which may be especially suited to the new device. As mentioned before maintaining state for a complex media presentation while transferring the session is an important requirement. Lastly, media presentations may have complex user interactions, which need to be recorded within the state. Thus, the basic drawback of using this solution arises from the fact that the state of the session before it is transferred is stored as a Session Description Protocol (SDP) file. While SDP is extremely powerful in describing a range of sessions, media, and device characteristics, it isn't powerful enough to describe the complete state of a complex presentation. In addition, the problems of time synchronization and adapting media to appropriate device without transcoding remain.

2.5 Time Synchronization

The architecture presented in Figure 2.1 implicitly assumes some level of time synchronization. The de facto standard of time synchronization over the Internet is the Network Timing Protocol (NTP). NTP [60] works by creating multiple possibly cross-linked multicast tree hierarchies over the Internet. At the root of these tree is always a high accuracy hardware timer such as an atomic clocks GPS-based timer, or radio clocks. Time "flows" down the NTP multicast tree using a packet exchange method

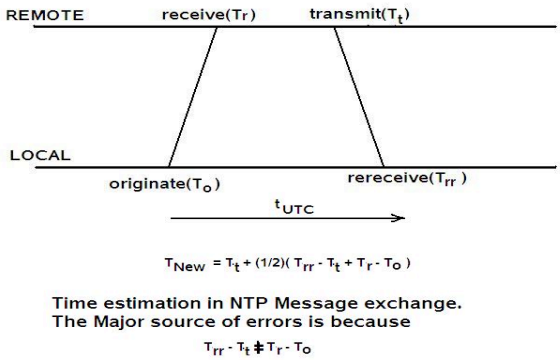


Figure 2.6: NTP message exchange

shown in Figure 2.6. Other protocols, such as, Precision Time protocol (IEEE 1058) [23], also exist but are not widely used over the Internet.

The other option for time synchronization is, of course, to fit all computers with accurate hardware clocks such as radio or GPS receivers. However, both these methods require line of sight to radio transmission towers/ GPS satellites to be effective. This is not always possible. Recently, gossip based protocols have appeared [20] including a protocol for time synchronization, the gossiping time protocol (GTP) [47]. Given master node in the network the protocol sets up an ad hoc multicast tree using random NTP like packet exchange. In O(log n) this random exchange of packets leads to most of the network being synchronized with the master node. A variation of the gossip based protocol as a supplementary contribution of this thesis is presented in Appendix B.

Time synchronization in infrastructure networks using NTP like protocols typically relies on the bounded round trip time of network messages. However, in cross-domain networks the RTT may experience high jitter and local NTP servers may be absent or unavailable. Wireless ad hoc network protocols have similar properties. The similarities include a distributed approach, low network overhead, overcoming unreliability in round trip times. We can thus look at wireless ad hoc network synchronization algorithms and adapt them here.

Algorithms presented in the IEEE 802.11 standard and subsequent enhancements presented in [85, 77, 105] work well for ad hoc networks. These algorithms synchro-

nize clocks using a global signaling mechanism (beacon method), are distributed and achieve (especially [105]) very high synchronization results. They, however, suffer from some disadvantages, in particular: they do not have the capability of choosing a particular node to follow. The beacon broadcast present in all these algorithms is also hard to extend for non-homogeneous networks.

Synchronous shared experiences, however, will run over the Internet. And even though a number of other time synchronization algorithms exist, NTP remains the predominant one over the Internet. Chapter 6 surveys the NTP network to answer Question 1.7. Similar surveys have been performed before in [68], the last one being five years ago. In the last five years, however, significant changes have happened to the Internet and the survey in Chapterchap:tsync highlights these differences with the previous works.

2.6 Summary

The goal of this chapter was to identify a generic architecture for synchronous shared experiences. Furthermore, this chapter specified the responsibilities of each of the components in the architecture, with respect to achieving coherence in synchronous shared experiences. The questions were classified into four main topics: quality of service, distributed media synchronization, user mobility and time synchronization. For each topic a review of the state of the art was presented and the shortcomings highlighted. Each of the following four chapters in this thesis will handle each of these topics and answer the questions raised therein.

CHAPTER 3

Quality of Service: Estimated Service[1]

Data transmitted by applications providing communication solutions in a synchronous shared experience, such as video and audio conferencing, is termed in this thesis as *communication stream*. These communication streams require time-bounded delivery, with end-to-end bounds of the order of $150ms$ [46] (real-time transmission). This chapter illustrates that the current Internet does not efficiently support communication streams and presents extensions to improve this efficiency. The future Internet must remain best-effort and thus real-time bounds cannot be guaranteed. The architecture of the Internet, however, can be made more efficient - in terms of throughput for the communication streams - measured as the amount of data that successfully makes its deadlines. The current approach of doing this is to over-provision the network making QoS mechanisms unnecessary. Thus, first, it is important to understand why quality of

[1] This chapter is based on publications

- I.Vaishnavi, D. C. A. Bulterman.*Estimate and serve: scheduling soft real-time packets for delay sensitive media applications on the internet.* Proceedings of ACM workshop on Network and Operating Systems Support for Digital Audio Video, 2009.
- I. Vaishnavi, P. Cesar, D. C. A. Bulterman, and O. Friedrich. *From iptv services to shared experiences: Challenges in architecture design*. Proceedings of IEEE Conference on Multimedia and Expo, 2010.

service (QoS) mechanisms are needed in the Future Internet.

It is commonly perceived that the bandwidth available in the Internet far exceeds the demand [78] and thus QoS mechanism are unnecessary. Instead, this chapter argues that the reality is in fact the opposite of this perception: Internet today gives an *appearance* of over-provisioning, since applications that cannot be supported simply do not become popular. In this way the current Internet acts as a bottleneck for further development of applications [70, 92], and of particular relevance to this work, communication streams. For example: the Internet2[2] between University of Illinois at Urbana-Champaign (UIUC) and the University of California at Berkeley (UCB) was designed to provide a high speed (Gbps) data link between various universities. Studies conducted within this thesis show that Internet2 connection between UIUC and UCB currently provides an average 8Mbps per-connection to TCP and about 50Mbps to UDP. This makes Internet2 already unable to support a single instance of the tele-immersive application [101] being developed at UIUC and UCB. Similarly, bandwidth studies between Centrum Wiskunde & Informatica in Amsterdam (10Gbps uplink to its service provider) and UIUC (Intermet2 uplink) at Urbana-Champaign show a TCP bandwidth of less than 1Mbps on an average and a UDP bandwidth of 5Mbps. Despite the high bandwidth uplink connections to their individual service providers, an end-to-end connection results in bandwidths that are barely enough to hold two instances of SD (standard definition) quality video conferencing sessions.

To overcome these limitations at the application level, various bandwidth adaptation mechanisms have been designed, such as scalable video coding for efficient video distribution over the Internet, progressive 3D streaming mechanisms [65] and prioritized view transmission [100] for 3D immersive applications. These bandwidth adaptation mechanisms are important for keeping the network stable. Bandwidth adaptation mechanisms, however, consider the network itself as a black box and try to adapt to it. Considering the network as a black box implies that these adaptation methods can not know anything about the actual state of the network. This results in inefficient end-to-end functioning of the application, in particular, the end-to-end throughput, as will be demonstrated in this chapter. This loss of end-to-end throughput due to bandwidth adaptation and congestion control mechanisms [76, 22] is a well-understood problem.

At the network level, in the current Internet, the only way of transporting real-time traffic across continents is by providing for the maximum bandwidth required (over-provisioning). This is because newer communication applications do not present a uniform bandwidth requirement profile as shown in Figure 3.1[3]. They further need to meet end-to-end deadline bounds of about 150ms. If the network is not overprovisioned, the occurrence of bandwidth peaks result in dropped packets, triggering bandwidth adapta-

[2] http://www.internet2.edu/

[3] Measured using Eureka, Appendix A

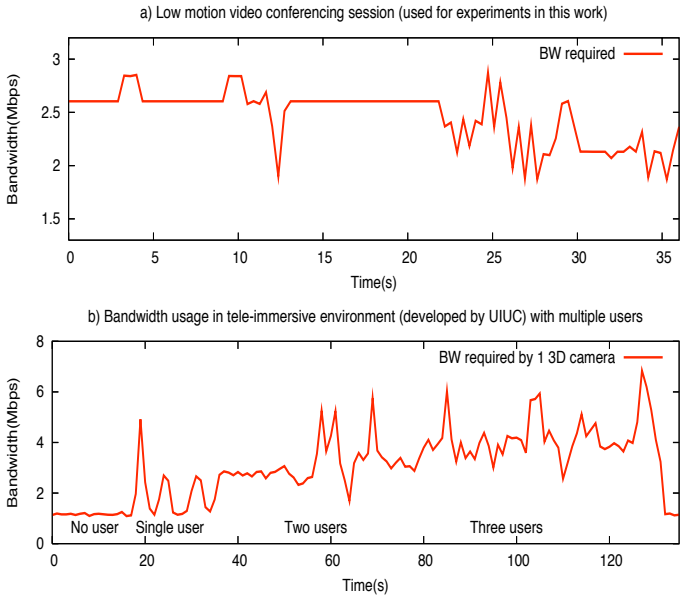


Figure 3.1: Measured bandwidth requirements for newer communication applications.

tion mechanisms (or congestion, flow control, if TCP-like protocols are used) causing the application to use even lesser bandwidth, further reducing bandwidth utilization.

Real-time traffic from conferencing streams requires priority over other traffic, due to short delay and predictable jitter requirements. This prioritization blocks out other traffic creating an overprovisioned link for real-time traffic. The current standard for providing prioritization is the Differentiated services (Diffserv) networks architecture [36]. Differentiated networks work by classifying packets into behavior classes, known as *per-hop behaviors*. All packets in a class (per-hop behavior group) require a similar type of service. The real-time group (or *Expedited Forwarding Class, EF*) is recommended for communication streams and has priority over other classes of traffic. However, Diffserv-EF was designed to provide an overprovisioned network by blocking the non-priority traffic out. Thereby Diffserv inherently assumes the right balance in band-

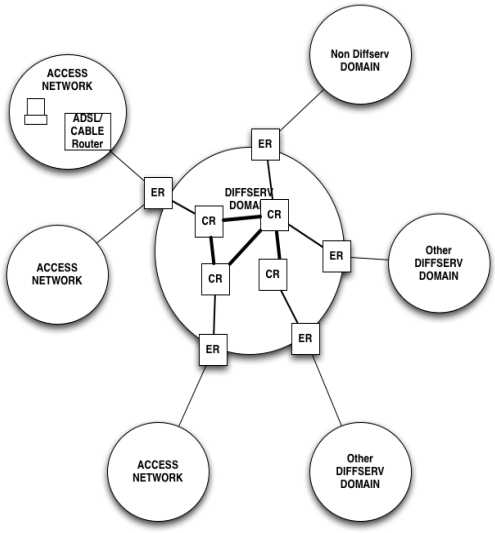


Figure 3.2: Diffserv Internet architecture: ER = Edge Router, CR = Core Router

width availability and demand. If the demand is much too low then Diffserv altogether is unnecessary. If the demand is much too high an overprovisioned priority link will not be possible. In practice this balance is difficult to achieve. Thus one must study the behavior of Diffserv in networks that are not overprovisioned.

Furthermore, overprovisioned links are not always feasible, predominantly because of economic reasons. The current Internet is split into domains as shown in Figure 3.2. Typically each of these domains is maintained by a company or a public institution and is connected to other domains or access networks via edge routers (ER). From a business viewpoint domains, which are connected to each other, need to negotiate service level agreements (SLAs). The addition of new users or higher bandwidth applications to any one network will in effect then require an update of all SLAs throughout the network. This is practically unfeasible. Hence, while Diffserv imposes a requirement of an overprovisioned link, this requirement cannot be blindly assumed. Therefore, for practical acceptance, it is still important to study its performance in a network that is not overprovisioned, in particular:

Question 1.1 How efficient is the Diffserv network towards communication streams, such as video conferencing, in networks that are not overprovisioned?

Section 3.2 describes the experiments and the results thereof performed over a Diffserv link to study its performance in networks that are not overprovisioned. In links that are not overprovisioned bandwidth adaptation methods become significant again. Measurements presented in Section 3.2 indicate a loss in end-to-end throughput due to adaptation mechanisms. The future Internet architecture must be able to maintain end-to-end throughput efficiency even in non-overprovisioned links, in particular:

Question 1.2 How can Diffserv-EF be extended to ensure efficient behavior of the future Internet in links that are not overprovisioned while maintaining scalability provided by the Diffsev architecture?

The Diffserv standard provides a scalable, best-effort prioritization scheme. Any new additions to this architecture should be evolutionary in nature. Thus, this work provides an extension to the Diffserv-EF channel that makes it more efficient in terms of bandwidth utilization. Unless otherwise specified in this chapter, only data that meets it deadlines is counted towards bandwidth calculations.

Lastly, only ensuring efficiency is not enough. It is also important to ensure that applications get minimum bandwidth service guarantees, or are refused admission if the real-time channel is too crowded. This is known as connection admittance. Connection admittance is required since communication applications cannot work under a certain minimum bandwidth. Such applications are known as *inelastic* applications. Previous work on connection admittance, done in the integrated service networks [8] architecture did provide connection admittance but failed in keeping the system scalable and economically efficient [3].This chapter addresses the question

Question 1.3 Is it possible to provide connection admittance in the future architecture, while still maintaining scalability provided by the Diffserv architecture?

The goal is to achieve better efficiency for bandwidth utilization as well as connection admittance for time-bounded applications. And doing so without maintaining any per-connection memory structures at the routers, thus keeping the architecture scalable.

The next section presents a generic test scenario used to evaluate performance of existing and proposed network architectures throughout this chapter. Section 3.2 looks at the performance of an non-overprovisioned Diffserv link answering the first question. It shows that the Diffserv architecture inefficiently utilizes available provisioned

bandwidth for real-time applications and is therefore not scalable. This is followed by our solution, Estimated service (Estserv) in Section 3.3. Section 3.3.5 presents the results of our new architecture, Estimated service, and shows that Estserv utilizes the available bandwidth efficiently even in under-provisioned links and provides scalable connection admittance in an over crowded network.

3.1 Generic Test Scenario

This section presents a scenario and the related test setup used to evaluate various network architectures in this chapter. Two students, Caroline and Kate, share an apartment. Their maximum Internet usage is when they simultaneously each have a video conferencing session with their families, while their P2P client is downloading videos from the Internet. When they decide to obtain a new Internet connection they are presented choices[4] for the bandwidth: 3Mbps, 5Mbps, 10Mbps or 25Mbps, each with half of the bandwidth reserved for real-time traffic and the other half for best-effort.

Figure 3.1a) shows that a typical video conferencing session requires an average of 2.5Mbps. Therefore, a 5Mbps real-time channel for Caroline and Kate should be enough. Our experiments with Diffserv-EF, presented later in Section 3.2, will show that due to bursts in the video stream, packets are dropped. This triggers bandwidth adaptation mechanisms, resulting in *under-utilization* of the available bandwidth. Thus, either the ISP has to actually provision for a 6Mbps on a 5Mbps real-time channel or Kate and Caroline need to buy a 25Mbps total connection instead of 10Mbps (5Mbps real-time, 5 best-effort) total connection. In the first case the ISP can serve fewer customers and in the second the users have to pay for more than what they actually use. Both scenarios are undesirable. This is termed as *over-provisioning*. Further it is important to provide some sort of best-effort connection admittance mechanism to prevent a collapse of their real-time connection in case they decide to start a large number of conferencing streams simultaneously.

While the scenario talks about only two end users, it scales to system administrators of larger institutions and companies. While the core of most ISPs may have more than sufficient bandwidth they will only expose bandwidth at the edges that has been paid for. Thus the edge nodes incorporate a policing function to limit the incoming as well as the out going traffic according to the agreed upon terms in the Service Level Agreement (SLA). The following test network setup used for experiments throughout this chapter replicates this edge node behavior.

[4] Symmetrical in upload and download for simplicity

3.1.1 Test Network Setup

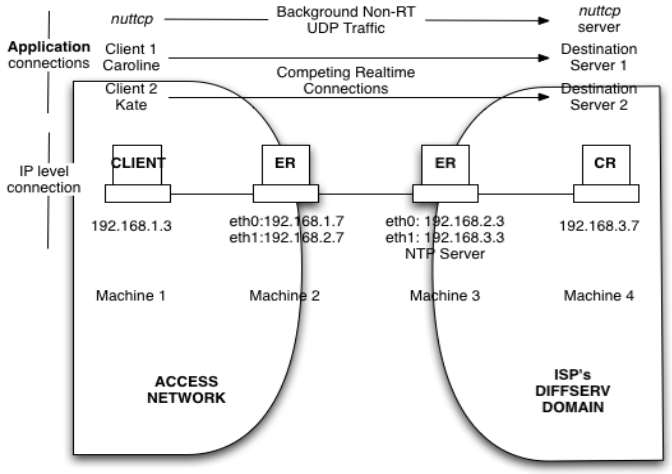


Figure 3.3: Overall network setup

In the experimental setup Linux machines are interconnected in a linear topology as shown in Figure 3.3. Machines 2 and 3 are the edge router (ERs) between an access domain pseudo clients on (machine 1) and an ISP core router machine 4. For simplicity, machine 4 is also the destination for machine 1 clients. The experiment measures the application's bandwidth behavior across the edge. Using the tc linux utility on the machine 2 and 3 a Diffserv edge routers are set up. Figure 3.4 shows this setup. Various network queueing disciplines are known as *qdiscs* in Linux terminology. The box marked as QDISC in the figure is a FIFO for Diffserv-EF. The figure shows a prio (priority) qdisc for ctrl/NTP traffic followed by a DSMARK (Diffserv) qdisc with a *tcindex* filter (a Diffserv Code Point (DSCP) classification filter) followed by an hierarchal token bucket (htb) qdisc of maximum bandwidth ceiling 10Mbps. An htb is normally chosen over a simple token bucket so that the bandwidth unused by higher classes can be appropriately redistributed amongst the lower ones. Smaller bandwidths are chosen for the experiments in this chapter so that the delays on one computer are not significant and the bandwidths are in accordance with the scenarios in Section 3.1. The htb qdisc consists of 2 classes the Expedited Forwarding (EF) class and the Best-Effort (BE) class. The EF and the BE classes are given a reserved bandwidth of χ_{EF} and χ_{BE} Mbps. The values are decided on a per-experiment basis. At first BE TCP traffic

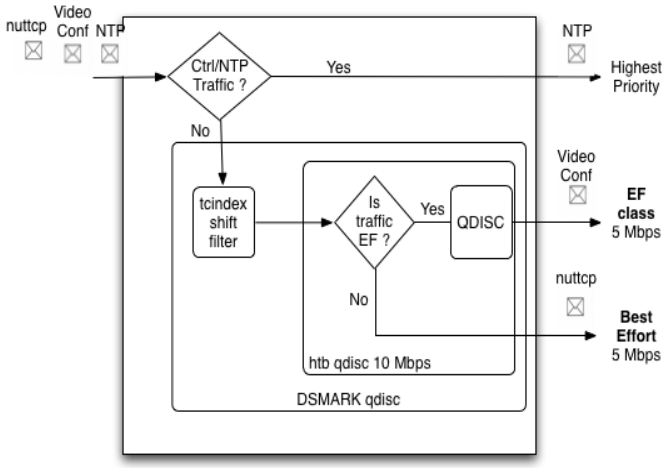


Figure 3.4: Packet classification for providing QoS used in eth1 of machine 2 and 3 (Figure 3.3): For traditional Diffserv: QDISC = FIFO, for Diffserv extended to Estserv : QDISC = Estserv

is sent using the $nuttcp$ utility from node 1 to node 4 representing the best-effort non real-time P2P connection. $nuttcp$, is a linux bandwidth measurement utility tries to send TCP packets to machine 4 from 1 and reports the approximate throughput (± 0.5 Mbps).

Two video conferencing (C++) client programs on machine 1 (Kate, Caroline), compete for the real-time bandwidth by sending the next frame every $1/(f.p.s.)$ seconds to machine 4 of the video belonging to the profile shown in Figure 3.1a). The deadlines are 100ms $= d - T_{app} - T_{cap}$. This video, same as the one in Figure 3.1a), requires an average bandwidth of 2.5 Mbps. The video conferencing application use UDP as the transport protocol as explained in the next section.

3.1.2 Transport Protocol Used

This section justifies the choice in transport protocol used over the test network setup presented in the previous section. The problem of under-utilization of available bandwidth is due to either bandwidth adaptation mechanisms (with UDP) or reliability and congestion/flow control mechanism (TCP). TCP is not meant for real-time traffic and is therefore not studied in this work. Datagram Congestion Control Protocol (DCCP)

[22] is a recent protocol recommended for real-time applications. DCCP is in essence UDP with congestion control, since normal UDP is aggressive and excessive use may cause the Internet to fail. Preliminary experiments with DCCP are presented later in Section 3.3.8. This chapter focuses on the traditional protocol recommended for real-time traffic: UDP. UDP is typically used with application level bandwidth adaptation mechanisms, such as scalable video coding [71]. Therefore, to perform this study an ad hoc bandwidth adaptation mechanism based on frame skipping [38, 30] was developed. this adaptation mechanism is presented in the following section.

Table 3.1: List of symbols used in this Chapter

Symbols	Explanation
χ	Bandwidth
N	Number of hops
ψ	Transmission speed of a router
C	Channel utilization
d	Deadline
μ, q_{len}	Queue length at a router
T_{app}, P_B	Receiving side processing time, P is probability distribution of T
T_{cap}, P_A	Capture and sender processing time, P is probability distribution of T
t_{fwd}	Forwarding time at a router
P	Generic probability density function
g	Function that yields the probability density function for the end-to-end time
$p(t = x)$	Probability that $t = x$
$p_i(t = x)$	Probability that per hop time, $t = x$ at router i
p_{est}	Minimum time for a router to forward packets
t_{dl}	Absolute time stamp of the deadline
t_{now}	Time now
t_{for}	Time taken to forward a packet
p_j (not as a function)	Available per hop time for packet j
f	Estimate of end-to-end time
p_{con}	Congestion estimate at a router
f_{incr}, f_{decr}	Increment and decrement functions for p_{con}

3.1.3 Bandwidth Adaptation Mechanism

This section presents an ad hoc bandwidth adaptation method used in the tests based on the on the frame skipping mechanism [38, 30]. All multimedia applications have a minimum bandwidth requirement below which they cannot function and an ideal bandwidth at which the user experience is optimal. This allowance is used by various applications to adapt themselves to the estimated network bandwidth availability. For the video in the experiment a frame rate of optimal 30fps and a minimum 10fps is used. There are two parts to any bandwidth adaptation: congestion detection and quality reduction.

