Ontwerp en evaluatie van content distributie netwerken
voor multimediale streaming diensten

Design and Evaluation of Content Distribution Networks
for Multimedia Streaming Services

Tim Wauters

Promotoren: prof. dr. ir. B. Dhoedt, prof. dr. ir. M. Pickavet
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Elektrotechniek

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. P. Lagasse
Faculteit Ingenieurswetenschappen
Academiejaar 2006 - 2007

ISBN 978-90-8578-130-1
NUR 986
Wettelijk depot: D/2007/10.500/4

Universiteit Gent
Faculteit Ingenieurswetenschappen
Vakgroep Informatietechnologie

Promotoren: Prof. Dr. Ir. Bart Dhoedt
Prof. Dr. Ir. Mario Pickavet

Universiteit Gent
Faculteit Ingenieurswetenschappen

Vakgroep Informatietechnologie
Gaston Crommenlaan 8, bus 201
B-9050 Gent, België

Tel: +32 9 331 49 00
Fax: +32 9 331 48 99
Web: http://www.intec.ugent.be

Proefschrift tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen:
Elektrotechniek
Academiejaar 2006-2007

Voorwoord

Met het schrijven van deze thesis zet ik een punt achter 5 jaar projectwerk en doctoraatsonderzoek. Een standaard doctoraat, met doctoraatsopleiding en dito beurs, is het niet geworden. Je kan het eerder omschrijven als een bundeling van de onderzoeksresultaten binnen projecten als CoDiNet, Vlaanderen Interactief, Muse, CIcK en FIPA. Niet altijd even eenvoudig om de rode draad niet te verliezen, maar interessant was het zeker, door de grote verscheidenheid aan onderwerpen en de samenwerking met verschillende industriële partners.

Zonder de inzet en steun van vele mensen, zowel binnen als buiten de IBCN groep, was dit boek echter nooit geschreven geweest. Vooreerst wil ik professor Paul Lagasse bedanken voor het voorzien van de nodige faciliteiten binnen een van de meest toonaangevende vakgroepen van de Gentse Universiteit. Verder ben ik dank verschuldigd aan professor Piet Demeester, hoofd van onze snelgroeiende IBCN groep, voor het vertrouwen en de vrijheid bij het verwezenlijken van dit doctoraat. Meer in het bijzonder wil ik mijn promotoren professor Bart Dhoedt en professor Mario Pickavet, evenals professor Filip De Turck en doctor Didier Colle, bedanken voor hun advies bij de verschillende projecten en de geanimeerde discussies rond mijn doctoraatsonderzoek. Ook vele (ex-)collega's hebben rechtstreeks bijgedragen aan dit werk. Bedankt Jan Coppens, Peter Backx, Thijs Lambrecht, Koert Vlaeminck, Wim Van de Meerssche, Bruno Volckaert, Kristof Lamont en Jeffrey De Bruyne voor de leuke samenwerking rond publicaties en projecten. Een welgemeende dank u is ook op zijn plaats voor de technische ondersteuning van Brecht Vermeulen, Bert De Vuyst, Pascal Vandeputte en Wouter Adem. Ook de vele mensen van het administratieve team bij Intec wil ik hierbij niet vergeten: Martine Buysse, Ilse Van Royen, Davinia Stevens, Marleen Van Duyse, Karien Hemelsoen, Ilse Meersman en Bernadette Becue.

Om van de stress van het doctoraatswerk af te raken kon ik gelukkig ook rekenen op een grote groep vrijwilligers binnen de IBCN groep. Een pico was nooit veraf bij de overblijvenden uit de kaartersclub vanop Urbis, Filip De Turck, Bart Lannoo, Pieter Thysebaert en Koert Vlaeminck, intussen aangevuld met spelers uit verschillende gouden generaties zoals Filip De Greve (de enige mij bekende

Herbertfan buiten mezelf), Johannes Deleu, Nico Goeminne, Niels Sluijs, Tim Stevens, Bruno Van den Bossche, Frederic Van Quickenborne, Dieter Verslype en Gregory Van Seghbroeck. Op diezelfde momenten werd boven de eer (lees "vlag") van onze bureau 3.13 hoog gehouden tijdens de netwerkgames door Koen Casier, Stijn De Smet, Bart Jooris, Abram Schoutteet, Dimitri Staessens, Peter Vandenberghe en/of Jeroen Vanhaverbeke. Samen met Lien Deboosere, Ruth Van Caenegem en nieuwkomers Peter Dedecker en Olivier Verhooghen zorgden zij voor een aangename werksfeer, verder aangevuld met gelegenheidsgebakjes, verjaardagsdrinks en de intussen beruchte IBCN-weekends en -bbq's.

Voor de laatavondontspanning, om mijn zinnen en menig pintje te verzetten, kon ik steeds terecht bij mijn "Gentse" vrienden, merci Bart, Klaartje, Wim, Eva, Birger, Eveline, Bjorn, Katrien, Koen, Evelien en Kristof. Langs de sportieve kant kon ik mij ook steeds uitleven bij de pingpongers uit Stekene, Gent en vooral Buggenhout. Uiteraard gaat de meeste dank uit naar mijn ouders, broer en zus, voor de steun en de hulp die ik niet enkel de laatste 5 jaar, maar ook al die jaren daarvoor heb mogen ontvangen.