In our mechanism congestion is detected when the acknowledgement for a heartbeat packet sent out every M data packets is not received within 500ms or if the heartbeat packet misses its associated deadline. The heartbeat packet is just another data packet with a special label marked on it. When this congestion situation is detected quality reduction is performed by a probabilistic dropping procedure is engaged at the client. Every time congestion is detected the video conferencing application reduces the video quality by reducing the frame rate uniformly in steps of 5 frames per-sec. These frames are never sent onto the wire by the client. This is done until a minimum of 10fps is reached, below which a connection loss is said to have taken place. The frame rate is increased by 5fps, if two consecutive heartbeat packets are properly acknowledged. More advanced ways of dropping the bandwidth requirement based on congestion control techniques, video encodings can be used but are not the focus of this work. The bandwidth adaptation mechanism described here is in someways similar to [38].

3.2 Performance of Diffserv-EF in non-overprovisioned links

This section presents the performance studies done on a non-overprovisioned Diffserv-EF link. Non-overprovisioned links are defined as links where the bandwidth availability in the network is below that of the maximum bandwidth requirement of the application. Non-overprovisioned networks can therefore be further classified into three categories based on average bandwidth requirement of the application: (1) above average bandwidth availability, (2) average bandwidth availability, and (3) below average bandwidth availability. This section examines the performance of a Diffserv EF link under these three conditions.

Previous work [37, 88, 33, 28] also studies the performance of various Diffserv implementations for various traffic loads. While these papers study Diffserv performance under heavy load, none of them study the performance of Diffserv-EF in non-

overprovisioned conditions. Furthermore, most previous work does not perform the tests in combination with bandwidth adaptation mechanisms, which are used by default in all modern multimedia applications. All previous work considers short queue lengths for the Diffserv-EF class to ensure end-to-end predictability. However, the actual values used vary from 5-500 packets in [28] to ensuring 25ms deadlines in a closed network [88]. The Diffserv-EF standard [19] also recognizes the influence of queue lengths on delay. However, the standard is silent on what the recommended queue length is for an ISP system administrator on the Internet. The next section investigates this value mathematically. A number of symbols will be used in the discussions that follow. An overview of all the symbols used in this Chapter is presented in Table 3.1.

3.2.1 Setting up Diffserv EF - Short Queue Lengths

Short queue lengths are recommended to ensure end-to-end delay predictability. The length of the queues for closed networks can be determined based on the targeted delay, since the maximum hop count and the topology is known. However, for system administrators of ISPs this is unknown. The practice, therefore, is to assign very short queue lengths (less than 5 packets) for the EF class to ensure end-to-end delay predictability. This practice can be mathematically justified as follows. Let us assume that the queue length at a router is μ Mbit, the number of hops in the connection is N and the deadline is d, the physical connection speed of the router is ψ, T_{app} is the time required by the end application processing, T_{cap} is the time required for source capture and processing. If the deadline is divided over all routers equally then

$$\frac{d - T_{app} - T_{cap}}{N} > \frac{\mu}{\psi} \tag{3.1}$$

$$\mu < \frac{(d - T_{app} - T_{cap})\psi}{N} \tag{3.2}$$

Diffserv is a generic architecture and cannot be adapted to each application requirement individually. Assuming generalized worst case values:

- $d = 150ms$, an acceptable [21] value of delay for interactive multimedia applications.
- $N = 50$ is the high end hop count seen in networks (the standard TTL is even higher at 128 or 256).
- $T_{app} + T_{cap} = 50ms$ worst case capture, processing and reconstruction time for 3D immersive applications

yields Equation 3.3.

$$\mu < \frac{(d - T_{app} - T_{cap})\psi}{N} = .002\psi \tag{3.3}$$

According to Equation 3.3, each router in the Diffserv network on the Internet should assign the queue length based on the outgoing bandwidth available to its Diffserv-EF class if weighted fair queuing (WFQ) is used to schedule between various classes. However, if the EF class is given absolute priority over all other classes, then the total outgoing bandwidth of all the classes together must be used. Thus, for an outgoing home connection, policed at 10Mbps[5] total upload the maximum queue length that it can have is 2 (1500B) packets to ensure any end-to-end deadline is met. Thus while a 5Mbps EF connection should be able to support two of the video conferencing sessions of bandwidth profile shown in Figure 3.1a)[6], the short queue length means that it can only effectively support one at full quality. While in the future better connection speeds will become cheaper, there will also be newer applications such as 3D immersion that require equally more bandwidth. Lastly, the future Internet must be incorporative and consider countries that are less well connected to the Internet. In the future these countries will account for a higher percentage of the Internet bandwidth usage.

3.2.2 Results

Utilizing the setup detailed in the above sections three tests were conducted to analyze the performance of Diffserv-EF under non-overprovisioned networks for:

- Case 1) above average provisioning,
- Case 2) average provisioning and
- Case 3) below average provisioning.

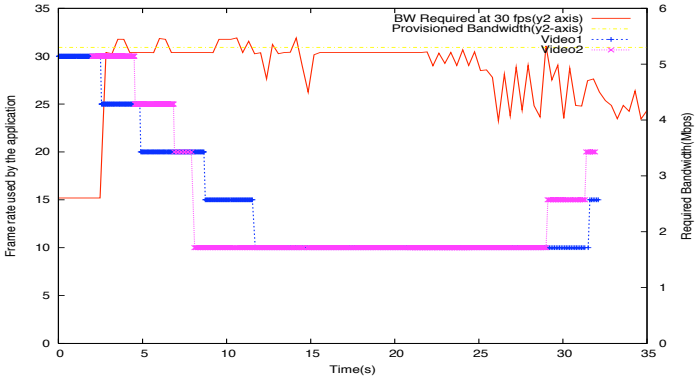
The videos used in the experiment require an average of 2.51Mbps and a maximum of 2.63 Mbps for their entire duration. Thus, for these two videos the values used are

- Case 1) $\chi_{BE} = 4.7Mbps, \chi_{EF} = 5.3Mbps$
- Case 2)$\chi_{BE} = 5Mbps, \chi_{EF} = 5Mbps$ and
- Case 3) $\chi_{BE} = 5.75Mbps, \chi_{EF} = 4.25Mbps$.

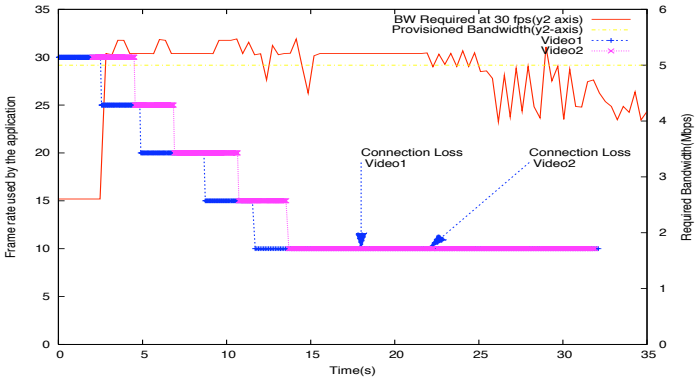
All packet sizes used in the experiment are 1024B, therefore in accordance with Equation 3.3 the EF class has a three packet queue length.

[5]This value is unaffected by the actual physical network capability at an average, which may be in more than 100s of Mbps, since the IP level policer will not enqueue packets on the physical transmit queue if they violate the 10Mbps upper-limit.

[6]assuming other classes are full

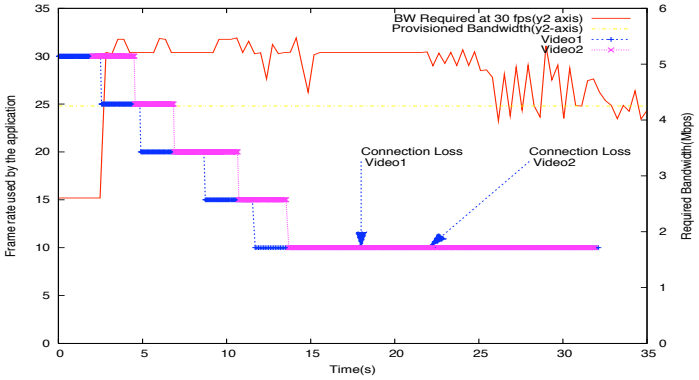


a) Case 1: Bandwidth provision above average
but below maximum requirement = 5.3Mbps



b) Case 2: Bandwidth provisioned at close to average requirement = 5Mbps

Figure 3.5: Results with non-overprovisioned Diffserv-EF. Case 3 on next page...



c) Case 3: Bandwidth provisioned at below average requirement = 4.25Mbps

Figure 3.5: ...Continued from previous page: Results with non-overprovisioned Diffserv-EF

The results of the experiment are presented in Figure 3.5 and Table 3.2. The figures show the bandwidth adaptation mechanisms behavior under the three cases, while the table shows the overall throughput for the session. The results answer Question 1.1 raised in Chapter 1. For cases 2 and 3 the results are similar, demonstrating that the loss of packets due to bursts triggers the bandwidth adaptation mechanism ultimately resulting in connection loss of both the videos conferencing sessions. Results for case 1 are mildly better and show that connection loss does not take place and towards the end the frame rate seems to recover a bit. The throughput, measured as the number of packets that successfully make their deadline, is shown in Table 3.2. Out of the 23 MB sent, a meager amount makes it successfully to the destination (machine 4) in all three cases. The third column in the table measures the channel utilization efficiency, C. This number is defined as:

$$C = \begin{cases} \chi_{measured}/\chi_{EF} & \text{if } \chi_{EF} < \chi_{Requested} \\ \chi_{measured}/\chi_{Requested} & \text{if } \chi_{EF} > \chi_{Requested} \end{cases} \tag{3.4}$$

where $\chi_{measured}$ is the average measured bandwidth from the experiment, χ_{EF} is the bandwidth provisioned in the EF class and $\chi_{Requested}$ is the average bandwidth required by the application at 30 fps. Thus a channel in the experiment is more efficient

Table 3.2: Results with Diffserv-EF for 2 simultaneous video conferencing connections Video 1 (Total stream size = 12M) and Video 2 (Total stream size = 11M).

	Received sizes for Video 1.	Received sizes for Video 2.	Channel utilization efficiency
Case 1	4.4M	3.6M	.348
Case 2	2.2M	2M	.189
Case 3	2M	1.9M	.2

if the value of C is closer to 1. This can happen either if it uses all the provisioned bandwidth or the application receives the complete bandwidth requested for. Note that for the purposes of calculation of utilization efficiency average bandwidth values over the measured period are used.

The loss of throughput, illustrated by the low channel efficiency values shown in Table 3.2, is unacceptable for the future Internet. To improve this inefficiency, and answer Question 1.2, this chapter proposes an architecture for real-time data transmission, called Estimated service (Estserv), as an evolutionary extension to the current Diffserv-EF class. Estserv makes the EF class aware of the timing constraints of the respective application data. These timing constraints are used to create a *scheduling* and an *estimation* mechanism over the real-time packets. The EF class can now have *longer* queues with "per-hop" earliest deadline first scheduling, while the estimation mechanism is used for "connection admittance". This scheduling mechanism results in the EF class being able to prioritize/reject packets on a per-packet basis. This prioritization is independent of the flow, does not maintain state or separate queues and performs efficiently in non-overprovisioned networks. Note that in adequately provisioned networks this architecture automatically resolves to the normal Diffserv-EF as there are no queues and therefore no scheduling is performed.

3.3 Estimated Service

The previous section demonstrates the dismal performance of the Diffserv-EF channel in non-overprovisioned networks towards real-time applications. To solve this problem this work looks at the single most defining characteristic of real-time applications: timeliness. This work draws from what the real-time community has been doing for years: earliest deadline first (due date) scheduling [56, 55]. This work adapts the earliest due date first scheduling mechanism to multi-hop networks. Similar ideas were presented before in [27]. The major difference between [27] and this work is the strict ad-

herence of the mechanism presented here to best-effort principles. Best-effort implies that the intermediate routers cannot maintain connection or queue states and cannot pre-reserve resources. The idea of earliest due date scheduling seems straightforward but it needs to be adapted to take other factors into account: firstly, deadlines are not the sole concern in networks, path lengths also play an important role. Secondly, there has to be a mechanism of connection admittance inbuilt in the architecture (Question 1.3), to ensure the network does not get over-crowded. Finally, multiple hops need to be time synchronized to each other.

With regards to time synchronization, in 1989 a study was conducted in [61] on the accuracy of clock synchronization provided by NTP over 100000 nodes distributed across the world. It found that the majority ($> 50\%$) of the clocks in the NTP network where within 10ms of each other, Enterprise networks, LANs, and WANs can do much better. These surveys of the current health of the NTP network in the Internet are re-updated in this work (Question 1.5). This is presented later in Chapter 6. The deadlines for end-to-end transport of communication streams are in the order of 150ms. This one order of difference in magnitude is inline with the principles of Time Triggered Architectures (TTA) [53], that are used real-time systems. Section 3.3.9 will consider cases with synchronization failure.The next section presents the theoretical overview behind the design of the estimated service architecture. It is worth pointing out that the theory is significantly different from its implementation to account for scalability issues. However, to understand the implementation, a general understanding of the theoretical framework is important.

3.3.1 Theory

Let us imagine a task with minimum deadline time from creation d. Let us say the task is created at node n_A to be executed at node n_B. Assume that there are N+1 hops from node n_A to n_B (thus N nodes in between). Each node takes t_{fwd} time to forward a packet. P_A and P_B are processing times at n_A and n_B. Thus, if

$$d > N\overline{t_{fwd}} + \overline{P_A} + \overline{P_B} + (N+1)C \; where \; \overline{y} = max(y) \tag{3.5}$$

where C is some transmission constant per-link. Then hard-real-time guarantees can be provided that the process will be executed. This implies

$$N < \frac{(d - C - \overline{P_A} - \overline{P_B})}{(\overline{t_{fwd}} + C)}. \tag{3.6}$$

This equation is similar to Equation 3.2. It implies that if the deadline characteristics and the other aforesaid upper-bounds are known, one can design a distributed network of longest path N+1 with guaranteed hard real-time processing, where N is the

highest integer value satisfying Equation 3.6. In practice, these upper-bounds are not guaranteed, therefore a hard real-time system cannot be designed. However, can the probability of a deadline miss be computed? If so, a distributed soft real-time system with a known probability of deadline miss can be designed.

The probability density function for the time taken to communicate and process a network packet is a multi-variate N+2 dimensional function, g, given by

$$P(time\ taken(t)) = g(t_{fwd_l}, P_A, P_B) \quad where\ l = 1...N \tag{3.7}$$

At a macroscopic level the probability of the total time taken being less than the deadline, d, needs to be calculated. Since individual times are added to compute the end-to-end time and each of these time variables is independent[7] the probability density function is given by

$$\begin{aligned} p(t = x) \quad = \quad & \int_0^x p_1(t_{fwd} = t_1)... \int_0^x p_i(t_{fwd} = t_i)... \\ & \int_0^x p_N(t_{fwd} = t_N) \int_0^x p(P_A = t_A)... \\ & p(P_B = x - (\sum_{i=1}^{N} t_i + t_A))dt_1...dt_i...dt_N dt_A \end{aligned} \tag{3.8}$$

Thus the cumulative probability is then simply

$$p(t < d_i) \quad = \quad \int_0^{d_i} p(t = x)dx \tag{3.9}$$

Thus if these distributions are known, then the system can estimate the reliability for a given value of deadline or conversely minimum deadline required to give the user some system reliability. This probability distribution for a node is termed as Node Network Forwarding Character (NNFC).

For practical implementation, the algorithm is split in two parts: (1) Scheduling and (2) Estimation. The scheduling part answers the question "Which packet to send first?" while the estimation part answers the query "Is there a point in sending this packet?". The following sections explain this division in detail.

3.3.2 Scheduling

The idea behind the scheduling mechanism is that each packet's IP header is marked at the sender with its associated deadline. Further its TTL field is set to the exact number

[7]The time taken by a router to transfer a packet is independent of the time taken by other routers at that particular time instant. This assumption is relaxed later.

of hops to the destination[8]. When a router receives the packet, it checks how much time the packet has left per-hop to meet its deadline. If this value is below a certain threshold, then this packet is dropped. Otherwise it is insert-sorted in the delivery queue based on this time. More formally:

- Network control packets including time synchronization algorithm packets (such as NTP packets) have highest priority.
- Real-time packets are second. Within the real-time packets queue packets are scheduled as a partial order
  $O = \{p_1, ...p_i, ..p_j, ...p_n\}$ over packets,
  $A = \{1..i, ..j, ..n\}$ such that

$$p_i \leq p_j \Leftrightarrow \frac{(t_{dl_i} - t_{now})}{TTL_i} \leq \frac{(t_{dl_j} - t_{now})}{TTL_j} \; \forall i, j \in P \tag{3.10}$$

- In case of routers running a real-time OS, such as special purpose networks, the deadline, d, before which the next packet must be dequeued is

$$d = f(p_{est}, t_{for}, R) - \frac{(t_{dl} - t_{now})}{TTL} \tag{3.11}$$

where f is an estimate on the required end-to-end time. The function f depends on minimum time required for processing per-hop, p_{est}; the forwarding time of the router in question, t_{for}; and the required reliability for the connection, R[9]

Equations 3.12 and 3.10 implicitly imply that the network needs to be time synchronized. This can be achieved using the existing time synchronization algorithms, the most prominent being NTP [60]. Note that this architecture requires synchronization packets to be marked as control flow packets.

3.3.3 Estimation

The theoretical framework describes a *revolutionary* approach to the future Internet, which requires major router modification and overhaul. In the short term, however, for any new architecture to be successful it needs to present an *evolutionary* path to the future Internet. It is for example difficult to assume that all nodes in the current Internet will be time synchronized. Further, it is *not scalable* to maintain an estimate per-router of all the real-time connections that pass through the router. It also requires further

[8]The connection setup procedure returns the number of hops, see Section 3.3.4

[9]This is a connection oriented approach, which is later evolved to a connection-less approach.

study on how to maintain such estimates in a distributed scalable fashion without serious overhead, such as maintaining an estimate per-destination instead of per-flow. This chapter first presents the theoretical estimation algorithm. Then our solution to the scalability issue is presented. The approach is made scalable by splitting both the estimation part and the network to *congestion estimate* and *0-hop network*, respectively

- During *connection setup phase* a client "estimates" the total delay to its destination node and calculates the number of hops[10]. If the estimate is unacceptable or the connection requesting packet is lost (not acknowledged), the *client application* rejects the application's connection request. This is how connection admittance is performed.
- During *the communication phase*, each node[11] including the client, maintains a minimum acceptable per-hop estimate, p_{est} required to transmit the packet. Each router further adds an estimate of the routers time to process and forward the packet, t_{for} . Packets may be rejected if:

$$\frac{(t_{dl} - t_{now})}{TTL} \leq f(p_{est}, t_{for}, R) \tag{3.12}$$

i.e. it is impossible for the packet to make its deadline given the estimated processing times of the remaining routers.

Dropping packets that are "unlikely" to meet their deadlines is only required when there is congestion at the router. Maintaining the estimates on a free router is unnecessary. Furthermore, when congestion occurs it is important that packets that are less likely to meet their deadlines are dropped first. To solve this issue the estimate, p_{est} from Equation 3.12, is spilt into two different estimates a $congestion\_estimate, p_{con}$ and a $path\_estimate, p_{path}$. In the implementation, presented later only the user client maintains path estimates to each destination. A router maintains the congestion estimate that is a measure of how congested it is itself. This is a scalable approach since routers do not need to know about the destination or flow characteristics of each particular packet. The following subsections explain what the role of each of these estimates is, and how they are calculated in our implementation.

Providing Scalability: The Congestion Estimate

The role of the congestion estimate is in some ways analogous to the window size in the TCP sliding window protocol or the congestion window of the congestion control

[10] How this is done is presented later in Section 3.3.4

[11] Not a scalable approach. Further in Section 3.3.3 will show that instead of every node only the client needs to maintain these estimates.

protocols. If a router experiences congestion it is better to drop packets with shorter deadlines per-hop than random packets as RED [29] does. According to Equation 3.12 this implies that packets with short deadline requirements or long path lengths begin to get dropped before other packets. This is done using the *congestion estimate*, p_{con}. Starting with $p_{con} = 0$, if, congestion occurs (subject to various criteria), then the estimate is increased according to an increment function, f_{incr}. If the congestion has alleviated then the estimate is decremented according to the decrement function, f_{decr}. The combination of various increment and decrement functions and congestion and decongestion criterion were studied by experimentation. The following *linear* functions gave the desirable behavior in terms of the congestion estimate value.

$$\begin{aligned} p_{con_{new}} &= f_{incr}(p_{con_{old}}) = p_{con_{old}} + \delta \\ &\quad iff\ dq_{len}/dt > 0\ and\ q_{len} > .8q_{max} \end{aligned} \tag{3.13}$$

$$\begin{aligned} p_{con_{new}} &= f_{decr}(p_{con_{old}}) = p_{con_{old}} - \delta \\ &\quad iff\ dq_{len}/dt < 0\ and\ p_{con_{old}} > \delta \end{aligned} \tag{3.14}$$

Equation 3.13 implies that p_{con} is incremented, if and only if, the current queue-length is greater than 80% of the total length and is increasing with time. Equation 3.14 implies that p_{con} is decremented, if and only if, the queue length is decreasing.

Thus all packets in a queue whose per-hop deadline is less than the p_{con} will be dropped. This approach is scalable as no connection states are maintained. It is fair as when there are two identical connections, each with the same deadline and the same number of hops, the packets whose per-hop deadline is lesser than p_{con} are dropped for both connections.

Providing Admission Control: The Path Estimate

Connection admittance control is provided using the *path estimate*, p_{path}. In the original theory, each node in the network, for each route in its routing table, maintains a estimate of how long it takes to the destination. If the remaining time to deadline is smaller than the estimate to the packet's destination the packet is dropped, rejecting the connection. Maintaining these estimates at each router, however, is not a scalable approach. As a solution to this problem, a concept of 0-complete networks is created.

The 0-Complete Network

Estserv networks that support the p_{path} estimate up to n-hops from the client, are termed as *n-complete*. Networks *all* of whose nodes support the p_{path} estimate are simply called *Hop-complete Estserv networks*. This distinction makes sense because it

allows various institutions or service providers to implement this on the "outside core" and entry/exit routers. Thus, according to this definition "0-hop complete" networks support the p_{path} estimate only on the clients, i.e., the client maintains all its estimates to all the nodes it makes a real-time connection to. Similarly, "1-hop networks" are networks where every possible one hop from every client node (all access network ERs) in the network supports p_{path} estimation.

0-hop networks are *not* intended as an approximation to hop-complete Estserv networks. Instead they are the current solution to the scalability, economic and backward compatibility issues in the acceptance hop-complete networks. The usage of *0-hop networks* in current solutions is recommended, since they require only the addition of a new module to the current OS' network stacks. The rest of this work only refers to *0-hop complete Estserv networks until otherwise mentioned.* Hop-complete networks can, however, be used in smaller fixed route dedicated networks.

Authors in [26, 5] present the Real-time Channel Admission Protocol (RCAP) which also looked at connection admittance mechanisms for earliest due date based scheduling algorithm in multi-hop networks presented in [27]. The approach was based on estimating the end-to-end deadline using a control protocol and on the way reserving resources at each intermediate router. In doing so they violate the principles of best-effort design for the Internet. This is now known not to be a scalable approach and is the primary difference between this work and the previous approach.

3.3.4 Implementation

The scheduling algorithm presented in Section 3.3.2, including the congestion estimate, was implemented as a module in the Linux kernel. The scheduling algorithm is accessed via the tc (traffic controller) utility. For a detailed explanation of Linux forwarding and QoS concepts please refer [2]. All Linux qdiscs must implement a number of functionalities. The important ones for our explanation are the enqueue and the dequeue function. A detailed explanation of the other functions can be found in [2]. Thus, a FIFO qdisc would *enqueue* at the tail, *dequeue* from the head and *drop* when the queue is full. The Estserv scheduling module's *enqueue* function is implemented as an insertion sort over the calculated per-hop time for each packet. While the *dequeue* function simply dequeues from the head of the queue. The module is inserted using the linux tc utility at the end of the Diffserv-EF PHB, as shown in Figure 3.4. This shows how estserv is an evolution of the current Diffserv architecture. Except for the QDISC field in Figure 3.4 the rest of the setup is exactly the same. The video conferencing application presented in Section 3.1.1 is reused including the bandwidth adaptation mechanism described in Section 3.1.3.

Figure 3.6 shows the packet flow of the system. When a new real-time connection

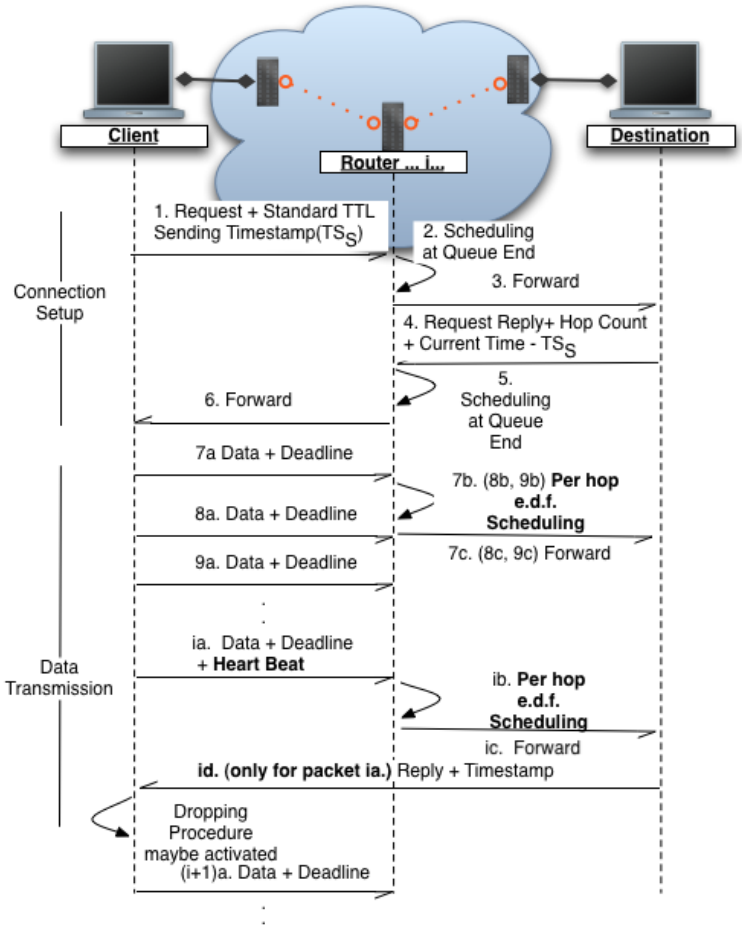


Figure 3.6: Steps of execution: notice that the routers only perform deadline based scheduling i.e. just insertion sort on the EF queue based on deadline.

is required (steps 1-6 in Figure 3.6) the client sends a setup packet to the destination. This packet is marked with the required deadline and a special connection setup request label. The TTL of this packet is set to the IP standard. The intermediate router

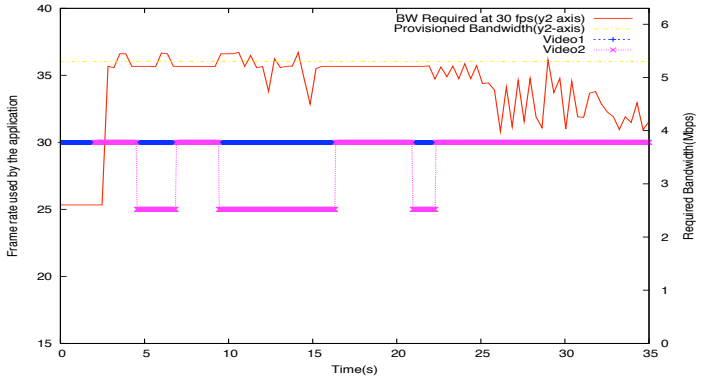
recognizes the label as a connection setup request.

Routers always append this packet at the end of the EF queue (step 2)[12]. If the required deadline has expired or the congestion estimate cannot be met, the request packet is dropped by the router. This gives preference to connections that already exist over newer ones. This implements the *connection admittance*, answering Question 1.3, just by properly scheduling and dropping the request packet. In Figure 3.6 step 4 the destination calculates the difference between the current time and the timestamp on the packet, and the hop count as the difference between standard IP TTL (128/256) and the TTL on the packet. The destination then replies to this packet with this time difference and the hop difference added in data fields. The client on receiving the reply can now calculate the time the packet took to the destination and the number of hops. If this time is less than the required deadline then the client starts sending data packets to the destination (step 7 onwards) with the TTL field set to the number of reported hops. This process is in essence very similar to RSVP [99] used in integrated service architectures. However, unlike RSVP the routers do not need to perform any processing on the packets and no bandwidth reservation is made. The connection admittance is therefore on a best-effort basis to transfer the connection setup request packet.

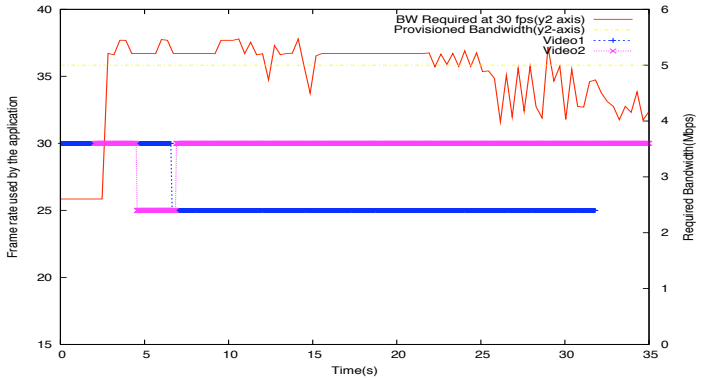
As explained previously in Section 3.1.1 a set of the data packets are marked with a reply label (Figure 3.6, step ia). These packets are acknowledged by the destination in a similar way as the connection setup packet. This packet is transmitted at constant intervals and is always the last packet in that particular frame transmission. These packets, called heartbeat packets, are normal data packets with a special reply label. If the replies to these heartbeat packets are not received by the client then a *connection loss* is said to have taken place. This engages applications adaptation mechanism at the client (explained in Sec 3.1.3). As long as a stable connection cannot be established, the client keeps sending heartbeat packets with exponentially higher intervals between packets. The application is intimated of a lost connection if the replies to the heartbeat packets are not received for H tries. Conversely, if two consecutive heartbeat packets receive replies within a specified time, $(3 * deadline)$, then the application thinks that more bandwidth is now available and moves back up to a better media quality, if not at the highest already.

Note that almost *all actions are performed by the client or destination machines* while the routers only follow a scheduling mechanism based on insertion sort into the forwarding queue. This makes the architecture scalable.

[12]The connection request packet is not scheduled according to the scheduling mechanism presented in Sec 3.3.2

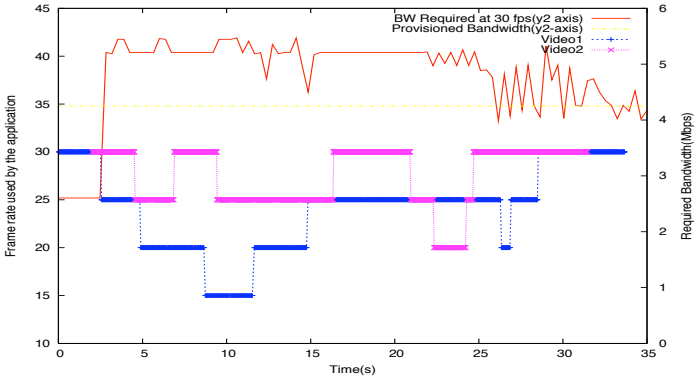


a) Case 1: Bandwidth provision above average
but below maximum requirement = 5.3Mbps



b) Case 2: Bandwidth provisioned at close to average requirement = 5Mbps

Figure 3.7: Results with non-overprovisioned Estserv-EF. Case 3 on next page...



c) Case 3: Bandwidth provisioned at below average requirement = 4.25Mbps

Figure 3.7: ...Continued from previous page: Results with non-overprovisioned Estserv-EF

3.3.5 Results

The experimental setup for looking at the performance of Estserv is exactly the same as the one used for Diffserv-EF as presented in Section 3.1.1. The QDISC in Figure 3.3 is replaced by the estserv qdisc described in the previous section. The three cases of non-overprovisioned network are identical to the three used in the Diffserv-EF experiment vis-a-vis:

- Case (1) $\chi_{BE} = 4.7Mbps, \chi_{EF_{Estserv}} = 5.3Mbps$
- Case (2)$\chi_{BE} = 5Mbps, \chi_{EF_{Estserv}} = 5Mbps$ and
- Case (3) $\chi_{BE} = 5.75Mbps, \chi_{EF_{Estserv}} = 4.25Mbps$

All packet sizes used in the experiment are 1024B thus the EF class has a packet queue length of three.

The results presented in Figure 3.7 and Table 3.3 show a better channel utilization for the EF channel in the Estserv architecture and compared to the standard Diffserv-EF channel in Figure 3.5 and Table 3.2. Thus Estserv provides an answer to question 1.2, however, the question remains if this utilization can be improved. These results

still do not answer the question of scalability and connection admittance, which are presented in the next section. Lastly, these experiments are carried out in a simplified laboratory environment; how do they behave in the real Internet is presented in Section 3.3.7

Table 3.3: Results with Estserv-EF for 2 simultaneous video conferencing connections Video 1 (Total stream size = 12M) and Video 2 (Total stream size = 11M).

	Received sizes for Video 1.	Received sizes for Video 2.	Channel utilization efficiency
Case 1	11M	10M	.91
Case 2	9.7M	11M	.9
Case 3	7.3M	7.8M	.78

3.3.6 Connection Admittance and Scalability

To evaluate connection admittance and scalability the same experiments where repeated with 5 and 10 clients instead of 2. The results are shown only with Case (2) $\chi_{BE} = 2Mbps, \chi_{EF} = 5Mbps$ in Table 3.4 for 5 clients and 3.5 for 10 clients. Each client is started 2 seconds after the previous client.

Table 3.4: Results with Diffserv-EF for 5 simultaneous video conferencing connections Video 1 (Total stream size = 12M) and Video 2 (Total stream size = 11M).
Video 3 (Total stream size = 11M) and Video 4 (Total stream size = 10M).
Video 5 (Total stream size = 9M)

	Received sizes for Video 1, 2, 3, 4, 5	Channel utilization efficiency
Diffserv EF 5 clients	2.1M, 2.1M, 2.3M, 2.5M, 2.6M	.42
Estserv EF 5 clients	9.8M, 5M, 10M, 0, 0	.95

The results show that Diffserv seems to equally divide bandwidth amongst all the connection while the overall efficiency still remains low. This is a waste since conferencing streams cannot work below a certain bandwidth availability. In effect Diffserv results in all the connections being lost. In such cases it is useful to give preference to existing connections over newer incoming ones. As can be seen from the Tables 3.4 and 3.5, the connection admittance mechanism presented in Section 3.2 for Estserv

Table 3.5: Results with Diffserv-EF for 10 simultaneous conferencing connections
Video 1 (Total stream size = 12M) and Video 2 (Total stream size = 11M).
Video 3 (Total stream size = 11M) and Video 4 (Total stream size = 10M).
Video 5(9M), Video 6(9M), Video 7(8M), Video 8(7M), Video 9(7M), Video 10(7M)

	Received sizes for Video 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	Channel utilization efficiency
Diffserv EF 10 clients	1.2M, 1.4M, 1.4M, 1.5M, .8M, 1.2M, 1.6M, .85M, 1.4M, 1.7M	.4
Estserv EF 10 clients	9.7M, 6M, 11M, 0, 0, 0, 0, 0, 0, 0	.8

seems to do this reasonably well. Only the first three connections are accepted in both the cases. This is better since now at least three people can utilize the Internet to some level of satisfaction, whereas Diffserv leads to a congestion collapse. The utilization in Estserv as compared is somewhat low for the 10 client case. This is because, as shown in Figure 3.8, video client 1 ends at $t = 37s$ freeing up bandwidth for other videos. This results in an increase in the frame rate of video client 2. At this point some of this bandwidth remains partially unused resulting in a slightly lower utilization.

With regards to scalability, CPU and Memory usage was measured for machine 2 for the cases of 2 clients, 5 clients and 10 clients; with both Estserv and Diffserv. Machine 2 showed only small fixed amount of increased CPU (2%) and memory usage (100M) from Diffserv, this amount was independent of the number of clients sending packets through the connection. Since this memory and CPU usage is independent of the number of clients it demonstrates that Estserv maintains the constant order scalability of Diffserv.

3.3.7 Backward Compatibility and the Internet.

The design of the future Internet must be evolutionary in nature for economic reasons. It then becomes important to understand how Estserv interacts with the current Internet (Case a) and with Diffserv (Case b).

For Case (a) assume that the access network that acts as the main bottleneck, as in [13]. Then all that is required is to install the edge link to ISP as an Estserv link. Furthermore, time synchronization is only required over this edge, leaving the remainder of the Internet as is. Under this assumption home computers now only need to synchronize to the service provider's gateway router The rest of the Internet can simply ignore the deadline headers on the IP packet. Experiments with the setup shown in Figure 3.9 were performed between access network at CWI Amsterdam and the UIUC

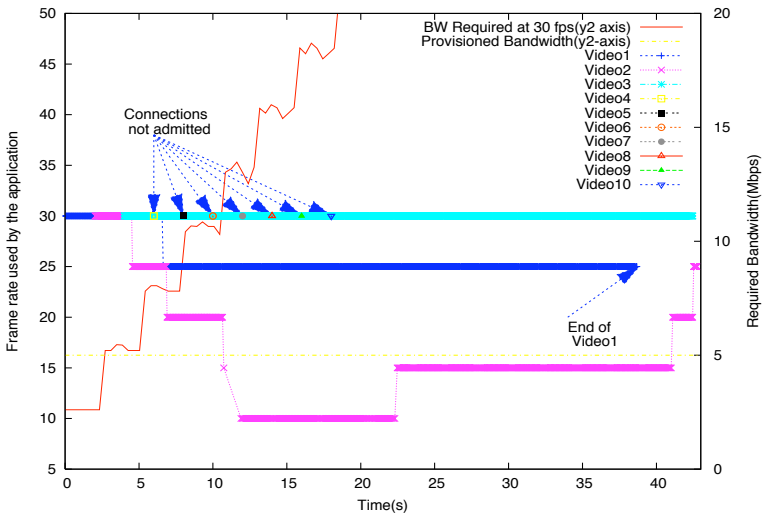


Figure 3.8: Results with 10 clients Estserv demonstrating connection admittance

network with the access router as Diffserv router (Case a.1) and then Estserv router (Case a.2) . The results were very similar to those presented above and demonstrated high utilization for Estserv.

For Case b), the interaction of Estserv with Diffserv, two experiments were performed. The setup is same the as in Figure 3.3. In the first experiment (Case b.1), machine 2 in Figure 3.3 was setup as a Diffserv router and machine 3 as an Estserv router. For the second case (Case b.2) the ordering was reversed. Then all the earlier experiments were repeated, with results in Case (b.1) almost identical to results from experiments with just the Diffserv router, as in Section 3.2.2. In Case (b.2) there was a minor loss in utilization as compared to the results with just the Estserv routers, as in Section 3.3.5. The result are similar enough to not be presented here again.

These experiments demonstrate that Estserv is backward compatible to both the current Internet as well as Diffserv. This eases its adoption in the future.

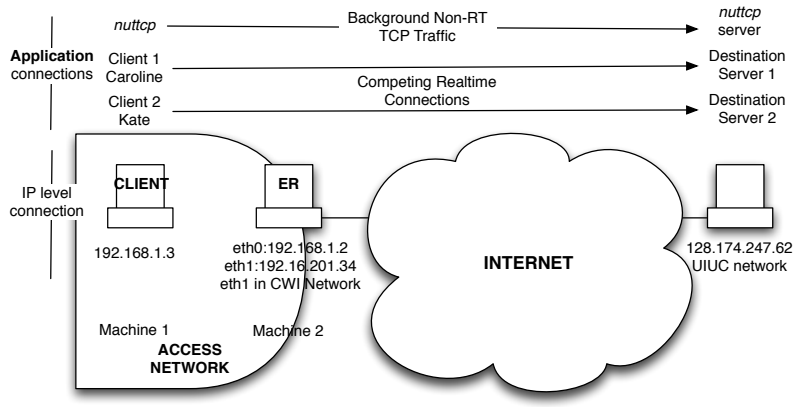


Figure 3.9: Experimental setup with the real Internet

3.3.8 DCCP

In recent work, Datagram Congestion Control Protocol(DCCP) [22] has been recommended for Real time transport. Using the linux *iperf* tool a DCCP connection is set up from machine 1 to machine 4 in Figure 3.3 with a total bandwidth demand of 4 Mbps. Every 2 seconds an additional 2Mbps burst is created that lasts for 2 seconds. Figure 3.10 shows this bandwidth requirement profile. DCCP kernel code is modified to include the estimation algorithm.

Figure 3.10 shows the results. It shows clearly that the saw-tooth behavior of TCP-friendly protocols is eliminated when using Estserv. Instead the Estserv bandwidth profile follows the actual requirement profile exactly subject to the limit imposed by the provisioning. The bandwidth profile while using Diffserv on the other hand oscillates considerably. At every burst a lot of packets are dropped this is followed immediately with a back-off period thereby causing under-utilization.

At the time of writing of this thesis the DCCP implementation in linux is not very reliable. During experiments unforced connection losses often occurred. Further, the implementations of various linux kernel versions were incompatible with each other. Thus, experiments were repeated for three different linux kernel versions with identical results. It was also not possible to set up a DCCP connection via a NAT gateway. Fixing these issues is out of the scope of this work. However, the results shown here along with the results for bandwidth adaptation, suffice to demonstrate the increase in channel

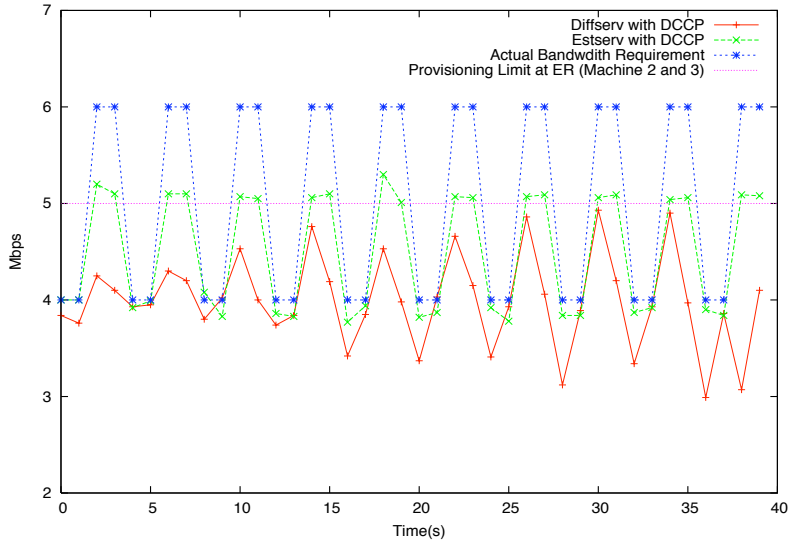


Figure 3.10: Results for Datagram Congestion Control Protocol (DCCP) using CCID2

efficiency with Estserv.

3.3.9 Overhead

This section discusses a peripheral issue of Estserv's design: overhead. The *overhead* associated with Estserv can be divided into two threads of discussion: the extra bytes sent in the header and processing delay at routers.

The *header* consists of a 4 byte label that differentiates between the setup, heartbeat (piggybacked on normal data packets) and normal packets. In addition the deadline *timeval structure* costs 8 bytes. The hop count is contained within the TTL field. In total 12 bytes extra per-packet are sent. This adds a 0.8% bandwidth overhead on 1400 byte packets. This is small compared to the amount of bandwidth utilization it provides.

The *processing delay* encountered at the router is only due to reading the headers. Routers do not have to write back to the packets except for the normal decrementing of

the TTL field. Thus including the hop count in the TTL field avoids any new recalculation of the checksums. Further more routers do not maintain any per-flow state/queues.