Gent, september 2006

Tim Wauters

Table of Contents

List of Figures

List of Tables

List of Acronyms

10BASE-T	10 Mbps BASEband Twisted pair

A

AAA	Authentication, Authorization and Accounting
ADSL	Asymmetric Digital Subscriber Line
ASIC	Application Specific Integrated Circuits
ATM	Asynchronous Transfer Mode
AVC	Advanced Video Coding
AVI	Audio Video Interleave

B

BAS	Broadband Access Server

C

CapEx	Capital Expenditures
CATV	Cable Television
CDN	Content Distribution Network
CfA	Cache from All sources

CfE	Cache from Elected sources
CfS	Cache from Server only
CGI	Computer-Generated Imagery
CMP	Caching Multicast Protocol
CPN	Customer Premises Network
CPU	Central Processing Unit
CWDM	Coarse Wavelength Division Multiplexing

D

DiffServ	Differentiated Services
DMZ	Demilitarized Zone
DNS	Domain Name System
DOCSIS	Data Over Cable Service Interface Specification
DSL	Digital Subscriber Line
DSLAM	Digital Subscriber Line Access Multiplexer

E

E2E	End-to-End
EDL	Edit Decision List

F

FEC	Forwarding Equivalence Class
FIFO	First In First Out
FSK	Frequency-Shift Keying
FTP	File Transfer Protocol

G

GbE	Gigabit Ethernet

H

HDTV	High-Definition TeleVision
HE	Head End
HFC	Hybrid Fiber Coax
HP	Homes Passed
HTTP	Hypertext Transfer Protocol

I

ICMP	Internet Control Message Protocol
ICP	Internet Cache Protocol
IGAP	Internet Group membership Authentication Protocol
IGMP	Internet Group Management Protocol
ILP	Integer Linear Programming
IntServ	Integrated Services
IP	Internet Protocol
IPTV	Internet Protocol TeleVision
IPX	Internetwork Packet eXchange
ISDN	Integrated Services Digital Network
ISO	International Standards Organization
ISP	Internet Service Provider
iVoD	interactive Video on Demand

L

LAN	Local Area Network
LbL	Link-by-Link
LFU	Least Frequently Used
LRU	Least Recently Used

M

MP3	MPEG-1 Audio Layer 3
MPEG	Moving Picture Experts Group
MPLS	MultiProtocol Label Switching

N

NAT	Network Address Translation
NetBIOS	Network Basic Input/Output System
NP	Non-deterministic Polynomial time
NPU	Network Processing Unit
nVoD	near Video on Demand

O

OpEx	Operational Expenditure
ORION	Overspill Routing In Optical Networks
OSI	Open Systems Interconnect

P

P2P	Peer-to-Peer
PPP	Point to Point Protocol
PSTN	Public Switched Telephone Network
PVR	Personal Video Recorder

Q

QAM	Quadrature Amplitude Modulation
QoE	Quality of Experience
QoS	Quality of Service

R

RF	Radio Frequency
RISC	Reduced Instruction Set Computer
RPA	Replica Placement Algorithm
RTCP	Real-Time Control Protocol
RTP	Real-Time Protocol
RTSP	Real-Time Streaming Protocol

S

SAP	Session Announcement Protocol
SDP	Session Description Protocol
SDTV	Standard-Definition TeleVision
SF	Survival of the Fittest

SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical NETworking
SPX	Sequenced Packet Exchange
SR	Storage Renting
SSL	Secure Sockets Layer
STB	Set-Top Box

T

TCP	Transmission Control Protocol
TLS	Transport Layer Security
tsTV	time-shifted TeleVision
TTL	Time-To-Live

U

UDP	User Datagram Protocol
URL	Uniform Resource Locator

V

VC	Virtual Circuit
VCR	VideoCassette Recorder
VLAN	Virtual Local Area Network
VoD	Video on Demand
VoIP	Voice over Internet Protocol

W

WDM	WaveLength Division Multiplexer
WLA	WaveLength Adapter

List of Symbols

α	storage cost relative to transport cost (C_S / C_T)
β	Zipf parameter
γ	link cost parameter
$\lambda_i(t)$	request rate for object i
τ_i	start time of object i

A

$\boldsymbol{A}$	set of cache nodes
$A_{n,o}$	variable indicating the popularity of object o in node n

B

b_i	bitrate of object i
b_e	binary variable indicating if link e is in use
b_s	binary variable indicating if a server has to be installed at server node s

C

c_e	cost of link e
C_S	cost to store one unit in one node

C_T	cost to transmit one unit over one link
$c_{m,1}$	cost of a wavelength multiplexer
$c_{m,2}$	cost of a Gigabit Ethernet multiplexer
c_s	cost of a server port
c_x	cost of a switch port

D

$\boldsymbol{D}$	set of destination nodes
$\boldsymbol{D}_o$	set of destination nodes for object o

E

$\boldsymbol{E}$	set of links

F

F	total number of objects
f	maximum number of simultaneously failing cache nodes

G

$\boldsymbol{G}$	graph
g_n	number of switch ports used at node n

H

$h_I(t)$	hit rate of cache I

$h_{e,d,o}$	binary variable indicating if edge e is used to transmit object o to destination node d

I

i	index of object
i_1	smallest object index for which only one copy is stored
I_n	set of incoming links of node n
i_N	largest object index for which N copies are stored
i_s	installation cost for server s

K

K	number of TV channels

L

l	total bandwidth capacity
l_e	load on link e

M

m_n	storage capacity of node n
$m_{1,n}$	number of wavelength multiplexers used at node n
$m_{2,n}$	number of Gigabit Ethernet multiplexers used at node n