3.4 Summary

The goal of this chapter was to evaluate the feasibility of the current Internet QoS standard Differentiated services towards supporting real-time communication over the Internet in the form of communication streams. The Differentiated service per-hop behavior that supports real-time communication is the Expedited Forwarding (EF) class. The Diffserv-EF standard assumes over-provisioning. In practice over-provisioning is not always possible. The first objective of this chapter was to perform actual tests to investigate the performance of the current Diffserv-EF standard over network links that are not overprovisioned. In particular:

> Question 1.1 How efficient is the Diffserv network towards communication streams, such as video conferencing, in networks that are not overprovisioned?

Experiments presented in this chapter demonstrated that Diffserv-EF is inefficient in terms of bandwidth utilization over links that are not overprovisioned. This work defined channel utilization as the ratio of bandwidth utilized successfully to the minimum of the available bandwidth or the required bandwidth. The experiments demonstrated very low channel utilization (<0.4) for non-overprovisioned links. This leads to the next question:

> Question 1.2 How can Diffserv-EF be extended to ensure efficient behavior of the future Internet in links that are not overprovisioned while maintaining scalability provided by the Diffserv architecture?

This chapter then presented an extension to the Diffserv architecture, named Estimated Service, which added deadline based scheduling to the Differentiated service architecture. Experiments performed with this new extension demonstrated high channel utilization (>0.8). However, a high utilization is not sufficient for communication streams. Communication streams cannot function below a certain minimum bandwidth. Thus if the channel is too busy it must reject newer incoming connections in favor of existing ones. This is known as connection admittance. Connection admittance was previously studied as a part of the integrated Services architecture: RSVP. However, RSVP requires each router in the connection path to know every connection's state and other requirements. This is incompatible with the best-effort requirement

within the Diffserv design. The question in particular was:

> Question 1.3 Is it possible to provide connection admittance in the future architecture, while still maintaining scalability provided by the Diffsev architecture?

This chapter provided a scalable connection admittance packet by introducing a path delay estimating mechanism between the client and its destination for every half connection. Each router in the connection path schedules the connection request packet from a client requesting a new connecting at the end of its EF queue. Doing so gives the least priority to newer incoming connection requests over existing ones. Results of experiments performed using this mechanism presented in this chapter show that the first connections were given preference over the later ones. The connections succeeded as long as there was bandwidth available to support them. After which no new connection were established. Furthermore this chapter showed that the estimated service extension is as scalable as the Diffserv mechanism. Lastly it is worth noting that in overprovisioned networks Estimated Service resolves to the standard Diffserv mechanism, thereby not influencing its functioning in properly provisioned networks.

CHAPTER 4

Media Synchronization Over The Internet[1]

In shared synchronous experiences, it is important that all users in the session experience the shared media content together, in a *synchronized* way. Further, these users may have different Internet service providers and varied geographical locations, thus placing them in different network domains. This chapter addresses this issue of synchronized media play out across network domains on the Internet. For coherence, all participants in a *synchronous* shared experience must experience the media content

[1]The work presented in this chapter is a contributed to:

- I. Vaishnavi, , P. Cesar, O. Friedrich, S. Gunkel and D. Geerts. *From IPTV to Synchronous Shared Experiences: Challenges in Design: Distributed Media Synchronization.* Accepted for publication in Elsevier Journal of Signal Processing: Image Communication, 2011.
- D. Geerts, I. Vaishnavi, R. Merkuria, P. Cesar, and O. van Deventer. *Are we in Sync? Synchronization Requirements for Watching Online Video Together.* Accepted for publication at CHI 2011, Vancouver. For this paper the need for user tests, the system design and the implementation of the experiment itself were contributed by this thesis. Other aspects without which the experiments would not be possible, such as designing and conducting the tests, were contributed by experts from KU Leuven.

Work done in this chapter also contributed to IETF draft:

- H. Stokking, M. van Deventer, O. Niamut, F. Walraven, R. van Brandenburg, I. Vaishnavi, F. Boronat, M. Montagud. *RTCP-XR block type for inter-destination media synchronization.* IETF Draft, 2011

together. This implies that the nodes in different service provider networks, even in different domains, need to be able to co-ordinate the play out of the media content. The first question that arises is the quantification of "same time", in particular:

Question 1.4: What levels of distributed play-out synchronization does a distributed media system need to achieve?

The minimum skew in the play out of the media, which the users can detect, is an unknown value. This value depends on a number of factors, such as the type of communication stream, or the media stream content. To investigate this value user tests need to be performed. Performing user tests requires the development and validation of a synchronization algorithm. This leads to the second challenge: to develop a synchronization algorithm that would work over the Internet for synchronous shared experiences. This chapter investigates the possibility of re-utilizing synchronization algorithms used in existing synchronous shared experiences, in particular distributed gaming. Specifically, the second question addressed in this chapter is:

Question 1.5: Can event synchronization, as used in gaming, be extended to synchronize user actions and to achieve distributed media play-out synchronization in synchronous shared experiences?

The main challenge that differentiates implementing a cross-domain media synchronization algorithm from typical distributed media synchronization algorithms is that different users may exist in different service provider domains and access the same content located at different sources. In this cross-domain use case, implementing a distributed synchronization mechanism requires time-synchronization, accurate content timelines, cross-domain media content identification, cross-domain user sessions, and a cross-domain signaling mechanism. To implement the synchronization algorithm, this work utilizes the innovations in distributed synchronization used in gaming and distributed media applications [40, 7]. In particular, the local lag mechanism [59] is implemented over a distributed control signaling architecture [43]. Implementation of this local lag mechanism requires the ability of the nodes to signal their respective playout status to each other. Based on this signaling the play out of the media at other node can be adjusted. These signaling messages are called *synchronization control messages*. The working of the synchronization algorithm is validated using two clients over the Internet, one in Amsterdam and the other in Seoul. Our experiments between Amsterdam and Seoul achieve accuracies to within 500ms.

Given our cross-domain media synchronization algorithm, user tests were performed to answer Question 1.4. Two sets of user tests were performed to investigate user tolerance to synchronization level differences. The first ran over a seven day pe-

riod and tested users in different environments: TV, mobile, and PC. The results of this sets of experiments were not conclusive. The number of factors that influenced the test were far too many to draw concrete conclusions. Learning from the experience of these first tests, a second set of tests was then performed to identify the user tolerance for play-out differences in identical situations, both using a PC. The only parameter that changed was communication stream between audio and text for various levels of synchronization. The detailed setup and results from this set of experiments is presented in Section 4.2.

This chapter uses the work reported in other chapters as building blocks to achieve cross-domain media synchronization. The *synchronization control messages* introduced in this chapter are marked as control packets in the Estimated service architecture (presented in Chapter 3) and are transmitted via the highest priority channel over the Internet, as discussed later in Chapter 7. Proper understanding of these messages requires the client nodes to be time-synchronized within acceptable bounds. Later, Chapter 6 surveys the health of the NTP network and shows that a majority of properly maintained nodes over the Internet are within 100 ms of each other. Given end-to-end deadline guarantees and time-synchronized networks (both within certain bounds), it becomes possible to apply previously developed synchronization schemes to our shared experience architecture. Throughout this chapter social TV [16] is used as a particular case of synchronous shared experience. Social TV involves two or more non-co-located users watching the same TV content (media stream) while communicating with each other using audio, video or text chat (communication stream).

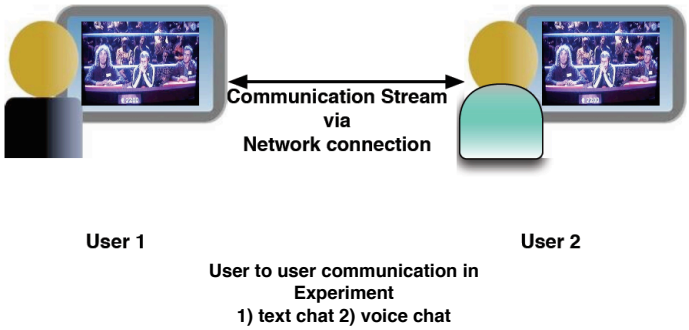


Figure 4.1: User test setup

This chapter is organized as follows. The next section presents the adaptation of the

event synchronization algorithm to achieve media synchronization for the social-TV [16] use case. An adaptation of this algorithm is then used to find the minimum acceptable user tolerance to differences in media stream synchronization levels for social-TV in Section 4.2. Finally, Section 4.2.3 presents the the results of these user tests. A brief summary of this chapter is presented in Section 4.3.

4.1 Media Synchronization

Newer shared experiences over the Internet present a challenge of cross-domain media synchronization. This section describes how synchronization algorithms, typically used in distributed gaming, can be adapted to other synchronous shared experiences over the Internet. In particular, we look at the scenario shown in Figure 4.1, where users are watching a video at two geographically separated locations while communicating with each other synchronously. The videos need not originate from the same source, but must have a temporal relation to each other. A number of simplifying assumptions are enlisted below:

- The communication stream can be extended to send status update messages.
- The rendering devices are capable of running the synchronization algorithm and are powerful enough to process events within time bounds.
- Sessions can be set up across domains. Our implementation utilizes iSessions [34].
- The participating nodes can be time-synchronized within certain bounds.

Assuming time-synchronization within certain bounds, the "Local Lag with Time Warp" [59] algorithm can be adapted to achieve event synchronization. Event synchronization is required between nodes so that when one of the users performs an action this is consistently reflected to all other users, similar to a distributed gaming infrastructure. For example: if the main media stream is paused at one end then the pause should be executed at other nodes, within a bounded tolerance limit. The algorithm consists of two parts: (1) local lag to compensate for short term inconsistencies and (2) time warp to undo inconsistencies that may still occur due to various network delay and jitter factors.

The concept of local lag is illustrated in Figure 4.2. Instead of executing the user action as soon as possible, an event execution time in the future is chosen. This ensures that the event can travel to all destinations and can be executed simultaneously depending on the skew in time-synchronization. This delay value is the local lag. The value of the local lag needs to be small enough so that the user does not notice the delayed execution, but large enough that the event has enough time to travel to other users. In [59] authors recommend a value of 150ms for applications that span the globe. These event updates need to reach their destinations before this specified delay. Since these events

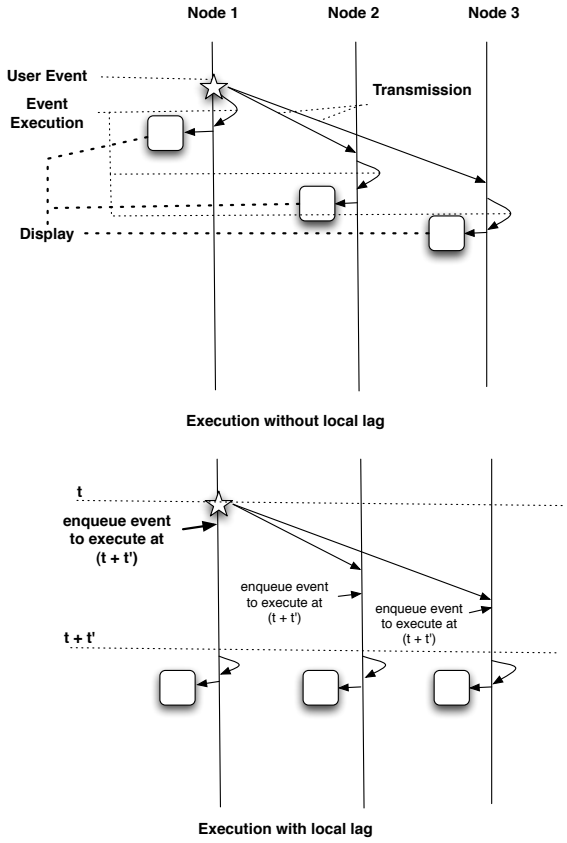


Figure 4.2: Local inconsistencies can be avoided by using local lag

are sporadic in nature and require very little bandwidth they can be sent as network control message that have strict priority on the Internet over other messages.

The local lag technique only attempts to resolve small term inconsistencies; it does not provide any guarantees on consistency. Time warp is the process of rolling back changes to the last known consistent state in case, inconsistencies are detected. Time warping is therefore application dependent and thus not directly addressed here. For our specific scenario of distributed video watching, time warp resolves to jumping back

or forward in the playout time. This, however, implies that the renderer needs to store frames, which have already been played out for a short period of time to facilitate the jumping back.

4.1.1 Signaling Architectures

Previous research in distributed synchronization mechanisms recommends 3 different signaling architectures: (1) master-slave [44] (2) sync-maestro [39] and (3) distributed control [43]. In the chosen scenario there are significant argumentative reasons not to choose sync-maestro. Sync-maestro requires the maintenance of a central sync-server which receives position updates from all the nodes, re-calculates their individual adjustments and signals them to accordingly re-adjust their play out speed or position. This technique requires the maintenance of a dedicated sync server, which is not scalable [41]. Furthermore, it assumes that all nodes can contact this server directly, which may not be the case in cross-domain applications. This architecture is typically used in distributed games to maintain a worldwide view of the game, as a single server greatly simplifies problems relating to causality and replication consistency.

The other two approaches, master-slave and distributed control, are quite similar and more feasible. In the master slave approach one of the nodes is considered a master and multicasts playout update messages to all other nodes in the system, whereas in a distributed control system all nodes can send playout position updates. Each receiving node chooses to follow one of these update packets as the reference timeline. As long as all the nodes choose the same reference timeline the system will remain synchronized. The choice between these two architectures is largely application dependent. For example: in an e-classroom environment the teacher should be the master node which directs the student nodes as to what part of the presentation to play out. In a distributed friends watching football scenario the distributed control scheme is more appropriate, since there is no need for master re-election if the current master drops out.

Lastly a fourth network architecture, which is a hybrid between the sync-maestro and the distributed control approach, can also be used. In this approach a sync-maestro in the local domain collects the status updates from all the end-nodes in that domain and controls their play out. In a cross-domain session, each of the domain sync-maestros further run a distributed control signaling based algorithm amongst themselves. This approach is suitable if a very large number of nodes belong to the same session, such as massively multiplayer online games. Within either of the, master-slave or distributed control architectures, network architectures, the local lag and time warp algorithm [59] can be used to achieve event synchronization. In our validation experiments, presented later in Section 4.1.3, both, master-slave and distributed control, signaling architectures

are implemented. The results from experiments with both these signaling architectures are identical.

4.1.2 Execution

Given event synchronization, media synchronization can be achieved by replacing the event message by playout position updates. The generation of the playout position updates is made periodic. The value of this periodicity depends on the particular application. Since these events are not user generated, the local lag is determined by the period of updates and the particular application semantics, rather than user tolerance to local event execution.

When two or more users join a session to watch media content together, a distributed session needs to be set up. A number of steps are required to achieve media synchronization during the session setup. In particular, a multicast channel needs to be setup and the users need to negotiate on a time-synchronization sources. The multicast channel can be set up re-utilizing the communication stream infrastructure. The control messages of the communication stream can be extended to create a multicast channel. In our implementation XMPP was used to signal user presence and XMPP IQ extended to include position updates.

During execution the synchronization agent at each node acts as an smart event queuing and handling thread. All events are placed in a queue, insertion sorted over execution time. When the execution time of a particular event is reached, the associated event is executed. For periodic position updates in the distributed signaling architecture, all nodes send position updates but only one, the reference node's update needs to be executed. In our implementation the reference node is the node that has the smallest media buffer size.

A common timeline needs to exist amongst the various versions of the content being viewed. For a stored media case, such as VoD, it is the current play-out time into the session. For broadcast video content, the standard [25] describes auxiliary streams that carry this timing information. For more complex presentation structures, such as those which are written in descriptive languages solutions or extensions that maintain state, such as SMIL state [49] can be used.

4.1.3 Implementation and Validation

Two reference implementations were developed. The first one was implemented in Java, using Java Media Framework as the media renderer component. All nodes played the identical video file located at different sources. This implementation was used for testing with clients in Amsterdam and Seoul for a 30-minute video clip. Every 30 seconds each client logged on the current play position, CPT_i, along with the NTP time,

NTP_i, at that moment. Figure 4.3 plots the results of $CPT_{Amsterdam} - CPT_{Seoul} + \{NTP_{seoul} - NTP_{Amsterdam}\}$ for two runs; 1^{st} run: with the master-slave signaling architecture and 2^{nd} run: with distributed control signaling architecture. A total of five repetitions for each signaling architecture were carried out with indistinguishable results. The graph suggests a constant error in play out of around $300ms \pm 100ms$. A further uncertainty in these measurements due to error in NTP measurements is $< \pm 100ms$ resulting in an overall worst case error of $300ms \pm 200ms$. This constant skew value is observed due to constant differences in seeking into the video file at every position update by the renderer. The boundedness of this seek operation was assumed at the start of Section 4.1. Nonetheless this worst case error of 500ms is acceptable as will be shown by user tests presented in Section 4.2.

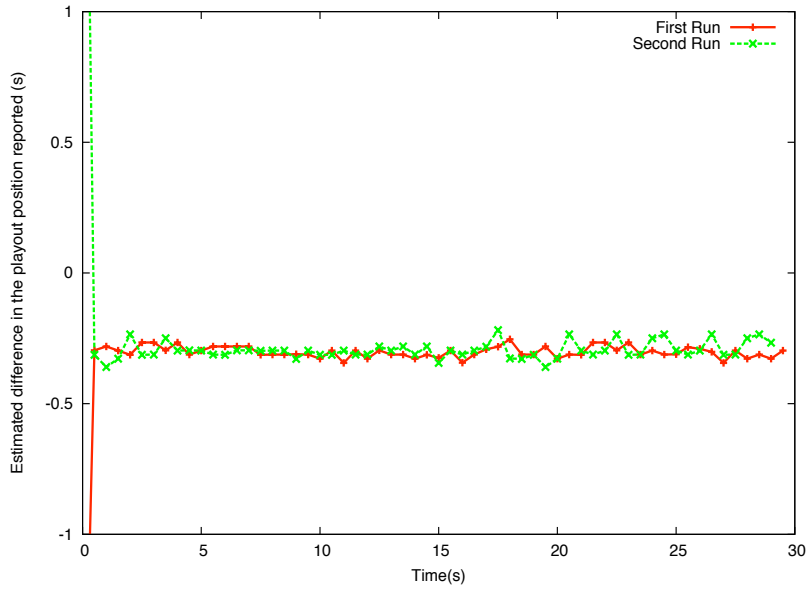


Figure 4.3: Validation results: 1^{st}run) master-slave signaling, 2^{nd}run) distributed control signaling.

A second implementation for three different clients in three different domains was developed. The three clients were a PC, an unmanaged TV and a mobile phone. The PC version was implemented using .NET and with VLCplayer used as the media renderer. A Philips NET TV was used to implement an unmanaged TV case and was coded in javascript and the in built renderer used for media rendering. The mobile phone was implemented using HP's reference implementation of the JSR 309 standard *http://jcp.org/*. This work also lead to the enhancement of the JSR 309 standard. Figure 4.4 shows the screenshots of this implementation.

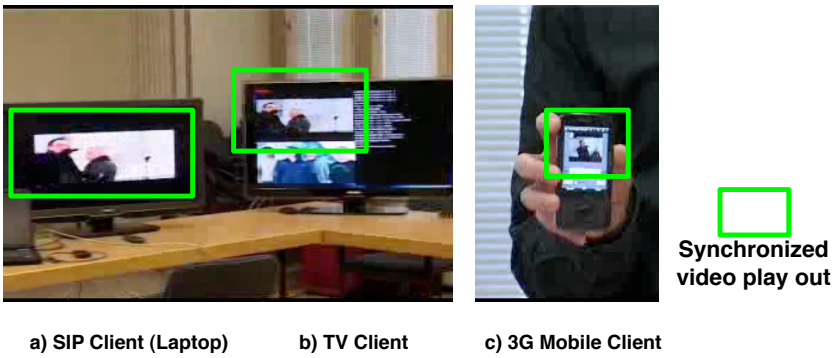


Figure 4.4: Example implementation in the iNEM4u project

4.2 User Perception

An important requirement for synchronous shared experiences is that the media needs to be synchronized in order to have some common ground to talk about [98]. It is, however, theoretically impossible to exactly synchronize media play out over a network. The test results in the previous section shows an estimated error in synchronization in video playout to be around $300ms \pm 200ms$. Is this tolerable to users? Therefore an important research question is to know which synchronization level still enables users to have a satisfying shared experience, impacting the design of future social systems.

Currently, 150ms [46] is used as a rule of thumb, a value drawn from telecommunications research. This rule states that maximum end-to-end, one-way delay when talking remotely should not be over 150ms [46]. Below this value users cannot perceive the delay in communication. Based on this result, one can conclude that 150ms

is the lower bound for synchronizing shared video content. However, no actual user studies have been done to determine the range of acceptable synchronization levels for social video watching. This maybe, in part, because of the large number of parameters that this value may depend on.

This section presents a study to answer this question (Question 1.4). It reports on the results of a user study using the implementation from the previous section. To isolate some of the parameters, the study focuses in users watching a quiz show in a Social TV viewing environment. Based on the user study acceptable synchronization levels for Social TV [16] are determined.

4.2.1 Setup

A within-subjects lab experiment was conducted with 18 couples (partners, friends, or family), so a total of 36 people took part in the tests, consisting of 12 males and 24 females. The age ranged from 15 to 68 years old, reflecting the broad potential target audience of shared video watching.

In each experiment, the two participants were placed in two different observation rooms. Both participants were shown two episodes of a popular local quiz show at different locations simultaneously, with varying amount of play-out skew. The show was chosen because a quiz is a very sociable genre [31]. The content was carefully edited to offer consistent content during the test. During the first episode, participants could voice chat with each other using a headset. The headset was also used for listening to the audio track of the quiz show. Thus, participants could not hear the noise from the audio track of their partners show. During the second episode, participants could only text chat with each other. The order of text chat and voice chat conditions was randomized over the different test sessions, in order to remove any habituation effects.

Without informing the participants, every seven minutes the synchronization level of the videos was changed. This length was chosen in order to allow participants enough time for having a substantial conversation, as well as being able to present several conditions to participants during a two-hour test session. In each condition (voice chat and text chat), five synchronization levels were presented to users: 0 seconds (perfect sync), 500 milliseconds, 1 second, 2 seconds and 4 seconds. These values were chosen during a pilot test with two experts, in which it was discovered that synchronization difference becomes detectable between 500ms and 2s. Based on these expert results, 500ms, 1s, and 2s were taken as test condition, and 0s and 4s were chosen to test the more extreme cases. These levels were presented in a randomized order for each set of participants and each condition. As a difference in synchronization between two participants implies that (the video of) one person is ahead (leading), and one person is behind (lagging), the order of who is leading and who is lagging was also

randomly varied.