N

N	number of nodes
n_1	number of wavelengths per fiber
n_2	number of GbE signals per wavelength
n_i	number of servers for object i

O

$\boldsymbol{O}$	set of objects
o	object
O_n	set of outgoing links of node n

P

p_e	delay on link e
p_{max}	maximum delay
p_s	number of ports used at server s

R

R	total number of requests
r_i	number of requests for object i
$r_{n,o}$	number of requests for object o in node n

S

s	total storage capacity

$\boldsymbol{S}$	set of server nodes
s_i	size of object i
s_{max}	maximum number of server ports per server node
$S_{n,o}$	variable indicating the storage cost for object o in node n

T

T	total duration of time interval
T_i	duration of object i
$T_{n,o}$	variable indicating the transport cost for object o in node n

U

u_e	bandwidth capacity of link e

V

$\boldsymbol{V}$	set of nodes
v_{max}	maximum number of video streams per GbE link
v_d	number of streams for destination d
$v_{e,d}$	number of streams on link e for destination d

X

X	cache size per object
$\boldsymbol{X}_{0,n}$	set of non-switch nodes at node n
$\boldsymbol{X}_{1,n}$	set of switch nodes at node n
$x_{n,d,o}$	binary variable indicating if node n is used to store object o

for destination node d

Z

$z_{n,o}$ binary variable indicating if node n is used to store object o

Nederlandse samenvatting

Tot tien jaar geleden werd het Internet voornamelijk gebruikt voor client-server toepassingen zoals e-mail en surfen op het Web. In een client-server model bedient één centrale server meerdere eindgebruikers, waarbij het Internet als communicatiemedium gebruikt wordt. Het Internet biedt echter geen garanties bij het versturen van pakketten, zodat deze verloren kunnen raken of te laat of dubbel toekomen. Bovendien kunnen systemen met één centrale server geen hoge belasting aan, zodat eindgebruikers lange antwoordtijden ondervinden. Waar e-mail en surfen op het Web nog minder tijdskritische diensten zijn, vragen recente multimediale diensten zoals streaming video en audio een hogere dienstkwaliteit (QoS). Deze toepassingen vereisen een lage en nagenoeg constante netwerkvertraging en verbruiken een aanzienlijke hoeveelheid bandbreedte in het netwerk. Architecturen met één centrale server voldoen daarom niet meer aan de hoge eisen van de volgende generatie multimediatoepassingen.

Diensten die meer recent op de markt gebracht werden, maken daarom gebruik van alternatieve netwerkmechanismen. Het aanbieden van QoS via IntServ of DiffServ [1] kan de dienstkwaliteit op bepaalde netwerkconnecties verbeteren. Verder zijn er twee aparte evoluties merkbaar op het gebied van architecturen voor het aanbieden van bestanden: een servergebaseerde en een peer-to-peer (P2P) gebaseerde aanpak. In de servertak vinden we concepten zoals de server farm, waar de belasting aan de serverkant verspreid wordt over meerdere servers. Deze techniek kan echter de QoS problemen in het kernnetwerk niet oplossen, zodat gedistribueerde servers en caches geïntroduceerd worden. Deze netwerkentiteiten bevinden zich dichter bij de eindgebruikers, zodat de netwerkvertraging en de belasting op de centrale server verminderd worden, ten koste van een verhoogde opslagkost. De gebruikte opslagmechanismen zijn echter eerder inefficiënt: in het geval van gedistribueerde servers worden alle originele bestanden simpelweg gekopieerd naar elke server en bij proxy caches worden enkel de objecten bewaard die onderschept worden bij de levering door de server naar de eindgebruikers.

Als gevolg daarvan kreeg het concept van Content Distributie Netwerken (CDN's) recent veel aandacht van de industrie en de onderzoekswereld. In dergelijke netwerken, gelijkaardig aan gedistribueerde server farms, bevinden zich meerdere surrogaatservers aan de rand van het netwerk, waarop meerdere replica's van de beschikbare objecten bewaard worden. Deze surrogaatservers verlagen niet enkel de belasting van de centrale server, maar ook die van het kernnetwerk op een effectieve manier. In tegenstelling tot proxy caches, die enkel aanvragen voor lokaal onderschepte objecten kunnen beantwoorden, kunnen surrogaatservers gebruikers bedienen over het hele netwerk. Bovendien worden de bestanden pro-actief verdeeld over de surrogaatservers, waar proxy caches enkel lokaal onderschepte objecten kunnen opslaan. Eindgebruikers worden dan geherrouteerd naar de meest geschikte surrogaatserver, wat de QoS merkbaar verbetert [3].

De belangrijkste diensten die in dit boek bestudeerd worden zijn video-op-aanvraag, televisie over het Internet (al dan niet met ondersteuning van interactiviteit) en multimediaproductie en -opslag. Deze diensten zijn de Internet equivalenten van traditionele diensten zoals kabeltelevisie, verhuur van videofilms of multimediaproductie op videoband. Door deze diensten in digitale vorm over het Internet aan te bieden, kan een meerwaarde geboden worden zoals interactiviteit bij televisieprogramma’s (bv. pauzeren en terugspoelen) of het verwerken van video aan snelheden die veel hoger liggen dan bij de huidige technologie.