After each seven minutes (before the next synchronization change), the participants were asked to fill in a web-based questionnaire, asking a series of questions related to togetherness, noticeability and annoyance of the synchronization differences. In total each participant filled in 10 questionnaires (5 during each condition), resulting in 360 unique measurements. After the first questionnaire it would become clear that synchronization was one of the issues which was questioned. Therefore the participants were instructed in advance to only talk about the content of the show, and not discuss the test itself nor explicitly try to figure out the synchronization difference of the videos.

4.2.2 Measurement Error Bounds

In order to control the play-out skew during the user tests, a system is required that can play media synchronized in two locations and can be manipulated by the observers. A simplified version of the implementation presented in the previous section was used to achieve the chosen level of play-out skew. One of the participants computers was chosen as a master, which continuously sent out position updates to the other computer (the slave). The slave computer received these updates and jumped to the recommended position.

Prior to the tests, this mechanism was validated within the test environment and a margin of error was established for the play-out skew levels. It was found that the error in the skew levels was maximum 150ms with an average of 8ms difference and a standard deviation of 59ms. Thus in this experiment a synchronization level of x implies an interval of $x \pm 0.15$ seconds.

This value is different from the $500ms$ error in validation results for the synchronization algorithm presented in Section 4.1.3, since the both the participant clients were almost identical in hardware and software configuration and were placed across a LAN instead of the Internet.

4.2.3 Results

This section focuses on the results of noticeability and annoyance caused due to play-out skew in the user tests. Results for other surrounding issues, such as togetherness, can be found in [32]. As both, noticeability and annoyance, are closely related, they will be discussed together. To measure the noticeability and annoyance of play-out skew, the Degradation Category Rating (DCR) MOS score as described in [45], used for degraded speech signals, was adapted with values ranging from 1 (not noticeable) to 5 (noticeable and very annoying). Figure 4.5 shows the plot of this rating for both text and voice conditions. In the voice chat condition, synchronization difference becomes

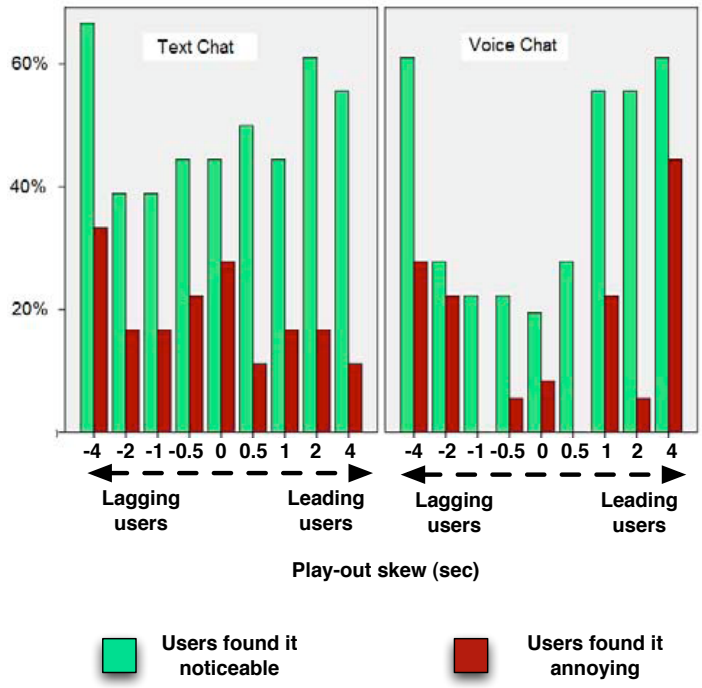


Figure 4.5: Noticeability and annoyance results for varying skew levels

noticeable around 1 second for the leading participant, and around 4 seconds for the lagging participant. This is visible from the strong difference the ratings from 2 to 4 seconds for the lagging participant and from 0.5 to 1 second for the leading participant.

People in the text chat condition give rather random answers, not correlated with the play-out skew, indicating that they do not notice a difference based on synchronization level (play-out skew). This randomness in rating can, probably, be attributed to other factors (such as reaction time of the other participants). However, this difference in between voice chatters and text chatters is mainly attributed to non-active chatters. To test the effect of text chat activity on making the synchronization difference more noticeable, the participants were divided into an active group (more than 400 words per session), with N=15 participants, and a non-active group (less than 400 words per

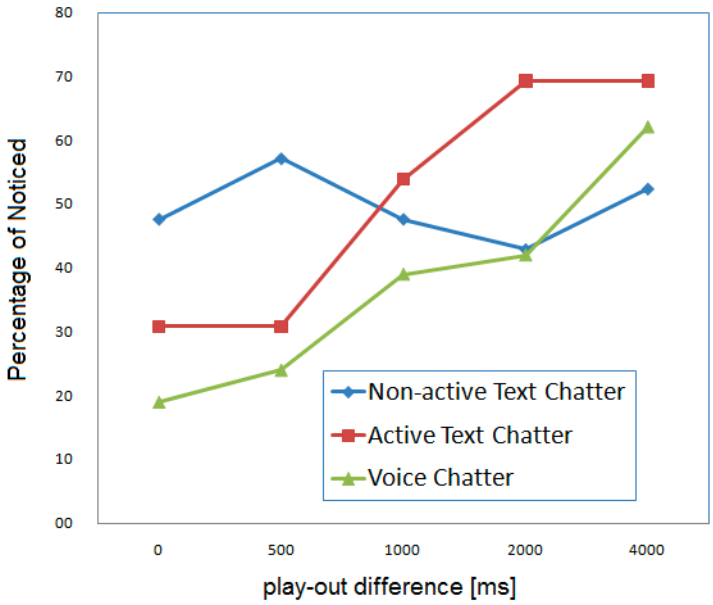


Figure 4.6: Fast typing text chatters notice differences in synch level similar to in the video case

session), with N=21 participants.

In all cases there is a set of users who are perfectly in sync (0s play-out skew, in the middle of both the text and voice chat graphs) but still feel out-of-sync. This is an important aspect of synchronization research, implying that while most user couples feel in-sync in the interval {-1s,1s} there can be other factors that influence their experience. Thus while designers of synchronous shared experiences should aim at a {-1s,1s} interval for play-out skew it is vitally important to provide users with a "skew level adjustment knob". This "knob" can be utilized to adjust play-out skew based on the users' feel of the shared synchronous experience. A point worth mentioning here is that this knob can only be provided for shared experiences where the correct and fair functioning of the experience does not depend on causality of user actions.

4.3 Summary

The goal of this chapter was to address distributed media synchronization in synchronous shared experiences. Distributed media synchronization is required so that users in a synchronous shared experience can coherently communicate about the media stream(s) they experience.

Perfect synchronization is theoretically impossible, which leads to the first question:

> Question 1.4: What levels of distributed play-out synchronization does a distributed media system need to achieve?

This chapter reported on the results of user tests that investigated the tolerance of users to different synchronization levels. A particular case of synchronous shared experiences, Social TV, was used. Five different synchronization levels were used: $\{0, 0.5, 1, 2, 4\}$ seconds. The results demonstrated that a significant number of users noticed the synchronization difference at the 1s mark using audio communication. With text communication the results were not conclusive. However, the results for active text chatters (words per session > 400) were similar to the audio communication case, with a significant percentage of users noticing a difference in play out around the 1-second mark.

The second question addressed in this chapter looked at designing a distributed synchronization mechanism across the Internet. Chapter 1 identified that distributed games were the first popular synchronous shared experiences to exist across the Internet. This previous work in synchronization research can be re-utilized. In particular the question addressed was:

> Question 1.5: Can event synchronization, as used in gaming, be extended to synchronize user actions and to achieve distributed media play-out synchronization in synchronous shared experiences?

This chapter presented our experiences in the adaptation, design and implementation of the event synchronization algorithm used in distributed games. Two reference implementations of this adaptation were created. The execution of the implementation was tested with one client in Amsterdam and a second one in Seoul. The results demonstrated a synchronization skew of $300 \pm 200ms$ between the two locations irrespective of the signaling architectures. User studies presented within this chapter shows that this value is within acceptable limits of user tolerance.

An important result that can be drawn from this chapter is that while most users will not notice a play-out skew below 1s, the perceptibility of the users varies quite a bit.

During user experiment it was observed that this depended on a lot of factors, such as the age difference between the two users, the difference in familiarity with computers and the differences in general levels of responsiveness. Thus, while for developers of shared synchronous experiences it is important to lie within this 1s value of play-out skew, it is also important to expose some level of play-out adjustability. This will help users adjust their play-out speeds, until they feel they are in sync with each other.

CHAPTER 5

User Mobility[1]

During the last decade, users are surrounded by an increasing number of small, powerful and always-connected devices. This connectedness is fueling the desire to stay in touch even while traveling. Thus, synchronous shared experiences over the future Internet will span multiple devices, networks and modalities all surrounding a single moving user. *Coherence* within an environment where the user may constantly move from one network to another, or one device to another, would imply that his synchronous shared experience moves with him. This movement and adaptation of the shared experience with the user is termed *user mobility*. As an example, a user might want to transfer his shared experience from his high-definition television screen at home to his mobile device while leaving his home. This means that the media and the communication

[1]The work presented in this chapter contributed to publications:

- I. Vaishnavi, P. Cesar, A. J. Jansen, B. Gao, and D. C. A. Bulterman. *A presentation layer mechanism for multimedia playback mobility in service oriented architectures.* In ACM conference on Mobile Ubiquitous Multimedia, 2008.
- P. Cesar, I. Vaishnavi, R. Kernchen, S. Meissner, C. Hesselman, M. Boussard, A. Spedalieri, D. C. A. Bulterman, and B. Gao. *Multimedia adaptation in ubiquitous environments: benefits of structured multimedia documents.* In ACM Symposium on Document Engineering, 2008.
- R. Kernchen, S. Meissner, K. Moessner, P. Cesar, I. Vaishnavi, M. Boussard, and C. Hesselman. *Intelligent multimedia presentation in ubiquitous multi-device scenarios.* IEEE Journal on Multimedia, 2010.

stream, which are a part of the shared experience, also need to be transferred in a seamless manner. Further the shared experience needs to adapt to the new user's context. For example, he may only resort to audio communication stream on the mobile phone to save screen space while on the high-definition TV he would want both audio and video.

This implies that shared experiences must move *as well as adapt* with the user. Each device, however, provides its own user interface, rendering and connection capabilities. This requires re-negotiation of the session parameters to adapt to the new target devices and networks. The traditional approach is to consider user mobility as a collection of session mobility requests. This means that each session in the shared experience, i.e. each stream in the media content and each session in the communication stream is independently re-routed and adapted to the new network and the new device's capabilities. Since these adaptations are independent of each other, re-synchronizing them with the other participants is complex, involves identical repetitive signaling in the control plane and is therefore inefficient. The question addressed in this chapter is:

Question 1.6 How user mobility can be achieved in a simpler, scalable and in turn more efficient manner?

Synchronous shared experiences consist of media streams and synchronous communication streams. There is an inherent difference in the requirements of these two streams with respect to session mobility. This is shown in Table 5.1. The difference stems from the fact that communication streams are live, unlike stored media streams. Further unlike broadcast live media streams communication streams cannot be delay rendered. In essence this fruitlessness of re-transmitting communication streams makes session mobility somewhat easier, in that, no state information needs to be stored. As shown in Table 5.1, *user interactions* are limited for communication streams, while for media playback and bufferable live broadcast there are VCR-like controls, hyperlinks, advanced preferences and so on which need to be preserved. Next, while *communication streams* are simple multiplexed audio video streams, media streams can be quite complex with intermedia spatial and temporal relationships. Thus the expression of *media state* becomes more complex than the traditional simple time-stamp into the stream. The media state may now consist of user interactions, preferences and complex intermedia relationships, which need to be preserved for a better user experience.

Temporally efficient transfer is an important requirement in communication, since the user may miss out on content. This, however, can be relaxed for media streams, since the user may pause the presentation. *Transcoding* is the only option for live communication streams because alternative versions do not exist. However, for media streams in service-oriented architectures the same media is usually present with the service provider in multiple formats. These formats may be better suited for the tar-

Table 5.1: A comparison of requirements for transferring user presentations between communication and media stream

	Communication Stream	Media Stream
User interaction with Stream	Limited, almost non existent	Can be complex
Media Complexity	Not complex	May have complex inter-media relationships
Stream state	Inexistent as transmissions are real-time	Can be complex, may include user interactions and intermedia relationships
Temporally efficient transfer	Required	Nice to have
Transcoding	Only option. Other formats do not exist (real-time stream)	Not required, more suitable alternate media encodings may exist
Device discovery and description	Required	Required
Selection of media best suited for the device	Not required, media is transcoded	Required, for enhanced user experience

get device and thus provide a better user experience while also reducing the network overhead due to absence of transcoding. For example, imagine a user transferring a session from his mobile phone to HDTV: no matter how advanced the transcoding function, there is bound to be serious loss of video quality. Both mechanisms require target *device descriptions and discovery mechanisms*. Live applications require deciding which formats to transcode the live stream to, while in playback (in our solution) this is required to select the *map media format* to the target device.

The de facto standard for providing session mobility, SIP-refer call with SDP (Session Description Protocol) description [82], was only intended for communication streams and not for media presentations. The factors enlisted above and in Table 5.1 make it unsuitable for use with shared experiences, which are in essence an integration of media presentations with communication. Extensions to incorporate complex presentation-state semantics and dynamic content selection into SDP can be made. SDP, however, is a session level description protocol and a mechanism would be re-

quired to communicate media relationships typically expressed at the higher layers. Currently, however, the only way of achieving user mobility using the SIP-SDP mechanism would be to integrate multiple identical session mobility requests, for each media item in the presentation. This is inefficient in terms of signaling and does not utilize the known intermedia relationships.

The solution presented here is to consider the entire shared experience as one presentation. This presentation can be expressed in languages, such as SMIL[2] and NCL [86]. When a user mobility request is received, the current state of the presentation is saved in same presentation description languages using solutions, such as [49] . When a rejoin request is received the presentation is re-started at the target network. This restart means that presentation is automatically adapted to the new target network. The state file is fetched and this state is re-applied to the current presentation restoring synchronous view of the shared experience.

This chapter presents this procedure in detail, in particular: (1) How is the presentation adapted to the new device? (2) How can the state of the current presentation be restored at the new target device? (3) What are the network level issues, if any, in doing so? It shows that this procedure is significantly simpler requiring much less signaling overhead and therefore much more scalable and efficient, and can provide better user experience.

This chapter is organized as follows. The next section presents the assumptions used throughout this chapter, followed by the presentation of the architecture itself in Section 5.1.1. *Presentation-layer continuity* as a solution to the above mentioned problem is presented in Section 5.2. Section 5.3 presents the reference implementation of the presentation-layer continuity mechanism and discusses the results thereof. Section 5.4 presents a overall concluding summary of this chapter and possible directions of future work.

5.1 Assumptions

This section presents the assumptions made on the service provider network. These are:

- The service provider network provides multimedia services for a range of device capabilities from HDTVs to mobile devices.
- For doing so it possesses pre-encoded media formats in terms of size and encoding already available at the media server targeted to specific devices. Thus, the same presentation may be present on the server in different coding formats, like the mobile phone version and the cable TV version.

[2] http://www.w3.org/AudioVideo

- The service provider network has a mechanism of discovering and appropriately describing the devices it serves. Methods of doing this are presented in a previously in [12].

5.1.1 Architecture

Based on the assumptions, this section considers the conceived architecture. Figure 5.1 shows the overall generic architecture of the system with the relevant components highlighted. The main focus in this chapter is on the mobility manager's functionality and how it interacts with the other related components in the generic shared experience architecture, Chapter 2.

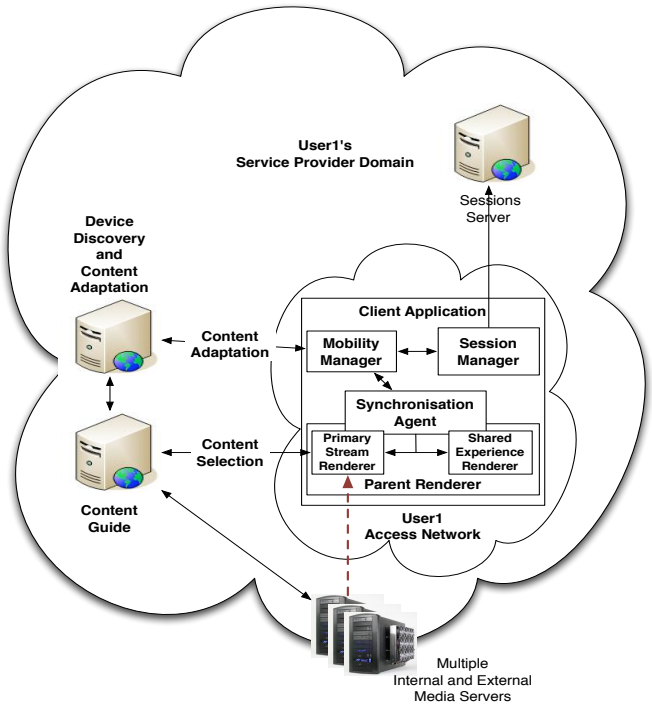


Figure 5.1: Our architecture

The *device to content adaptation* runs in the back end and maps the devices acces-

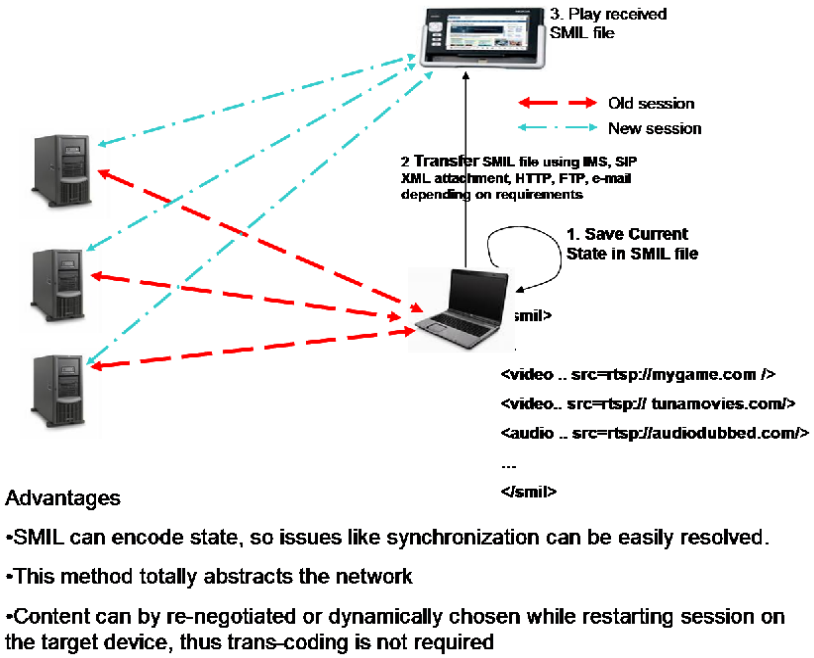


Figure 5.2: An algorithm for session transfer.

sible to the user to the appropriate media content format available at the media server. This mapping considers multiple factors, such as screen size, resolution, supported formats and so on. An in depth explanation of its working and the algorithms used is shown in [51].

The *content guide server* abstracts the raw media stored in the media servers into higher-level authored presentations, which can be enlisted and played out in the media stream renderer.

The *mobility manager* is invoked by the *session manager* or the *parent renderer* depending on whether the system decides to *push* the media to a new location based on changes in user context or whether the user *pulls* the media at his new destination.

5.2 Presentation-Layer Continuity

One of the key issues to transfer a session from one device to another is to keep a consistent and comprehensive description of the current state of the presentation. Previous work on this topic focused on solutions for atemporal documents [87, 72]. These solutions store a detailed representation of the information of the web browser, such as the current site being viewed and the user interactions. The broad steps to transfer state in the web, as presented in [87, 72], are:

- Pause the current presentation
- Store the current state of the presentation
- Adapt the content to the new device
- Transfer of the content to the new device(s)
- Restore the state in the new device

A solution along similar lines is to keep the current detailed state of the presentation (at the time the transfer is requested), as shown in Section 5.2.1, to stop the current session, and to start a new session that includes the media stream better suited for the new device in accordance with the saved state information. This solution requires an abstract representation of the multimedia service, so the system can conclude that the new content stream is indeed the same as the previous content stream. This state may then be transferred by appropriate means to the target device as shown in Figure 5.2.

So far an overview of how this mechanism would work has been provided. The questions that were raised at the beginning still need to be answered. Each of those questions will be answered in the following subsections in detail. Before going into details of each one of these steps a brief overview of a typical service oriented architecture is presented. This will put the terms used in the rest of the chapter into context. Section 5.2.1 shows with simple examples how the presentation is adapted to the selected device, whilst abstracting out the network details. Section 5.2.2 then talks about the network level messaging whilst abstracting out the presentation layer details. This chapter then looks at some limitations and presents the results obtained from implementation and compare them with previous approaches.

5.2.1 Presentation Adaptation Mechanism

This section discusses the mechanism of adapting the presentation while transferring it from one device to another. Each available encoding on the media server is compared to device specifications of each of the rendering devices accessible to the user. Based on the user profile and other heuristics a media encoding best matched to each device is selected. This mechanism of mapping appropriate media to devices is presented in

[51]. To transmit the presentation from one device to another a high-level description of the presentation is used, with the SMIL[3] standard, in which different media elements are temporally and spatially synchronized. In addition, SMIL state functionality is used in order to save and restore the current status of the presentation and thus making the transfer as smooth as possible. In [12] the syntactical and semantical issues relevant to presentation encoding are presented in detail. The following is a description of the issues important for this thesis.

Source Code 1 shows a presentation composed of a video and of subtitles. The *seq* tags mean that their child nodes must be played sequentially, while *par* stands for parallel. In this case, the presentation is intended to the high-definition television display. The video file comes to the home using the cable connection, while the subtitles are coming from an Internet subtitles provider because the mother tongue of the user is not a common one. The home server at home is in charge of synchronizing the media elements based on the given description. At some point the user decides to transfer the

Source Code 1 Original Presentation Description.

```
3 <layout>
4 <region id="v" width="1024" height="800"/>
5 <region id="s" width="200" height="220"/>
6 </layout>
...
09 <seq>
10  <par>
11   <video region="v" src=".../video.mov"/>
12   <text region="s" src=".../sub.rt"/>
13  </par>
14 </seq>
```

presentation from the television display to his mobile device. At that point, our system follows the next steps: (1) store the current state, (2) adapt the media content to the new platform, (3) transfer the modified presentation to the new device, and (4) restore the presentation in the new platform. Source Code 2 shows how the result of modifying the state of the presentation. If the presentation transfer takes place after 30 seconds of watching the video, a new presentation is generated in which the *clipBegin* and *clipEnd* attributes are added. These attributes point to the time offset into their respective media items for beginning and ending the play-out, respectively. In addition, a state variable indicating that the first *par* has been already displayed is added[4].

Finally, the content needs to be adapted to the new device; this includes the layout description as well. For example, the platform might decide to deliver another version of the same video with the saved state information that has been professionally

[3] www.w3.org/AudioVideo/

[4] This is shown for clarity. Actual SMIL syntax is slightly different.

Source Code 2 State Modification.

```
09 <seq>
10  <par state="disregard">
11   <video region="v" src=".../video.mov" clipEnd="30s"/>
12   <text region="s" src=".../sub.rt" clipEnd="30s"/>
13  </par>
24  <par >
25   <video region="v" src=".../video.mov" clipBegin="30s"/>
26   <text region="s" src=".../sub.rt" clipBegin="30s"/>
27  </par>
28 </seq>
```

produced for the specific new device in terms of video resolution and encoding. Furthermore based on the current state of the presentation while transferring the session it will re-contact the subtitling service on the Internet and fetch the appropriate subtitles. All network and service provider related negotiations are performed at the back end hidden to the user's application layer. Source Code 3 shows the result of such modification. A more in depth process of choosing the right content-device adaptations that is executed at the back-end is presented in [51].

Source Code 3 Content Adapt. and State Modification.

```
4 <region id="v" width="176" height="220"/>
5 <region id="s" width="200" height="80"/>
...
24  <par>
25   <video region="v" src=".../video.3gp" clipBegin="30s"/>
26   <text region="s" src=".../sub_small.rt" clipBegin="30s"/>
27  </par>
28 </seq>
```

Synchronization is now achieved as the target device(s) re-negotiates the content from the respective sources. In case the presentation is split across multiple devices then a distributed synchronization algorithm, such as the one presented in [95] may be used. For more details on how SMIL-State can be used to store complex presentation data please refer to [49].

5.2.2 Network Details

This chapter has so far presented how the state of the shared experience can be saved, adapted and applied using SMIL as an example. This way cleanly separates the presentation level details from the network level details. Such a separation allows the presentation level to work independent of the network layer and vice versa. The same can not be said for the case in [82] where SDP is "used" also to store presentation states

and device descriptions. Firstly, as stated earlier, presentation state is non-existent in conferencing streams thus the SIP-SDP based approach works fine, however, extending that approach to media playback is not optimal. Secondly, abstracting presentation and transport heuristics implies that the system can be designed to function with any type of network, Wi-Fi, Bluetooth or IMS infrastructures across the Internet. Lastly, this modular approach adheres to the increasingly popular service provider model. Different servers on the Internet can provide different services, such as a server for discovery another for media sources another for adaptation and so on, thus retaining the scalability for the future Internet.

This section will talk about how the transport level must behave to facilitate this transfer. Both the network models of presentation continuity, namely; the *push* and the *pull* model, are presented. Both these models are well understood in computer networks. The two models, which address two different use cases, are presented separately for clarity. The network level at which this exchange takes place is obscured by intention. This exchange can be implemented in the application layer or as an automatic transfer protocol in the transport layer. The choice is left to the particular implementations. All the presentation layer user mobility mechanisms are independent of the technologies used to achieve this transfer.

Assume for the purposes of this discussion that the user initiates a media presentation session on a device A. This is done in the architecture presented before using a content guide, similar in functionality to electronic programming guides. The selected presentation is then adapted to the device the user is currently using, with the help of the adaptation engine. Since the aim of this chapter is to focus on the session transfer part initialization details are omitted.

Push Model

In a ubiquitous setting, the service must be aware of the different devices surrounding the user and possess the device descriptions for each of them. Assuming there is a way to determine which device the user is currently using, the service may automatically initiate presentation transfers from one device to another following the user. In this method the service pushes the presentation to the new target device. The network interactions between the service and the devices are shown in Figure 5.3.

A change in the device being used, as shown in the figure, initiates an interrupt call informing the service that the user has shifted from device A to B (Step 1). After authorizing the user on device B and obtaining it description (Step 2, 3), the service asks device A to stop rendering the current session and to save the current state of the presentation as a SMIL file (4). Device A replies to the service with this SMIL file attached (5A). The service then asks the adaptation engine to re-adapt the presentation (6), while preserving its state, with appropriate media encoding and screen size, pro-

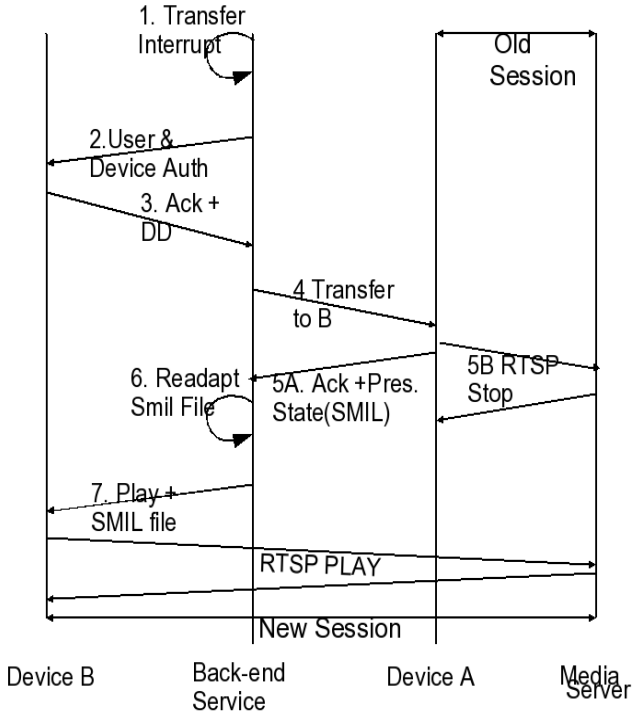


Figure 5.3: Messages while pushing session. DD stands for device description.

viding it with the current presentation state and B's device description. On receiving the new presentation file from the adaptation engine the service sends this SMIL file to B (7). B then renders this presentation. Meanwhile, device A may stop its presentation (5B).

Pull Model

In a more mobile setting, and in the absence of automated device discovery and user detection mechanisms, the user may be provided with an interface to facilitate transferring the presentation from device A to B. The network details for this are shown in Figure 5.4. Here the user voluntarily pulls the presentation to his device.

As shown, the user logs into the service on his target device (Step 1). The service

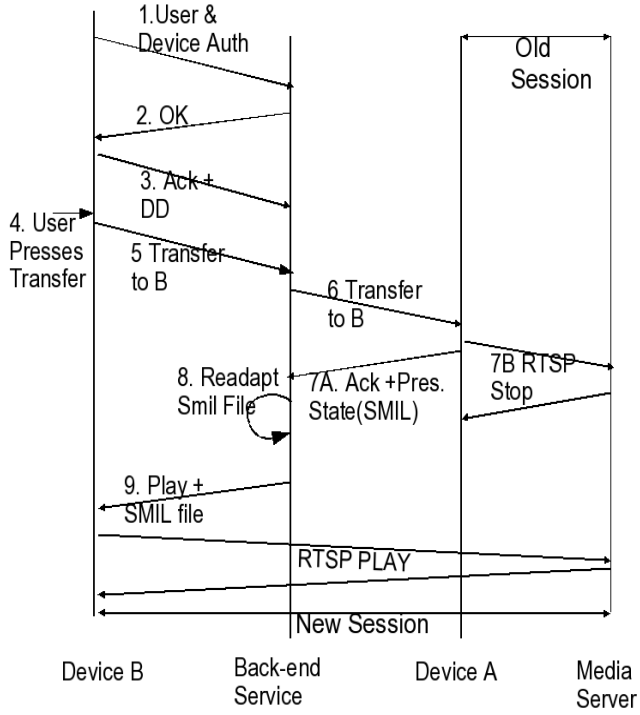


Figure 5.4: Network messages while pulling session. DD stands for device description.

after authorizing him presents him with information about his presentation running on device A (2). The user then asks the service to pull the session to his new device (4). This results in B sending a transfer request to the service with its device description (5). The service then asks device A to close its presentation while sending back the current presentation state (6, 7A). The service then adapts the presentation using the adaptation engine to device B as in the previous subsection (8). This new presentation is sent to B for rendering (9).

Note: In both the cases it was assumed that an adaptation mechanism exists and the service is aware of it. This, however is not a requirement. The service may simply send the presentation state it receives from A to B and then B may re-negotiate the appropriate media content with each individual media source. The advantage of doing this is that then the service need not be aware of the alternative media-encoding present

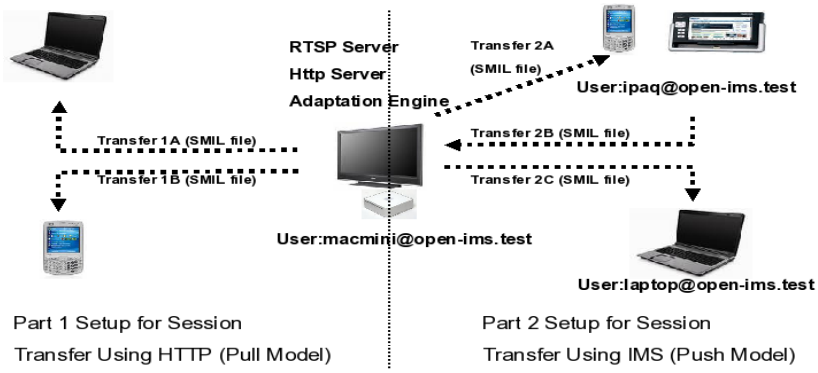


Figure 5.5: Actual experimental setup

at each media server.

5.2.3 Limitations

Even though the solution presented in this chapter is versatile and powerful, a couple of limitations in this approach can be identified, namely:

- This method cannot provide continuous stream transfer. The presentation needs to be stopped and re-negotiated at the new destination.
- Transferring sessions across service provider networks requires that the same media sources/servers are accessible in the other network. Chapter 4 and [34] show how universal sessions including unique universal content identifiers can be provided by service providers.

5.3 Implementation and Results

In the first experiment, an HTTP server running on a Mac mini (with a HDTV attached) was used to fetch the SMIL file to the user's device. This is shown in Figure 5.5 part 1. Thus the user *pulled* the session to a device he already possesses. To do so the user accesses a pre-configured web page, using the target device, where he fetches the SMIL file with the current state of the presentation. This SMIL file is then launched on a compatible player on his device. On all devices a version of the Ambulant[5] media

[5] www.cwi.nl/projects/Ambulant/distPlayer.html

Table 5.2: Devices used in experiments

Experiment	Source Device	Target Device
1A	Mac mini	Portege m400
1B	Mac mini	HP Ipaq
2A	Mac mini	HP Ipaq
2B	Nokia 770	Mac mini
2C	Mac mini	Portege m400

player with integrated IMS client (for Part 2) called minisip[6] is installed. Dynamic adaption of the presentation is performed at the webserver. Sessions are transferred from the Mac mini screen shown in the figure to the HP IPAQ mobile device and to another laptop, as presented in Table 5.2. Figure 5.6 shows an actual screen shot of this session transfer.

In the second experiment our architecture was adapted to the IP Multimedia Systems (IMS). When the user transfers his current session from a host device to a target device, the host device saves the current state of the presentation in a SMIL file. The SMIL file is then *pushed* to these new renderers as an instant message using IMS. The IMS server is pre-configured to intercept this presentation and then adapt it to the target device. The new device then takes this SMIL file and plays it. This setup is shown in Figure 5.5 part 2 and explained in Table 5.2. A presentation was started on device macmini@open-ims.test fetching RTSP streams from an RTSP server (not in the diagram). A transfer request was then sent to this device to transfer the session to device laptop@open-ims.test. This initiates an IMS message to "laptop" with the request to transfer the corresponding SMIL file with the appropriate state information. "Laptop" then simply has to render this file and send an acknowledgement to " Mac mini". Similar steps were carried out with ipaq@open-ims.test, the HP IPAQ mobile device, and are shown in the figure.

Another advantage of the presentation layer mechanism is that partial transfers where different component streams are transferred to different user devices can also be achieved. An experiment in which only one of the component streams of the presentation is transferred to another device, while the second continues playing on the originating device was also performed. The time taken to split and transfer the presentation was similar to the time taken in each of the individual cases. The distributed synchronization algorithm presented in Chapter 4 is then used to synchronize not just between users but also between the various component streams of the shared experience at the same user's end.

[6]www.minisip.org

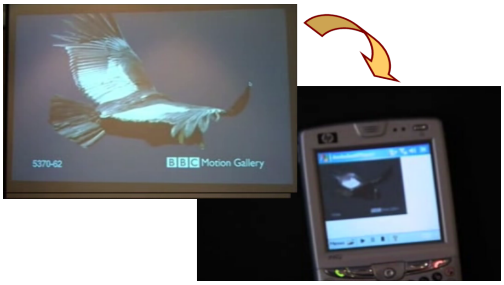


Figure 5.6: Screen shot of presentation transfer from a television screen to a mobile device.

Table 5.3: Max. time (*in ms*) taken to transfer presentation (50 runs). Experiment 1

Max Time taken (over 50 runs) to	Exp 1A	Exp 1B
Save state and Dwnld. file after adaptation	50	100
Restart RTSP	700	1200
Total	750	1300

Table 5.4: Max. time (*in ms*) taken to transfer presentation (50 runs). Experiment 2

Max Time taken (over 50 runs) to	Exp 2A	Exp 2B	2C
Save state and IMS transfer	50	50	50
Restart RTSP	1200	650	700
Total	1250	700	750

The results in Table 5.3 and 5.4 show the time taken during various stages of the presentation transfer for Experiment 1 and 2 respectively. It can be seen that the amount of time it takes is dependent on the processing capabilities of the device. The network time to transfer the message from one device to another was too small to measure since a dedicated wireless network without any other network traffic was used.

Assuming the results in [82, 52] provide acceptable time transfer limits, the time taken here to transfer presentations is acceptable, validating our method. The results are intended only to demonstrate the ability to acceptably implement this method and

are not intended to show a faster or a more efficient method.

Apart from signifying the differences in requirements in live and media playback mode, the true targets of the chapter were achieved as

- Control plane calls are simply a request to the target devices to play the attached SMIL file (with last previous state information)
- Trans-coding need not be performed since the target device now starts a new session with the servers, thus re-negotiating the content. Alternatively, a central server may match the content to device.
- The player (maybe distributed) is now responsible for ensuring media consistency in the new target device
- additional content maybe stored at the media servers, which maybe better suited to the target devices or target context, thereby enhancing user experience.

An added advantage of our approach from a user-centric view is that the user may choose to start the presentation on the target device as per his convenience.

5.4 Summary

The goal of this chapter was to design an efficient user mobility mechanism for coherence in synchronous shared experiences in the Future Internet. Given that synchronous shared experiences in the future will be composed of complex multimedia presentation involving a composition of multiple media components, the current standards achieve user mobility by breaking it down into a number of composing session mobility requests. This involves repetitive identical signaling which is inefficient in the control plane. In particular the question that was addressed was:

Question 1.6 *How user mobility can be achieved in a simpler, scalable and in turn more efficient manner?*

As an answer this chapter presented a *presentation layer user mobility* mechanism. This mechanism abstracts the individual media sessions, thereby making control plane signaling more efficient. The mechanism essentially works by saving the state of the current presentation prior to mobility, restarting the presentation at the new location in a context-aware manner and then re-applying the stored state. The main advantages of doing this were that the new presentation is automatically adapted to the new user context since it is initiated there and elimination of redundant signaling in the control plane.

The chapter presented the details of the mobility mechanism with both *push* and *pull* control mechanism. Further, two scenarios of session transfer, each using different underlying infrastructures, were implemented to show the underlying network independence. One scenario utilizes the IMS subsystem and the other uses HTTP. The work presented here is general enough to be plugged into any future form of synchronous shared experience, provided there exists a method to save and re-apply states for that shared experience. In the future, however, a standard needs to evolve for interfaces, signaling and varied messaging formats. These standards must also consider user preferences and context.

CHAPTER 6

Time Synchronization: Accuracy in the NTP Network

This chapter examines the assumption of time synchronization made throughout this thesis. In its ideal form Estimated service, presented in Chapter 3, assumes that it is possible to synchronize clocks within tens of milliseconds. Further, distributed media synchronization and presentation layer mobility for synchronized shared experiences, presented in Chapter 4 and 5 respectively, assume the existence of time synchronized clocks typically to hundreds of milliseconds. The question then arises:

Question 1.7 *What are the bounds on time difference that can be assumed between any two nodes in the NTP network?*

This chapter surveys the current NTP network using a technique known as "spidering". This technique has been used in previous works that surveyed the NTP network in 1989 [61], 1999 [63] and more recently in 2005 [68]. This chapter presents a comparison with the results from these previous works. The results of the present survey demonstrate that in general NTP performs well over the Internet and its performance is steadily improving. Almost 90% of the surveyed nodes are within 100ms of each other. Further, the survey shows 90% of the nodes estimate that they are within 10ms of their synchronization peers. This chapter assumes basic familiarity with NTP. For the purposes of this chapter, it is assumed that all nodes in the synchronous shared ex-

perience can access NTP servers. In case this is not possible or the respective use case desires a more ad hoc setup, Appendix B presents a solution for time synchronization, called neighbourCast.

The next section presents the technique of spidering the NTP network. This is followed by Section 6.2 that explains the implementation of this technique used in this work. Section 6.3 then presents the results in terms of *offset* and *dispersion* found in the surveyed network. The results are presented in comparison with the results from previous works. Finally, the answer to the question raised in the beginning of this chapter is summarized in Section 6.4.

6.1 Spidering

Spidering the NTP network consists of two steps: a breadth first search for NTP nodes and inquiring the time synchronization status of each of the NTP nodes. The method used to search for nodes is known as *walking the tree*, first presented in [61]. Similar to breadth first search in graphs, walking the tree starts from a known NTP node. It then creates a list of NTP nodes by inquiring the IP addresses of the other NTP nodes connected to the known NTP node. The connected nodes are the parent node, child nodes, peer nodes, and the list of nodes monitoring the known node. Please refer to [60] for their definitions. It then recursively asks the nodes in the newly created list the IP addresses of their parent, child, peer and monitoring nodes. This process discovers all the connected nodes in the NTP tree. This is done using two standard $ntpdc$ queries shown in Command Execution 1 and 2.

Command Execution 1 Inquiring about the peer nodes for ntp.uiuc.edu.

```
[root@localhost ~]# ntpdc -pn ntp.uiuc.edu
     remote           local      st poll reach  delay   offset    disp
==============================================================================
+130.126.24.53   130.126.24.44    2 1024  377 0.00040 -0.000762 0.15822
+128.255.32.25   130.126.24.44    2  512  176 0.00764 -0.000308 0.12134
=127.127.1.0     127.0.0.1       13   64  377 0.00000  0.000000 0.03064
*128.174.38.133  130.126.24.44    1 1024  377 0.00044 -0.000031 0.12175
=192.5.41.40     130.126.24.44    1 1024  377 0.04317  0.003274 0.13667
+130.126.24.24   130.126.24.44    2 1024  376 0.00032  0.000321 0.18315
```

Command execution 1 shows the list of peers to the NTP node ntp.uiuc.edu. Of these the peer node beginning with the "*", in this case 128.174.38.133, is chosen as the synchronization peer (or the parent node). This is the node ntp.uiuc.edu is synchronizing to. The command also enlists other hosts that are being monitored by ntp.uiuc.edu as potential fallback synchronization peers. Command execution 2 gives the list of

Command Execution 2 Inquiring about the list of nodes monitoring ntp.uiuc.edu.

```
[root@localhost ~]# ntpdc -n -c monlist ntp.uiuc.edu
remote address          port local address      count m ver code avgint  lstint
===============================================================================
76.224.127.173         61153 130.126.24.53      62849 3 1    590     31       0
32.106.77.48             123 130.126.24.53          2 3 1    590     64       0
12.110.169.7             423 130.126.24.53     385251 3 2    590      1       0
222.106.135.33           123 130.126.24.53          5 3 1    590     64       0
208.4.155.36             406 130.126.24.53     303036 1 2    590      4       0
98.214.170.136            97 130.126.24.53         25 3 1    590     64       0
69.28.122.206             12 130.126.24.53       3720 3 4    590     14       0
58.10.64.234             123 130.126.24.53          9 3 1    590     64       0
<...590 more similar lines>
```

NTP clients that are currently monitoring or have previously attempted to synchronize with "ntp.uiuc.edu". Using these two commands, a further lists of nodes in the NTP network is found. These commands are then recursively applied to these nodes to eventually generate a complete connected list of NTP nodes.

Command Execution 3

```
[root@localhost ~]# ntpdc -c sysinfo ntp.uiuc.edu
system peer:          truechimer.cites.illinois.edu
system peer mode:     sym_active
leap indicator:       00
stratum:              2
precision:            -20
root distance:        0.00043 s
root dispersion:      0.01720 s
reference ID:         [128.174.38.133]
reference time:       d0636c34.dd620170  Fri, Oct 15 2010 20:11:32.864
system flags:         auth monitor ntp kernel stats
jitter:               0.000854 s
stability:            0.000 ppm
broadcastdelay:       0.003998 s
authdelay:            0.000001 s
```

For each node in this list a second command is used to inquire about the NTP related statistics of the node. This is done using Command Execution 3. The values reported by this command can be used to understand various factors, such as: the NTP network topology, distance from root, synchronization errors jitter values. The purpose of this study is to focus on the precision of reporting time in the NTP network. Thus this chapter focuses on two values directly related to determining this precision: offset and dispersion. The results for these values are presented in Section 6.3. The next section first presents a brief description of the implementation.

6.2 Implementation

A C++ application is developed that uses the system commands described in the previous section to survey the NTP network. There are 2 threads in the application, the first one "walks the tree" generating a list of IP address in the NTP network. This thread begins a breadth first search of the nodes in the NTP network, starting with an initial list of level 1[1] and level 2[2] NTP servers enlisted on Wikipedia. This way all major known NTP networks can be covered. The second thread gathers the relevant NTP statistics from the discovered nodes. The program ran on a Linux system possessing the IP address 192.16.204.88 for about 2 months (Mar, Apr) in 2010. During its search it found 1033295 unique IP addresses. However, in a stark difference from the earlier works only 64682 replied to the $info$ request. The next section identifies possible reasons for this drop and presents results from this experiment.