Dit onderzoek focust vooral op het ontwerp van het netwerk en de optimale plaatsing van de bestanden hierop. Afhankelijk van de bestudeerde dienst en de gebruikte netwerktechnologieën en -architectuur, zullen verschillende oplossingen voorgesteld worden. Het kopiëren van alle originele bestanden naar surrogaatservers aan elk toegangspunt aan de rand van het kernnetwerk zal ongetwijfeld resulteren in de beste performantie. De geassocieerde opslag- en installatiekosten zijn echter immens. Daarom is het aangewezen om een optimaal evenwicht te zoeken tussen opslagkosten en transportkosten. Hiervoor worden gecentraliseerde algoritmen voor netwerkontwerp ontworpen, vergeleken met een analytische en/of Integer Linear Programming (ILP) formulering en geëvalueerd op verschillende netwerktopologieën. Deze algoritmen optimaliseren de plaatsing van de surrogaatservers of netwerkcaches en bepalen de nodige capaciteit van de servers en netwerklinks. Eenmaal het netwerk ontworpen is, moeten algoritmen die de replica's plaatsen opgesteld worden en qua performantie vergeleken worden met standaard heuristieken (RPA's) [4]. Deze gedistribueerde algoritmen zorgen ervoor dat de plaatsing van de replica's over de verschillende netwerkelementen op een dynamische manier aangepast wordt aan de heersende staat van het netwerk en aan de variërende

aanvraagpatronen van de eindgebruikers. Serverselectie, herroutering van aanvragen en het verspreiden van de belasting over het hele netwerk komen hierbij ook aan bod.

Hoe populairder een bepaalde dienst is, hoe dichter de opslagfaciliteiten zich bij de eindgebruiker moeten bevinden. De capaciteit van deze elementen moet echter sterk beperkt worden, om de total opslagkosten te beperken. Hierdoor spitsen nieuwe RPA's zich vaak toe op het plaatsing van fragmenten van bestanden, die zowel vast als variabel in de tijd kunnen zijn. Naast opslagfaciliteiten zijn vaak ook rekencentra vereist, bv. om videostromen te verwerken. Grid technologieën [5] worden voorgesteld om de eindgebruikers met deze opslag- en rekenfaciliteiten te verbinden en de verwerkingsopdrachten zo optimaal mogelijk te plannen.

References

[1] S. Giordano, S. Salsano, S. Van den Berghe, D. Giannakopoulos, and G. Ventre, "Advanced qos-provisioning in ip networks: The european premium ip projects", IEEE Communications Magazine, 41(1), January 2003.

[2] D. C. Verma, "Content Distribution Networks: An Engineering Approach", John Wiley & Sons, Inc., New York, 2002.

[3] A. Vakali and G. Pallis, "Content delivery networks: Status and trends", IEEE Internet Computing, 7(6):68–74, November 2003.

[4] J. Kangasharju, J. Roberts, and K. Ross, "Object replication strategies in content distribution networks", In Proceedings of Sixth International Workshop on Web Caching and Content Delivery, June 2001.

[5] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999.

English summary

Only a decade ago, the Internet was primarily used for client-server applications such as e-mail or Web browsing. In a client-server model, a single host serves multiple end users, using the Internet as a communication medium. The Internet however is a best-effort network, often resulting in poor network connectivity due to lost or duplicated packets, or packets arriving too late or out of order. Furthermore, single server systems cannot cope with very high loads so that end users may experience long response times. While e-mail or Web browsing applications are less time-critical, recently emerging multimedia services such as streaming audio and video require a much more stringent service quality. These applications need low delay and jitter and use considerable amounts of bandwidth on the network. Single-server architectures are therefore not sufficient anymore to support these next-generation streaming services.

More recent service deployments therefore make use of alternative network mechanisms. Quality of Service (QoS) provisioning techniques such as IntServ and DiffServ [1] can be introduced to increase the levels of QoS on certain network connections. Furthermore, two distinct evolutions in novel content delivery architectures can be observed: a server based and a peer-to-peer (P2P) based evolution. In the server based branch, at first the concept of server farms was presented to effectively balance the load at the server side. This technique however cannot solve the QoS problems occuring in the backbone network, so that distributed servers and caches were introduced. These network entities are placed closer to the end user, so that the network latency can be reduced and the central server offloaded, at the price of an increased storage cost in the network. The used storage schemes however are rather inefficient: in case of distributed servers, the original content is simply duplicated, and proxy caches merely use a passive pull mechanism: only objects that have locally been intercepted can be stored.

Therefore the concept of Content Distribution Networks (CDNs) [2] has recently been proposed. These networks are basically geographically distributed server farms, balancing the load over the network instead of at the origin server site only. These surrogate servers are located at the edge of the network, efficiently

offloading not only the central server, but also the backbone network. Multiple content replicas are distributed over all surrogate servers through a push strategy. Contrary to proxy caches, surrogate servers can answer requests from end users all over the network, instead of from local users only. Client redirection to the appropriate surrogate server is done according to the service policies, such that the end-to-end service quality can be improved considerably [3].

Primary services studied in this work are Video on Demand (VoD), broadcast or time-shifted television and multimedia content production and storage. These streaming services are the IP based equivalents of traditional services such as cable television, video rental services or tape based multimedia production. By offering these services digitally over the Internet, an added value can be provided such as interactivity (e.g. pause and rewind) for television or faster than real-time video rendering for multimedia production.