6.3 Results

Table 6.1 shows a comparison of the number of hosts found and the number of replies received with respect to this and previous studies. There is a minor drop in the number of hosts found but a significant drop in number of replies received with respect to the last study in 2005 [68]. A major reason for this drop is believed to be the default configuration of the newer releases of popular operating systems, which refuses $ntpdc$ from unknown IP addresses. Furthermore, various level 1 time sources, such as GPS and radio have now become affordable, creating a possibility of independent and disconnected NTP trees. The only way of finding the nodes in those trees would be to do a brute force check of all the hosts assigned address blocks in the IANA. This, however, borders on rogue Internet behavior and thus was not performed. The other reasons why hosts wouldn't reply are because they are behind a firewall, an NTP version mismatch with the spidering node, some may not have the ntpd daemon running, and other network factors, such as loss of the querying packet.

Table 6.2 shows the distribution of the surveyed nodes according to their stratum in percentages. Comparison to previous recent works is also presented. The comparison does not demonstrate any significant change in the organization of the NTP network. Lower stratum nodes are usually deeper in the local networks hidden behind firewalls and thus more difficult to reach. Stratum 2 and 3 form the majority of surveyed nodes which is logical, since, these strata are the ones exposed to the end users. There seems to be an increase in the number of level 1 nodes, pointing to the easy availability of time sources, such as GPS or Radio.

[1] http://support.ntp.org/bin/view/Servers/StratumOneTimeServers
[2] http://support.ntp.org/bin/view/Servers/StratumTwoTimeServers

Table 6.1: A comparison of the size of various NTP network surveys

Reference	Year	hosts found	replies received
[61]	1989	8455	946 (11.2%)
[35]	1993	15000	7251 (48.3%)
[62]	1995	Unknown	38722 (-)
[63]	1999	647401	175527 (27.1%)
[68]	2005	1290819	147251 (11.4%)
This work	2010	1033295	64682 (6.2%)

Table 6.2: Distribution (in percentages) of the surveyed nodes over stratum

Stratum	1999	2005	This work
1	0.55	0.38	1.46
2	15.29	20.6	27.2
3	48.61	58.98	50.5
4	21.84	17.73	12.44
5	4.06	1.28	2.6
6-15	1.49	1.03	0.8
16	2.77	0	5

The following sections present the results in terms of time offsets and dispersion of the various nodes.

6.3.1 Offset

Clock offset is the estimated amount of time that needs to be adjusted to match a client's clock to its synchronization peer. Thus a smaller offset represents better synchronization. Offset is considered to be one of the most important indicators of the performance of NTP. Figure 6.1 shows the cumulative graph of the absolute offset values recorded. The figure shows that about 90% of the nodes estimate themselves to be within 10ms of their synchronization peers. This result on performance of the NTP network shows that it is possible to assume the high level of synchronization accuracy required for an adapted version Estimated service presented in Chapter 3. In particular, when the user client nodes are synchronized only to their service provider edge node, which is their

physical neighbor. This adaptation still results in a significant improvement in channel utilization. This was presented with an experiment in Section 3.3.7 Offsets are, however, only estimates of the synchronization skew and not strict bounds. The figure also shows that almost 98% of the nodes are within 128ms, after which a very long tail exists. To look at actual theoretical upper bounds across the entire NTP network, we must look at the root dispersion metric presented in the next section.

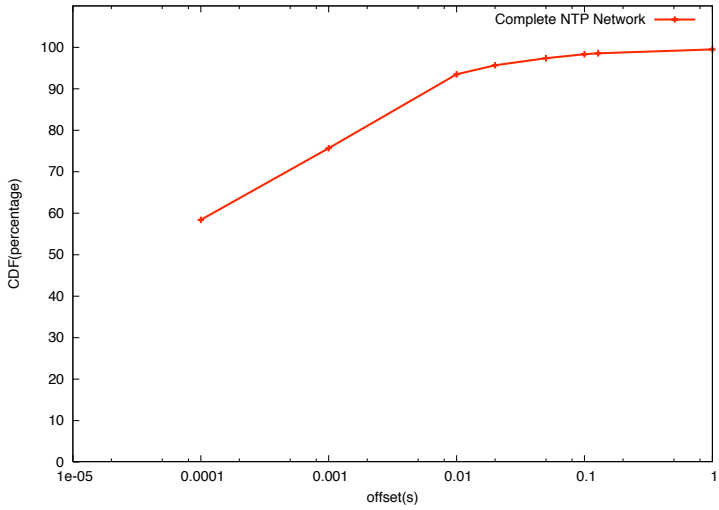


Figure 6.1: Cumulative graph of client to synchronization peer offsets

A comparison with earlier surveys in Table 6.3 demonstrates a slight improvement in the overall mean while the median remains similar. Thus within this 128ms the time differences between nodes are getting smaller, due to advances communication technology.

6.3.2 Dispersion

Dispersion is the maximum possible theoretical bound on time skew at a client. It can be measured with respect to the time at the root reference node or the client's synchro-

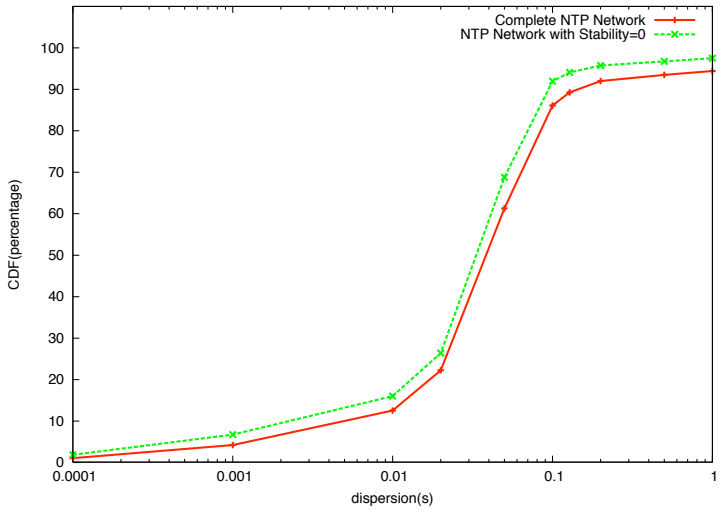


Figure 6.2: Cumulative graph of client to root dispersion

nization peer. Clients compute the root dispersion by summing up the dispersion to their synchronization peers recursively up through to the root node. Two sets of results are presented in this case. The first includes all the nodes in the network and the second with NTP reported *clock stability* of the nodes equal to zero. Clock stability is the exponential average of root mean square (RMS) frequency differences and is measured in parts per million (PPM). Stability value of zero signifies that the frequency variations in the clocks are largely insignificant i.e. the clock is stable. A large value of stability implies a variation in the frequency of the clock. Such variations could occur in nodes that may be old, exposed to environmental elements or just ill maintained. Thus, the results for clocks with stability equal to zero are shown separately from the overall results.

Figure 6.2 shows the cumulative graph of client to root dispersion. Interestingly 90% of the nodes surveyed are bounded within 100ms of each other. This validates our assumption of time synchronization in Chapters 4 and 5. With respect to Chapter 3 this

Table 6.3: Mean and median for offset values for offsets < 128ms

Year	Median (ms)	Mean (ms)
1999	1.8	8.2
2005	0.7	7
This work	0.9	4.8

results does not prove or disprove the assumption of 10-20ms maximum error across the Internet. It is, however, significant that NTP nodes can themselves calculate the maximum possible error in time they may have. If this value is less than the acceptable error they can use the Estserv architecture and, if not, the nodes may dynamically revert back to standard Diffserv. In the future, improvements in technology will reduce errors due to asymmetry in round trip times thereby improving this performance. One such improvement could be classifying NTP packets as control flow in the Internet. This reduces the differences in network traversal times of the NTP query and reply packets. This asymmetry in the to and from times is the main cause for synchronization errors in NTP.

Table 6.4: Mean and medians of dispersion in various studies (without tail)

Year	Median (ms)	Mean (ms)
1999	39	88
2005	40	74
This work	37	44
This work(stability=0)	35	39

Table 6.4 shows a comparison to previous surveys, ignoring the tail component. As in the previous section a significant improvement in the mean can be seen demonstrating the better performance of the current NTP network. Figure 6.2 and Table 6.4 also demonstrate a noticeable better performance for nodes with high clock frequency stability (= 0ppm) when compared to the normal network.

6.4 Summary

The goal of this chapter was to investigate the clock accuracy that can be assumed over a node synchronized using NTP on the Internet. In particular, the question was:

> Question 1.7 *What are the bounds on time difference one can assume between any two nodes in the NTP network?*

To answer this question this chapter presented the results of surveying the existing NTP network using a technique known as spidering. Spidering involves a breadth first search of the entire NTP network starting from a given list of known nodes while inquiring each node about its status. The results presented in this chapter show that close to 90 percent of the surveyed nodes are within 100ms of each other. Thus, conversely one may assume that any two nodes on the Internet, synchronized using NTP, are within 100ms of each other with a probability of about 0.9. This level of accuracy is sufficient for the work presented in other chapters of this thesis: distributed synchronization and the presentation layer mobility mechanism, Chapters 4, 5 respectively.

Estimated service (Chapter 3) however, requires accuracy to within 10 ms to work across the Internet. The results of this chapter show that this currently cannot be assumed on an Internet-wide scale. However, the peer-to-peer offset as presented in this chapter is bounded to within 10 ms for greater than 90 percent of the cases. This implies that, without assuming any improvement in the performance of NTP, in the future Internet all user clients can be synchronized to their neighboring service provider edge routers within 10ms with a high probability. Section 3.3.7 presented an adaptation to the Estimated service architecture where the queuing/scheduling mechanism is only activated on the service provider edge routers. This adaptation to Estimated service is implementable for the current as well as the future Internet. This adaptation yields similar results to normal Estimated service as shown in Section 3.3.7.

In the future, however, spidering may become an inaccurate technique to query the health of the network. Using this technique nothing can be said about the nodes which did not reply or were not queried. With more and more nodes refusing NTP management-query reply to unknown nodes and newer disconnected NTP trees due to the easy availability of level 0 time sources, the majority of the network will not be available for study. Thus, sooner or later an alternative mechanism to study the time dispersion of the nodes in the NTP network must evolve.

CHAPTER 7

Conclusions

In the future services will appear that integrate synchronous communication streams with other applications to form synchronous social experiences. Groups of people in synchronous shared experiences can interact and share services independently of their location and the network they are using. This thesis explored certain technical challenges of synchronous shared experiences. Currently, multimedia-based services are designed for one specific device and network. We envision a future in which friends and family in different domains can converse when watching multimedia content together. The change of paradigm from single end-user consumption to a group shared experience imposes a number of challenges in the way services are designed. This work described a generic architecture and highlighted three particular areas of research for achieving coherence in shared synchronous experiences, namely: quality of service, distributed media synchronization and user mobility.

Unhindered user interaction requires *Quality of Service* guarantees over the communication streams in synchronous shared experiences. However, the future Internet must stay best effort in nature. This implies that guarantees can never be provided. This thesis, thus, looked at the performance of the existing solutions. The current standard for providing Quality of Service is the Differentiated services (Diffserv) network architecture. Diffserv assumes overprovisioning, but in practice an overprovisioned network cannot always be guaranteed. Thus, this work studied the performance of the Diffserv architecture in networks that are not overprovisioned. The thesis recommends an evo-

lutionary extension to the Diffserv architecture, in particular the Expedited forwarding (EF) class. This extension is named Estimated service (Estserv). Results from identical experiments with Diffserv and Estserv demonstrated significantly improved bandwidth utilization in Estserv networks. Furthermore, high bandwidth utilization is not sufficient. Communication streams cannot function below a certain bandwidth limit. Therefore in over-crowded networks, connection admittance needs to be provided to avoid congestion failures. This thesis presented scalable connection admittance mechanism within the Estserv architecture. Our experiments demonstrated that this mechanism only admitted connections in till there was bandwidth available to support them. The connection admittance mechanism was found to be scalable and routers do not need to maintain any per connection memory structures.

Given unhindered communication, media streams need to be rendered in a synchronized manner across users. This thesis adapted the algorithms and architecture for synchronization in distributed games to distributed media playback. A particular case of synchronous shared experiences, Social TV [16], was studied. User tests were performed to identify the tolerance of users to differences in synchronization levels. The user tests concluded that most active users in a social TV setting do not detect differences in play-out under 1s. This value was previously unknown. Validation experiments on the synchronization algorithm found a worst case bound on skew to be $500ms$ between clients based in Amsterdam and Seoul.

Finally this thesis considered user mobility. Coherent movement of user's synchronous shared experiences from one user device or network to another is required in future networks. User mobility is a requirement that is distinct from session mobility, since each synchronous shared experience may contain multiple media and communication sessions. Each session needs to be moved and re-adapted to the new user's context. The previous approaches involved treating a user mobility request as a composite of these session mobility requests. This requires repetitive signaling in the control plane, which is inefficient. This work presented a presentation layer mechanism for achieving user mobility. The mechanism involves saving the state of the presentation in a descriptive language, such as SMIL[1]. This presentation state file can then be transferred using any appropriate means to the new location. The presentation is then re-initiated at the new location and the state reapplied. Re-initiating the presentation automatically ensures the best possible user experience for the new location. The repetitive signaling mechanism required in previous solutions is now avoided, making our approach more efficient and less prone to synchronization errors within the composite streams.

A total of seven research questions were addressed in this work. Section 7.1 restates these questions and summarizes the lessons learnt. The implications of the answers for

[1] http://www.w3.org/AudioVideo

the future of synchronous shared experiences and the Internet as a whole are analyzed. Section 7.2 then discusses some limitations and possible opportunities arising out of this work.

7.1 Summary

Synchronous shared experiences of the future will utilize a congregation of work done in multiple disciplines, including: media codecs, models for 3D rendering, bandwidth optimization, scheduling, synchronization, human computer interaction and causality. The contributions of this thesis have focused on three areas relevant to *coherence* of synchronous shared experiences: quality of service, distributed synchronization and user mobility. The developments in each of these individual areas, amongst others, will determine the architecture of the future Internet. This thesis recommends a new architecture that improves the efficiency of Diffserv-EF class while maintaining its scalability in non-overprovisioned networks. It further recommends a scalable connection admittance mechanism to maintain the usability of the Internet. Chapter 4 demonstrated how distributed synchronization can be designed and implemented in the given network architecture. User studies also demonstrated the tolerance to differences in synchronization level in a shared video watching scenario is about 1 second. Chapter 4 demonstrated how user mobility can be made more efficient for synchronous shared experiences by handling it at the presentation level instead of the session layer (as is currently done). During the course of this thesis, supplementary contributions were made to the areas of bandwidth measurement and ad hoc time synchronization. These are discussed in Appendix A and B. The contributions of this work are summarized in the questions below:

- Question 1.1 *How efficient is the Diffserv network towards communication applications, such as video conferencing, in network that are not overprovisioned?*

  Chapter 3 argued that the current Internet's bandwidth availability acts as a bottleneck in the development of many newer communication streams, such as teleimmersion [69]. Furthermore, the assumption of overprovisioned networks in practice cannot always be ensured, especially across cross domain links and in the face of newer bandwidth-demanding communication applications. Thus for practical purposes it is important to study the working of Diffserv in non-overprovisioned links. Experiments with Diffserv routers in Chapter 3 show that the current design for Diffserv-EF channel is inefficient in terms of *channel efficiency* under non-overprovisioned conditions. This naturally leads to question 1.2 in the search for more efficient architectures.

- Question 1.2 *How can Diffserv be extended to ensure efficient behavior of the future network in non-overprovisioned links while maintaining scalability within the Diffserv architecture?*

  This extension in the Diffserv-EF architecture improves channel utilization efficiency in Diffserv-EF in non-overprovisioned links, while maintaining its performance and scalability otherwise. Chapter 3 presented Estimated Service architecture as an deadline based scheduling extension to Differentiated service architecture. The Estserv architecture involves the insertion of a scheduling component at each intermediate router. This component schedules the incoming packets in order of their increasing "per hop deadlines". The results in Chapter 3 showed that estimated service improves end-to-end bandwidth utilization efficiency in non-overprovisioned links. In overprovisioned links, no queuing takes place, thereby naturally resolving Estserv to the normal Diffserv mechanism. The results also demonstrated that the scalability of Estserv was comparable to that of Diffserv. Experiments with DCCP also demonstrated and increased efficiency in face of congestion control mechanisms. Communication streams like video conferencing are, however, *in-elastic* applications, i.e. there is a minimum bandwidth under which these applications cannot function. This lead to Question 1.3 to a newer congestion control mechanism that would work with this architecture.

- Question1.3 *Is it possible to provide connection admittance control in the future architecture, while still maintaining scalability within the Diffsev architecture?*

  Experiments with Diffserv-EF demonstrated a fair distribution of available bandwidth to all incoming connections, even if this bandwidth was below the minimum usability requirement of the respective multimedia applications. Instead, existing multimedia connections must be prioritized over newer incoming connections. This can be achieved via connection admittance. Chapter 3 presented an RSVP-like mechanism to achieve connection admittance. However, unlike RSVP the intermediate nodes did not need to know anything about the newer incoming or existing connections, making it more scalable. The idea presented in Chapter 3 simply scheduled the connection request packet at the end of all other existing data packets. The connection packets where identified by a deadline value of -1 in their IP headers. This automatically gives existing connections priority over newer incoming ones without requiring the routers to know any connection specifics. The results in Chapter 3 demonstrated the effective working of the connection admittance mechanism. Given a smooth communication stream the users now need to feel that they are interacting with each other about

the same media streams, implying the requirement of a distributed synchronization mechanism on the media stream. This lead to a question about the levels of user tolerance to the skew in synchronized media play out, Question 1.4.

- Question 1.4 *What levels of distributed play out synchronization does a distributed media synchronization system need to achieve?*

  To evaluate the user tolerance to differences in synchronization play out levels of the media stream, given fluent communication streams, first requires the implementation of an infrastructure to synchronize the respective streams, Question 1.5. Given this implementation, Chapter 4 presented the results of user studies with 36 participants alternatively using voice chat and text chat as the communication stream. The experiments found that for a certain type of genre - in this case a quiz show - the synchronization level at which a significant number of users start to notice the skew is around a second. This value replaces the previous rule of thumb of 150ms that comes from tele-communication research [46]. This result has important implications for the design and development of social video watching applications. While currently developers are struggling to meet the 150ms rule of thumb for distributed media synchronization, this work shows that they need to now look at the 1s mark. This makes it possible to implement synchronization algorithms over networks with relatively large end-to-end signaling jitter and across less powerful clients such as mobile phones, such as the one presented as a answer to Question 1.5.

- Question 1.5 *Can event synchronization as used in gaming be extended to synchronize user actions and to achieve distributed media play out synchronization in future shared experiences?*

  Chapter 1 identified that the first synchronous shared experiences across the Internet were distributed games. Synchronization in distributed games is called event synchronization, with existing techniques such as the local lag with time warp and the bucket synchronization mechanism [59, 7]. Chapter 4 re-utilized the local lag mechanism to synchronize user actions, such as “pause”, “play” or “jump to scene” across all the shared experience participants. Furthermore, by classifying play-out position update packets as periodic pseudo user events, it became possible to achieve distributed media play out synchronization using the same algorithms. Chapter 4 adapted this mechanism to previous research on signaling architectures done in media synchronization. In particular, the chapter recommends the use of distributed control [43] for cross domain synchronous shared experiences. Validation results from the implementation of this system

showed an estimated worst case bound on skew level of about 500ms, which according to the results of Question 1.4 is acceptable.

- Question 1.6 *Can user mobility be more efficient than being considered as a collection of individual session mobility request?*

  Synchronous shared experience sessions, such as iSession [34], are composed of multiple individual media sessions. Research in mobility has so far focused on moving one media session from a device to another. Solutions for such composite presentations [14] involve complex mechanism of preserving associations between sessions, and then reissuing all the individual session mobility requests based on such associations. Chapter 5 argued that this is inherently inefficient, and suffers a number of drawback in terms of synchronization. As a solution Chapter 5 recommends a presentation layer mobility mechanism, where the entire shared experience session is saved in a descriptive session language, such as SMIL, and transferred via various available network mechanisms to the target device. In addition, various efficient modifications can be performed to the presentation to better match the target device. The target device re-negotiates the whole presentation and thus it is automatically adapted to the new environment of the user, including new service and content providers. The work done in Chapter 5 demonstrated the feasibility of this approach.

- Question 1.7 *How accurate are the clocks synchronized using NTP?*

  Throughout this work an assumption of time synchronization has been made. While Chapter 3 assumes up to tens of milliseconds of synchronization across the network, it also provides an alternative mechanism if this cannot be assumed. Chapter 4 and Chapter 5 also inherently assume time synchronization to hundreds of milliseconds. Are these assumptions justified? To answer this question we conducted a survey to update the results presented in [68] to find the level of synch spread in the NTP network. The survey showed that any two nodes on the Internet, synchronized using NTP, are within 100ms of each other with a probability of about 0.9, thereby justifying the assumptions made in Chapters 4 and 5. Our work also demonstrated that the approach used in the this and previous works, [61], [63], and [68], could soon become infeasible due to increasingly closed networks and more disconnected NTP trees.

As an answer to the main question this work argues that the design of the existing Internet has to evolve further to facilitate coherence in synchronous shared experiences. More efficient QoS techniques, such as Estimated service that do not violate

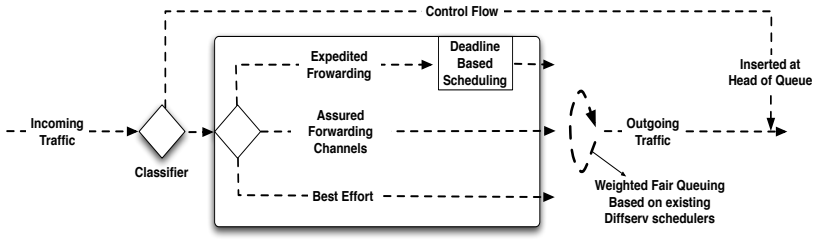


Figure 7.1: Router traffic prioritization in Estserv

the best-effort principles should be evaluated further for deployability on the existing Internet. Every new design for the Internet must be evaluated in terms of the ease of management, both, administratively and economically. Time synchronization should be inherent to the design of the future Internet to facilitate synchronous user mobility and distributed media synchronization mechanisms. Further, as a direct result of this work, control flow should be prioritized and over-provisioned whereas data, including real-time data, must have appropriate best-effort congestion avoidance mechanisms on the edge routers in place. This difference in approach is because control flow occupies a relatively small part of the bandwidth, unlike real-time data in future shared experiences.