The focus in this research is on the network design and content placement for these next-generation streaming services. Depending on the service being considered and the existing network architecture and technologies used, different solutions are presented. Although replicating the content of the origin server entirely to a large set of distributed servers at the edge of the network obviously results in the best performance and lowest network costs, the associated storage costs are immense. Therefore appropriate trade-offs between all existing costs have to be calculated. Different sets of centralized network design algorithms are developed, compared to Integer Linear Programming (ILP) and analytical formulations and evaluated on different network topologies. These algorithms tackle the server placement and network capacity planning problems. Once the dimensioning problem is solved, server or cache selection and replica placement algorithms (RPAs), both centralized and distributed, are presented as well and compared to standard RPAs [4]. These algorithms are required in order to dynamically replicate content to the surrogate servers (in case of CDNs) or cache content locally (in case of proxy caching), based on the current network state and varying content request patterns. Optimal server selection, request routing and load balancing are also taken into account.

The more popular a service is, the closer the storage facilities have to be located to the end users and, as a consequence, the larger the number of these facilities grows. To limit the storage costs however, the server or cache capacity of these network elements has to be much lower than at the origin site. Novel RPAs therefore determine the location of (replicas of) partial content, such as fixed or streaming fragments. Besides storage facilities, services such as multimedia production also need computational resources, e.g. rendering farms. To

interconnect the end users with all network facilities and to schedule the appropriate jobs, Grid technologies [5] are brought into play as well.

References

[1] S. Giordano, S. Salsano, S. Van den Berghe, D. Giannakopoulos, and G. Ventre, "Advanced qos-provisioning in ip networks: The european premium ip projects", IEEE Communications Magazine, 41(1), January 2003.

[2] D. C. Verma, "Content Distribution Networks: An Engineering Approach", John Wiley & Sons, Inc., New York, 2002.

[3] A. Vakali and G. Pallis, "Content delivery networks: Status and trends", IEEE Internet Computing, 7(6):68–74, November 2003.

[4] J. Kangasharju, J. Roberts, and K. Ross, "Object replication strategies in content distribution networks", In Proceedings of Sixth International Workshop on Web Caching and Content Delivery, June 2001.

[5] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999.

1 Introduction

This chapter situates the conducted research in a broader context and summarizes the motivations and major contributions of this work. It further outlines the structure of this dissertation and lists the publications in which this work was published.

1.1 Research context

Since its birth in the late 1960s [1], the Internet has evolved from a small scientific research network to a worldwide system, interconnecting millions of smaller domestic, commercial, academic and governmental networks. Currently, more than one billion people [2] use the Internet for e-mail, World Wide Web browsing, content sharing, online messaging, e-banking, video conferencing, online gaming, telephony and many other multimedia services.

Essentially, the Internet has a packet switched nature, supported by the standardized Internet Protocol (IP) [3]. In an IP network, data is split into multiple fragments and each fragment is sent individually over the network in an IP packet. These packets are examined by routers in the network and forwarded hop-by-hop based on internal routing tables. Every packet is treated equally by each router and no guarantees about the proper transmission of the packets can be provided. Traditional Internet applications like e-mail and Web browsing can

cope reasonably well with this best-effort nature of the Internet. Many emerging multimedia services however use streaming video content and thus have much more stringent quality of service requirements, such as low delay and jitter (variation on delay) and high bandwidth.

In order to facilitate the deployment of such next-generation multimedia services, different strategies to provide sufficient Quality of Service (QoS) have been proposed in the recent past. A straightforward solution is to overdimension the network such that a sufficient amount of resources is available for each service. This approach however is not very cost-efficient and still cannot provide a guaranteed QoS. Additionally, since applications become more and more bandwidth-intensive, the network eventually runs out of provisioned resources after all.

Other QoS provisioning mechanisms such as Integrated Services (IntServ) [4] and Differentiated Services (DiffServ) [5] can also provide certain QoS levels. However, due to IntServ's poor scalability and the need for end-to-end collaboration when crossing multiple DiffServ clouds, these technologies have not been widely adopted in the Internet. Furthermore, they increase the complexity of the network management platform significantly.

Content caching or replication strategies, on the other hand, have been studied and deployed on a large scale. Caching proxies [6] are placed close to the end users to reduce latency and offload the servers. They basically cache content based on local popularity metrics, which performs poorly if the requests for a given object are spread globally among many different caching proxies. Additionally, traditional proxies are ineffective when it comes to delivering streaming media.

Another promising solution is peer-to-peer (P2P) technology. This allows to distribute content between peers, instead of from a server. Despite the legal controversy, P2P mechanisms at the end user level are widespread in file sharing networks. An important goal in peer-to-peer networks is that all clients provide resources, including bandwidth, storage space, and computing power. Thus, as nodes arrive and the demand on the system increases, the total capacity of the system also increases. This is not true for a client-server architecture with a fixed set of servers, in which adding more clients could mean slower data transfer for all users. P2P mechanisms can also be deployed at the network level, so that caches can co-operate intelligently to deliver the content. The distributed nature of P2P networks also increases robustness in case of failures by replicating data over multiple (cache) peers, and, in pure P2P systems, by enabling (cache) peers to find the data without relying on a centralized index server. In the latter case, there is no single point of failure in the system.