7.2 Future Work

The thesis presented solutions to certain technical issues towards the realization of coherence in newer synchronous shared experiences over the Internet. As always, these solutions, in turn, raise further questions with regards to various issues. Some of the possible directions in further extending and refining this work are presented in the following sections.

The next section presents the design of the future Internet. Section 7.2.2, 7.2.3, 7.2.4 discuss opportunities for future directions in QoS, distributed media synchronization and user mobility. Section 7.3 presents brief concluding remarks.

7.2.1 The Future Internet

As a consequence of this work an evolutionary design for the future network can be developed based on the Diffserv architecture. Figure 7.1 shows this prioritization. The

control channel has absolute priority and is always overprovisioned. The reasons for this are that control flow occupies little bandwidth and is important to the working of the synchronous shared experiences and other applications over the Internet. Each of the other channels EF, AF and Best effort are allocated a reserved bandwidth based on the requirements in the SLAs in each of the respective domains. In the edge routers in each domain, the low delay classes EF and AF-low delay must use the Estimated service architecture to improve efficiency in case of congestion. This requires them to be time synchronized and thus connected edge routers must use agreed upon global time sync source. This is even more so valid for access networks and their connecting edge routers. Synchronization related messages (including position update messages, user mobility control requests and time synchronization exchanges) should be assigned control packet priority to achieve higher accuracy.

7.2.2 QoS

This thesis presents work that is the first step in conceptualizing Estimated service architectures. A number of future opportunities of research appear from the work done here. Using the Estimated Service architecture has one additional advantage that can be utilized in the future. Packet drop can now occur at routers selectively and yet without compromising scalability. This can be utilized to eliminate bandwidth adaptation processes in which the client must reduce their transmission. Instead, client may simple mark lower priority packets with a different deadline, thereby creating a difference in how a congested router would treat those packets as compared to packets with normal deadlines. For example, a typical mechanism for reducing video bandwidth usage is dropping B/P frames over I frames at the client. In the new mechanism B/P frames can be assigned a shorter deadline by the client than I frame packets when congestion is detected. This should result in B/P frame packets being dropped before I frame packets. The advantage of doing so is a further increase in channel utilization since the client no longer estimates when to drop the packets. Instead, the routers are the ones that actually drop them. Frames which the client would choose not be sent out earlier, are sent out with shorter deadlines. In case the congestion situation alleviates these frames might make their deadlines, thereby increasing channel utilization.

7.2.3 Media Synchronization

Future synchronous shared experiences will inevitably involve user interactions with virtual objects. In such cases, not only is the synchronization important, causality must also be addressed. Distributed real-time interactions with virtual objects might require stricter bounds on causal correctness than distributed gaming does. Further mecha-

nisms, such as time warp to correct inconsistent states may not always be possible. Further research is required to study these specific cases.

Further user testing needs to be performed to understand the requirements of causality and distributed synchronization. These include parameters for user tolerance study in Social TV that were not covered by this thesis, such as the influence of content on synchronization levels.

7.2.4 Mobility

This thesis illustrated how structured media documents that possess state can be used to make user mobility more efficient. While structured multimedia documents, such as SMIL can enlist media sources including RTP sources, further standardization effort is required to distinguish the behavior and handling of communication streams from media streams, both of which might be RTP sources. An obvious distinction is that communication streams do not incorporate state, as they are real-time. Media playback on the other hand can be paused and at a later time replayed

7.3 Final Remarks

The work done in this thesis yielded the building blocks for creating shared experiences of the future. In particular it distinguished the requirements of the two composing parts of synchronous shared experiences: communication streams and media streams. It looked at providing efficient bandwidth utilization for communication streams, distributed synchronization for media streams and overall mobility of the synchronous shared experience. This thesis experimented with certain basic forms of synchronous shared experiences, such as social TV. More work is required to create more complex and compelling shared experiences. For example, synchronous shared experiences, which incorporate 3D views of the user in virtual words where virtual objects may be constructed together. A number of existing[2] and upcoming projects address certain aspects of such experiences.

[2] www.ta2.eu

APPENDIX A

Measuring bandwidth usage with EUREKA

This Appendix includes the publication [94] re-formatted to this thesis. The research leading to these results has received funding (in part) from the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement n 216647. The publication has the following ISBN number 978-1-4244-7493-6/10/$26.00

APPENDIX B

Cross Domain Time Synchronization

This Appendix includes the publication [95] re-formatted to this thesis. The publication has the following ISBN number: 0-7695-3084-2/07 $25.00© 2007 IEEE.

References

[1] M. Allman, V. Paxson, W. Stevens, et al. Rfc 2581: Tcp congestion control. 1999.

[2] W. Almesberger. Linux network traffic control - implementation overview, 1999.

[3] G. J. Armitage. Revisiting ip qos: why do we care, what have we learned? acm sigcomm 2003 ripqos workshop report. *SIGCOMM Comput. Commun. Rev.*, 33:81–88, October 2003.

[4] J. Bailenson, K. Patel, A. Nielsen, R. Bajscy, S. H. Jung, and G. Kurillo. The effect of interactivity on learning physical actions in virtual reality. *Media Psychology*, 2008.

[5] A. Banerjea, D. Ferrari, B. Mah, M. Moran, D. Verma, and H. Zhang. The Tenet real-time protocol suite: Design, implementation, and experiences. *Networking, IEEE/ACM Transactions on*, 4(1):1–10, 1996.

[6] J. Beerands and F. de Caluwe. The influence of video quality on perceived audio quality and vice versa. *Journal of the Audio Engineering Society*, 47(5):355–362, 1999.

[7] F. Boronat, J. Lloret, and M. Garca. Multimedia group and inter-stream synchronization techniques: A comparative study. *Information Systems*, 34(1):108 – 131, 2009.

[8] R. Braden, D. Clark, and S. Shenker. Rfc 1633 -integrated services in the internet architecture: an overview.

[9] L. Burgstahler, K. Dolzer, C. Hauser, J. Jähnert, S. Junghans, C. Macián, and W. Payer. Beyond technology: the missing pieces for qos success. In *RIPQoS '03: Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS*, 2003.

[10] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. In *Proc. of Performance Evaluation*, 1996.

[11] P. Cesar, D. Bulterman, and J. Jansen. Non-intrusive user interfaces for interactive digital television experiences. *Proceedings of EuroITV*, 2007.

[12] P. César, I. Vaishnavi, R. Kernchen, S. Meissner, C. Hesselman, M. Boussard, A. Spedalieri, D. C. A. Bulterman, and B. Gao. Multimedia adaptation in ubiquitous environments: benefits of structured multimedia documents. In *ACM Symposium on Document Engineering*, pages 275–284, 2008.

[13] S. Chandra. Wireless network interface energy conservation for bottlenecked first mile networks. In *Proceedings of the 20th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '10, pages 117–122, New York, NY, USA, 2010. ACM.

[14] M. Chen, C. Peng, and R. Hwang. SSIP: Split a SIP session over multiple devices. *Computer Standards & Interfaces*, 29(5):531–545, 2007.

[15] CISCO. Cisco autoqos white paper. 2004.

[16] T. Coppens, L. Trappeniers, and M. Godon. AmigoTV: towards a social TV experience. In *Proceedings from the Second European Conference on Interactive Television" Enhancing the experience", University of Brighton*, volume 36, 2004.

[17] B. Davie. Deployment experience with differentiated services. In *RIPQoS '03: Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS*, New York, NY, USA, 2003. ACM.

[18] B. Davie, A. Charny, J. Bennett, K. Benson, J. L. Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. Rfc 2597 -assured forwarding phb.

[19] B. Davie, A. Charny, J. Bennett, K. Benson, J. L. Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. Rfc 3246 -an expedited forwarding phb group.

[20] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12. ACM, 1987.

[21] M. Dick, O. Wellnitz, and L. Wolf. Analysis of factors affecting players' performance and perception in multiplayer games. In *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–7. ACM, 2005.

[22] S. F. E. Kohler, M Handley. Designing dccp: Congestion control without reliability. *ACM Sigcomm*, 2006.

[23] J. Eidson, M. Fischer, and J. White. IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In *34 th Annual Precise Time and Time Interval (PTTI) Meeting*, pages 243–254, 2002.

[24] C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.*, 21(3):270–313, 2003.

[25] T. ETSI. 102 823 v1. 1.1 (2005). *Digital Video Broadcasting (DVB); Specification for the carriage of synchronized auxiliary data in DVB transport streams*.

[26] D. Ferrari. A new admission control method for real-time communication in an Internetwork. *Advances in Real-Time Systems. Prentice Hall*, 1995.

[27] D. Ferrari and D. Verma. A scheme for real-time channel establishment in wide-area networks. *Selected Areas in Communications, IEEE Journal on*, 8(3):368–379, 1990.

[28] T. Ferrari and P. Chimento. A measurement-based analysis of expedited forwarding PHB mechanisms. In *Quality of Service, 2000. IWQOS. 2000 Eighth International Workshop on*, pages 127–137. IEEE, 2002.

[29] S. Floyd. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking (TON)*, 1(4):397–413, 1993.

[30] K.-T. Fung, Y.-L. Chan, and W.-C. Siu. Low-complexity and high-quality frame-skipping transcoder for continuous presence multipoint video conferencing. *Multimedia, IEEE Transactions on*, 6(1):31 – 46, 2004.

[31] D. Geerts and D. de Grooff. Supporting the social uses of television: sociability heuristics for social TV. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 595–604. ACM, 2009.

[32] D. Geerts, I. Vaishnavi, R. Merkuria, P. Cesar, and O. van Deventer. Are we in sync? synchronization requirements for watching online video together. *Proceedings of ACM Conference on Human Factors in Computing Systems, CHI*, 2010.

[33] A. Ghaffar Pour Rahbar and O. Yang. Lgrr: A new packet scheduling algorithm for differentiated services packet-switched networks. *Computer Communications*, 32(2):357–367, 2009.

[34] D. Goergen, J. Zoric, J. O'Connell, O. Friedrich, and B. Zachey. A session model for cross-domain interactive multi-user iptv. In *Proceedings of the 7th IEEE conference on Consumer communications and networking conference*, CCNC'10, pages 306–311, Piscataway, NJ, USA, 2010. IEEE Press.

[35] J. Guyton and M. Schwartz. Experiences with a survey tool for discovering Network Time Protocol servers, 1994.

[36] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Rfc 3260 -an architecture for differentiated services.

[37] I. Hwang, B. Hwang, and C. Ding. Adaptive weighted fair queueing with priority (AWFQP) scheduler for DiffServ networks. *Journal of Informatics & Electronics*, 2(2):15–19, 2008.

[38] J. Hwang and C. Lin. Dynamic frame-skipping in video transcoding. In *Multimedia Signal Processing, 1998 IEEE Second Workshop on*, pages 616–621. IEEE, 2002.

[39] Y. Ishibashi and S. Tasaka. A media synchronization mechanism for live media and its measured performance. *IEICE Transactions on Communications*, 81(10):1840–1849, 1998.

[40] Y. Ishibashi and S. Tasaka. A comparative survey of synchronization algorithms for continuous media in network environments. In *Local Computer Networks, 2000. LCN 2000. Proceedings. 25th Annual IEEE Conference on*, pages 337–348. IEEE, 2002.

[41] Y. Ishibashi and S. Tasaka. Causality and media synchronization control for networked multimedia games: centralized versus distributed. In *Proceedings*

of the 2nd workshop on Network and system support for games, pages 42–51. ACM, 2003.

[42] Y. Ishibashi, S. Tasaka, and H. Miyamoto. Joint synchronization between live and stored media in multicast communications. *lcn*, 00:330, 2000.

[43] Y. Ishibashi, S. Tasaka, and Y. Tachibana. Adaptive causality and media synchronization control for networked multimedia applications. In *IEEE International Conference on Communications, 2001. ICC 2001*, volume 3, 2001.

[44] Y. Ishibashi, A. Tsuji, and S. Tasaka. A group synchronization mechanism for stored media in multicast communications. In *Proceedings of the INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, page 692. IEEE Computer Society, 1997.

[45] R. ITU-T and I. Recommend. P. 800. *Methods for subjective Determination of Transmission quality*, 18, 1998.

[46] R. ITU-T and I. Recommend. G. 114. *One-way transmission time*, 2003.

[47] K. Iwanicki, M. van Steen, and S. Voulgaris. Gossip-based clock synchronisation for large decentralized systems. *SelfMan, Springer-Verlag Berlin Heidelberg*, LNCS 3996(pp. 28-42), 2006.

[48] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. *SIGCOMM Comput. Commun. Rev.*, 32(4):295–308, 2002.

[49] J. Jansen and D. C. Bulterman. Enabling adaptive time-based web applications with smil state. In *Proceeding of the eighth ACM symposium on Document engineering*, DocEng '08, pages 18–27, New York, NY, USA, 2008. ACM.

[50] G. Jin, G. Yang, B. Crowley, and D. Agarwal. Network characterization service (ncs). In *Proc. of 10th HPDC*, 2001.

[51] R. Kernchen, S. Meissner, K. Moessner, P. Cesar, I. Vaishnavi, M. Boussard, and C. Hesselman. Intelligent multimedia presentation in ubiquitous multidevice scenarios. *IEEE Multimedia*, 17:52–63, 2010.

[52] F. D. Keukelaere, R. D. Sutter, and R. V. de Walle. Mpeg-21 session mobility on mobile devices. In H. R. Arabnia and R. Joshua, editors, *International Conference on Internet Computing*, pages 287–293. CSREA Press, 2005.

[53] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[54] T. Kovacikova and P. Segec. Ngn standards activities in etsi. 2007.

[55] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the Real-Time Systems Symposium*, pages 166–171, 1989.

[56] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of ACM*, 20(1), 1973.

[57] D. W. Massaro, M. M. Cohen, and P. Smeele. Perception of asynchronous and conflicting visual and auditory speech. *The Journal of the Acoustical Society of America (J. Acoust. Soc. Am.) ISSN 0001-4966*, vol. 100, 1996.

[58] S. Mate, U. Chandra, and I. D. D. Curcio. Movable-multimedia: session mobility in ubiquitous computing ecosystem. In *MUM '06: Proc. of the 5th Conf. on Mobile and ubiquitous multimedia*, page 8, 2006.

[59] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg. Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia*, 6(1):47, 2004.

[60] D. Mills. Network Time Protocol (Version 4) Specification, Implementation and Analysis. *Network*, 2006.

[61] D. L. Mills. On the accuracy and stablility of clocks synchronized by the network time protocol in the internet system. *SIGCOMM Comput. Commun. Rev.*, 20(1):65–75, 1990.

[62] D. L. Mills, A. Thyagarjan, and B. C. Huffman. Internet timekeeping around the globe, 1997.

[63] N. Minar. A Survey of the NTP Network, 1999.

[64] J. M. Moh and B. Wei. Pqwrr scheduling algorithm in supporting of diffserv. *IEEE International Conference on Communications, Helsinki*, 2001.

[65] S. Mondet, W. Cheng, G. Morin, R. Grigoras, F. Boudon, and W. T. Ooi. Streaming of plants in distributed virtual environments. *ACM Multimedia*, 2008.

[66] M. Montpetit, N. Klym, and T. Mirlacher. The future of IPTV: Adding social networking and mobility. In *Telecommunications, 2009. ConTEL 2009. 10th International Conference on*, pages 405–409. IEEE, 2009.

[67] B. Mukherjee and T. Brecht. Time-lined tcp for the tcp-friendly delivery of streaming media. In *ICNP '00: Proceedings of the 2000 International Conference on Network Protocols*, page 165, Washington, DC, USA, 2000. IEEE Computer Society.

[68] C. Murta, P. Torres Jr, and P. Mohapatra. Characterizing quality of time and topology in a time synchronization network. In *49th IEEE Global Telecommunications Conference, IEEE GLOBECOM, San Francisco, CA*. Citeseer, 2006.

[69] K. Nahrstedt, R. Diankov, R. Bajscy, Z. Yang, B. Yu, and W. Wu. A study of collaborative dancing in tele-immersive environments. *IEEE Symposium on Multimedia*, 2006.

[70] Nemertes. The internet singularity delayed; why limits in internet capacity will stifle innovation on the web. *Nemertes Research*, 2007.

[71] J. Ohm. Advances in scalable video coding. *Proceedings of the IEEE*, 93(1):42–56, 2005.

[72] K. Ohta, T. Yoshikawa, T. Nakagawa, Y. Isoda, and S. Kurakake. Adaptive terminal middleware for session mobility. In *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 394, Washington, DC, USA, 2003. IEEE Computer Society.

[73] F. Pereira and T. Ebrahimi. *The MPEG-4 book*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2002.

[74] S. Ramanathan and P. Rangan. Feedback techniques for intra-media continuity and inter-media synchronization in distributed multimedia systems. *The Computer Journal*, 36(1):19, 1993.

[75] S. Ramanathan and P. V. Rangan. Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks. *IEEE/ACM Trans. Netw.*, 1(2):246–260, 1993.

[76] R. Rejaie, M. Handley, and D. Estrin. Quality adaptation for congestion controlled video playback over the Internet. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 189–200. ACM, 1999.

[77] C. Rentel and T. Kunz. Network synchronization in wireless ad hoc networks. *Carteton Univ., Systems and Computer Engineering, Technical Report SCE-04-08*, 2004.

[78] L. G. Roberts. A radical new router. *IEEE Spectrum Magazine*, 2009.

[79] H. Schulzrinne and E. Wedlund. Application-layer mobility using sip. *SIGMOBILE Mob. Comput. Commun. Rev.*, 4(3):47–57, 2000.

[80] F. B. Segui, J. C. G. Cebollada, and J. L. Mauri. Multimedia group synchronization algorithm based on rtp/rtcp. In *ISM '06: Proceedings of the Eighth IEEE International Symposium on Multimedia*, pages 754–757, Washington, DC, USA, 2006. IEEE Computer Society.

[81] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer. Sip session mobility. *Internet Draft*, 2006.

[82] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer. Ubiquitous device personalization and use: The next generation of ip multimedia communications. *ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 3, No. 2, Article 12*, May 2007.

[83] S. Shalunov and B. Teitelbaum. Quality of service and denial of service. In *RIPQoS '03: Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS*, pages 137–140, New York, NY, USA, 2003. ACM.

[84] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The effects on latency on user performance in warcraft iii. In *Proc. of Netgames*, 2003.

[85] J.-P. Sheu, C.-M. Chao, and C.-W. Sun. A clock synchronization algorithm for multi-hop wireless ad hoc networks. *icdcs*, 00:574–581, 2004.

[86] L. Soares, R. Rodrigues, and M. Moreno. Ginga-ncl: The declarative environment of the brazilian digital tv system. *Journal of the Brazilian Computer Society*, 12:37–46, 2007.

[87] H. Song, H. Chu, and S. Kurakake. Browser session preservation and migration. *Poster Session of WWW 2002, Hawai, USA. 7-11. May, 2002. pp. 2.*, 2002.

[88] G. Stattenberger, T. Braun, M. Scheidegger, M. Brunner, and H. St ”uttgen. Performance evaluation of a Linux DiffServ implementation. *Computer Communications*, 25(13):1195–1213, 2002.

[89] H. Stokking, M. van Deventer, O. Niamut, F. Walraven, R. van Brandenburg, and I. Vaishnavi. Rtcp xr block type for inter-destination media synchronization. *Internt Draft, IETF website, www.ietf.org*, 2010.

[90] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Proc. of IMC*, pages 39–44, New York, NY, USA, 2003. ACM.

[91] P. Svoboda, W. Karner, and M. Rupp. Traffic analysis and modeling for world of warcraft. In *Proc. of ICC*, 2007.

[92] B. Swanson and G. Gilder. Estimating the Exaflood. *The Impact of Video and Rich Media on the Internet. A zettabyte by 2015*.

[93] G. Tselentis, A. Galis, A. Gavras, S. Krco, V. Lotz, E. Simperl, B. Stiller, and T. Zahariadis. *Towards the Future Internet - Emerging Trends from European Research*. 2010.

[94] I. Vaishnavi, A. Arefin, D. Bulterman, K. Nahrstedt, and R. Rivas. Eureka: A methodology for measuring bandwidth usage of networked applications. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 1004–1009. IEEE.

[95] I. Vaishnavi, D. Bulterman, P. Cesar, and B. Gao. Media synchronisation in non-monolithic rendering architectures. *Proceedings of IEEE International Symposium on Multimedia*, 2007.

[96] I. Vaishnavi, D. Bulterman, P. Cesar, B. Gao, and J. Jansen. Neighbourcast: A synchronisation algorithm for ad hoc networks. 2007.

[97] V.Jackobson. Congestion avoidance and control. *ACM Sigcomm*, 88.

[98] J. Weisz, S. Kiesler, H. Zhang, Y. Ren, R. Kraut, and J. Konstan. Watching together: integrating text chat with video. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 877–886. ACM, 2007.

[99] J. Wroclawski et al. RFC 2210: The use of RSVP with IETF integrated services, 1997.

[100] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy. ViewCast: view dissemination and management for multi-party 3d tele-immersive environments. In *Proceedings of the 15th international conference on Multimedia*, pages 882–891. ACM, 2007.

[101] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy. Enabling multi-party 3d tele-immersive environments with viewcast. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 6(2):1–30, 2010.

[102] R. Yavatkar and K. Lakshman. Communication support for distributed collaborative applications. *Multimedia Syst.*, 2(2):74–88, 1994.

[103] K. N. Z. Yang, B. Yu and R. Bajcsy. A multi-stream adaptation framework for bandwidth management in 3d tele-immersion. *Proc. of NOSSDAV'06, Newport, Rhode Island*, 2006.

[104] Y. Zhang, L. Chen, and G. Chen. Globally synchronized dead-reckoning with local lag for continuous distributed multiplayer games. In *NetGames '06*. ACM, 2006.

[105] D. Zhou and T. Lai. A Compatible and Scalable Clock Synchronization Protocol in IEEE 802.11 Ad Hoc Networks. *Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)-Volume 00*, pages 295–302, 2005.