A recent development addressing the limitations of traditional solutions to improve network QoS, is the concept of content distribution networks (CDNs) [7, 8]. Similarly to caching, a CDN stores multiple replicas of each content item, hosted at various surrogate servers that are typically located at the edge of the network. This way, the content only has to pass through a few nodes in order to reach the end user, resulting in better end-to-end QoS and network usage. Even though the CDN concept is very similar to proxy caching, some important differences should be noted. Caches are placed between the clients and the origin server and serve intercepted client requests. Surrogate servers in a CDN on the other hand can be placed anywhere in the network and clients can be redirected to any surrogate. Furthermore, caches often use a pull strategy to store replicas, while a CDN pro-actively pushes the replicas to various surrogate sites. Replicating the entire content of the origin server to all surrogate sites certainly results in the best performance. However, due to the associated costs of these replica sites, the capacity of surrogates is typically much lower than the capacity of the origin site. Therefore, a CDN requires a replica placement algorithm (RPA) [9] to decide which content to replicate on which surrogate server. Likewise, content retrieval algorithms are used to direct client requests to an optimal surrogate site.

The optimal choice for caching or replicating technologies ultimately depends on the characteristics of the content. Content with higher popularity is typically placed closer to the end users to avoid high network bandwidth costs. If only part of the content is popular (like the last few minutes of a currently broadcasted TV program), partial content storage might be envisaged. When content popularity is known upfront, push technologies are more beneficial than pull technologies.

1.2 Contributions

In this research, network design for next-generation bandwidth-intensive streaming services is combined with adaptable content placement and retrieval algorithms. Depending on network technologies, service requirements and content characteristics, different solutions for service deployment are proposed. This work is related to the research done by Jan Coppens. His Ph.D. mainly focuses on the design of an open, extensible CDN architecture, based on network monitoring, with intelligent replica placement and retrieval algorithms for core networks. In this work, the focus shifts to access network design and content placement, with specific topology and technology requirements.

The contributions of this research can be summarized as follows, based on the multimedia streaming service studied.

- **Video on Demand**

  The network design and replica placement problem for delivery of VoD content is studied (Figure 1.1). A ring based CDN design is proposed for an Asynchronous Transfer Mode (ATM) over Digital Subscriber Line (DSL) based infrastructure. The presented distributed replica placement algorithms aim at optimizing the trade-off between transport and storage costs, while balancing the network load. A distributed server approach is introduced for an Gigabit Ethernet (GbE) over Wavelength Division Multiplexing (WDM) optical infrastructure with a Hybrid Fiber Coax (HFC) access network.

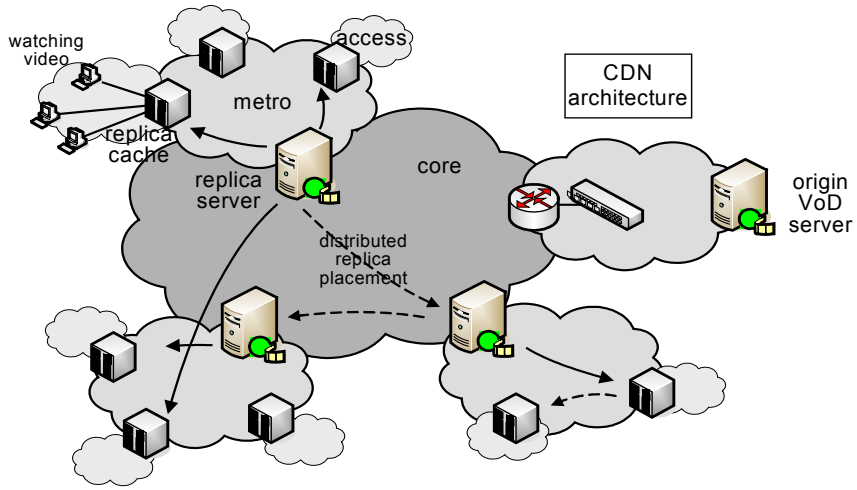


Figure 1.1 Network overview for video on demand

- **Time-shifted TV**

  A hierarchical or co-operative proxy based access network design is presented for a time-shifted TV (tsTV) service (Figure 1.2). The replica placement algorithms have similar metrics as for VoD, but store sliding intervals of streaming content, instead of whole files.

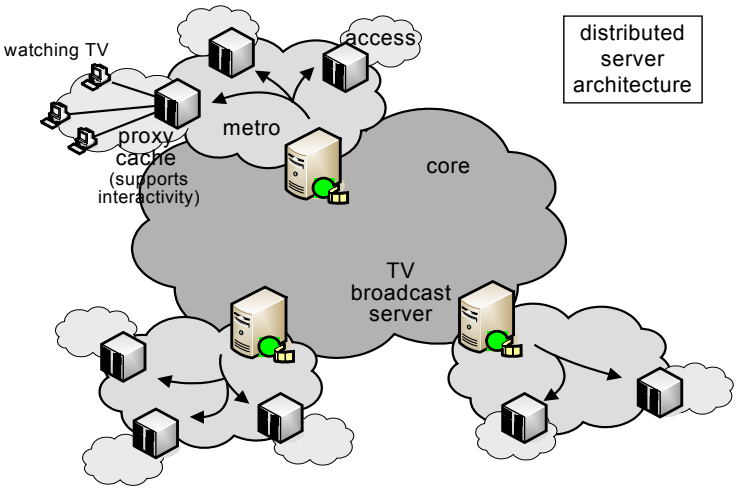


Figure 1.2 Network overview for Internet television

- **Broadcast TV**

  Standard and switched broadcast technologies are brought into play to design the HFC access network for a broadcast IPTV service (Figure 1.2). The installation cost for a given content demand is minimized.

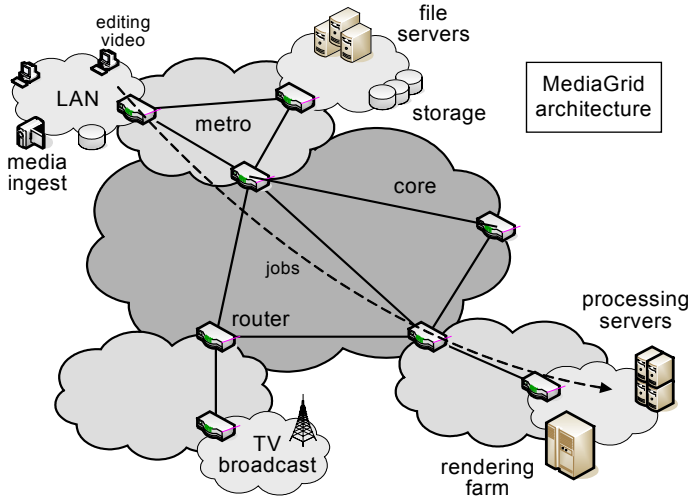


Figure 1.3 Network overview for multimedia production and collaboration

- **Multimedia production and collaboration**

  Algorithms to manage the server bandwidth on a MediaGrid infrastructure are proposed for a multimedia production and collaboration environment (Figure 1.3).

1.3 Organization

This dissertation is structured as follows. Chapter 2 gives an overview of server-based or p2p-based content delivery techniques to improve the network QoS and scalability compared to traditional solutions. The following chapters present service specific solutions for network design and content placement strategies. Chapter 3 describes the network design and replica placement for a Video on Demand (VoD) service deployment on ring based CDNs with p2p cache co-operation. The aim is to reach an optimal trade-off between storage and bandwidth costs, while balancing the network load. In chapter 4 a VoD solution using distributed servers instead of co-operating caches is presented. More detail on the transport costs is introduced, taking the installation costs for network elements using Gigabit Ethernet over WDM technologies into account. Chapter 5 elaborates on time-shifted television, an interactive IPTV service that allows users to watch recently broadcasted programs from the beginning and supports pause, rewind or fast forward commands. Sliding intervals of streaming content are stored on co-operating proxy caches, as determined by distributed replica placement strategies. Another IPTV service is detailed in chapter 6, where standard and switched broadcast technologies are combined for a broadcast TV service. The main goal of the access network design is to minimize the installation cost for a given user demand and available bandwidth spectrum. An introduction on the ongoing work on bandwidth management issues for distributed servers in media production and collaboration networks is studied in Appendix D. Finally, chapter 7 presents some conclusions on this dissertation.

Throughout these different service centric studies, a similar methodology is used. After a formal problem definition and a study of the architectural alternatives, the actual network design problem is tackled. Centralized heuristics are proposed and compared to an Integer Linear Programming (ILP) and/or analytical formulation of the problem. Depending on the service solution, a centralized or distributed approach for the actual content placement is studied and evaluated.

1.4 Publications

The results of this research are published in various scientific papers and presented at a number of international refereed telecommunication-oriented conferences. The following list provides an overview of the publications.

1.4.1 International journal publications

1. **T. Wauters**, D. Colle, E. Van Breusegem, S. Verbrugge, S. De Maesschalck, J. Cheyns, M. Pickavet, P. Demeester, "Virtual topology design issues for variable traffic", published in IEICE Electronics Express, Electronic journal, http://www.elex.ieice.org/, September 25, 2004, Vol. 1, pp. 328-332.

2. **T. Wauters**, D. Colle, M. Pickavet, B. Dhoedt, P. Demeester, "Design of Optical Content Distribution Networks for Video on Demand Services", published in Photonic Network Communications, ISSN 1387-974X, May 2006, Vol. 11, pp. 253-263.

3. T. Lambrecht, B. Duysburgh, **T. Wauters**, F. De Turck, B. Dhoedt, P. Demeester, "Optimizing multimedia transcoding multicast trees", published in Computer Networks, ISSN 1389-1286, January 2006, Vol. 50, pp. 29-45.

4. **T. Wauters**, J. Coppens, F. De Turck, B. Dhoedt, P. Demeester, "Replica Placement in Ring based Content Delivery Networks", accepted for publication in Computer Communications, ISSN 0140-3664, published by Elsevier, online at www.sciencedirect.com.

5. **T. Wauters**, W. Van de Meerssche, P. Backx, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, E. Six, "Proxy Caching Algorithms and Implementation for Time-Shifted TV Services", accepted for publication in European Transactions on Telecommunications (ETT).

6. **T. Wauters,** J. De Bruyne, D. Colle, B. Dhoedt, P. Demeester, L. Martens, K. Haelvoet, "HFC access network design for switched broadcast TV services", submitted to IEEE Transactions on Broadcasting.

7. B. Volckaert, **T. Wauters**, J. Baert, M. De Leenheer, P. Thysebaert, F. De Turck, B. Dhoedt, P. Demeester, "Design of a MediaGrid framework and simulation of workflows for collaborative audiovisual organizations", submitted to Future Generation Computer Systems: The International Journal of Grid Computing: Theory, Methods and Applications.

8. **T. Wauters**, K. Vlaeminck, W. Van de Meerssche, S. Van den Berghe, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, E. Six, "IPTV

deployment: trigger for advanced network services!", accepted for publication in The Journal of the Communications Network.

1.4.2 International conference publications

9. P. Backx, **T. Wauters**, B. Dhoedt, P. Demeester, "A comparison of peer-to-peer architectures", published in Conference proceedings of Eurescom 2002 Powerful Networks for Profitable Services, Heidelberg, Germany, October 21-24, 2002, pp. 215-222.

10. **T. Wauters**, J. Coppens, T. Lambrecht, B. Dhoedt, P. Demeester, "Distributed replica placement algorithms for peer-to-peer content distribution networks", published in Proceedings of Euromicro 2003, the 29th Euromicro Conference "New Waves in System Architecture", Belek-Antalya, Turkey, September 1-6, 2003, pp. 181-188.

11. D. Colle, **T. Wauters**, E. Van Breusegem, S. Verbrugge, S. De Maesschalck, J. Cheyns, M. Pickavet, P. Demeester, "Virtual topology design issues for variable traffic", published in OECC/COIN 2004 9th OptoElectronics and Communications Conference, 3rd International Conference on Optical Internet, Pacifico Yokohama, Kanagawa, Japan, 12-16 July, 2004, pp. 408-409.

12. **T. Wauters**, D. Colle, M. Pickavet, B. Dhoedt, P. Demeester, "Optical network design for video on demand services", published in Proceedings of ONDM05, Conference on Optical Network Design and Modelling, Milan, Italy, 7-9 February 2005, pp. 251-260.

13. **T. Wauters**, J. Coppens, B. Dhoedt, P. Demeester, "Load balancing through efficient distributed content placement", published in Proceedings (on CD-ROM) of the 1st EuroNGI Conference on 2005 Next Generation Internet Networks Traffic Engineering, Rome, Italy, 18-20 April 2005.

14. J. Coppens, **T. Wauters**, F. De Turck, B. Dhoedt, P. Demeester, "Evaluation of a monitoring-based architecture for delivery of high quality multimedia content", published in Proceedings of the 10th IEEE Symposium on Computers and Communications, ISCC 2005, Cartagena, Murcia, Spain, 27-30 June 2005, pp. 611-616.

15. J. Coppens, **T. Wauters**, F. De Turck, B. Dhoedt, P. Demeester, "Evaluation of replica placement an retrieval algorithms in self-organizing CDNs", published in Proceedings of (on CD-ROM) the IFIP/IEEE International Workshop on Self-Managed Systems & Services, SELFMAN 2005 co-located with IM 2005, Nice, France, 19 May 2005.

16. J. Coppens, **T. Wauters**, F. De Turck, B. Dhoedt, P. Demeester, "An architecture for delivery of streaming media content based on network monitoring", published in Proceedings of the 2005 International Conference on Communications in Computing, CIC'05 (part of The 2005 International MultiConference in Computer Sc, Las Vegas, Nevada, USA, June 27-30, 2005, pp. 177-183.

17. J. Coppens, **T. Wauters**, F. De Turck, B. Dhoedt, P. Demeester, "Design and performance of a self-organizing adaptive content distribution network", published in Proceedings (on CD-ROM) of NOMS2006, the 10th IEEE/IFIP Network Operations & Management Symposium, Vancouver, Canada, 3-7 April 2006, pp. 534-545.

18. J. Baert, M. De Leenheer, B. Volckaert, **T. Wauters**, P. Thysebaert, F. De Turck, B. Dhoedt, P. Demeester, "Hybrid optical switching for data-intensive media grid applications", published in Workshop on Design of Next Generation Optical Networks, Ghent, Belgium, 6 February 2006, pp. 9-14.

19. **T. Wauters**, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, E. Six, "Co-operative Proxy Caching Algorithms for Time-Shifted IPTV Services", published in Proceedings of the 32nd EuroMicro Conference 2006, Dubrovnik, Croatia, Augus 29 - September 1, 2006, pp. 379-386.

20. **T. Wauters**, K. Vlaeminck, W. Van de Meerssche, S. Van den Berghe, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, E. Six, "IPTV deployment: trigger for advanced network services!", published in Proceedings of the 45th FITCE Congress 2006, Athens, Greece, August 30 - September 2, 2006, pp. 36-40.

21. **T. Wauters**, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, E. Six, "Management of time-shifted IPTV services through transparent proxy deployment", published in the proceedings of IEEE Globecom 2006 (on CD-ROM), San Francisco, USA, 27/11-01/12, 2006.

1.4.3 National conference publications

22. P. Backx, B. Duysburgh, T. Lambrecht, L. Peters, **T. Wauters**, P. Demeester, B. Dhoedt, "Enhanced Applications through Active Networking", published in 2nd FTW PHD Symposium, Interactive poster session, paper nr. 99 (proceedings available on CD-Rom), Gent, Belgium, December, 2001.

23. **T. Wauters**, P. Backx, J. Coppens, B. Dhoedt, P. Demeester, "P2P architectures for content delivery", published in 3rd FTW PHD Symposium, Interactive poster session, paper nr. 22 (proceedings available on CD-Rom), Gent, Belgium, December, 2002.

24. **T. Wauters**, J. Coppens, B. Dhoedt, P. Demeester, "High quality multimedia delivery on CDNs", published in 5th FTW PHD Symposium, Interactive poster session, paper nr. 010 (proceedings available on CD-Rom), Gent, Belgium, December, 2004.

1.5 Patents

The research on time-shifted television (Chapter 5) resulted in a European patent application, for which the outcome is still pending.

1. E. Six, T. Van Caenegem, W. Van de Meerssche, F. De Turck, **T. Wauters**, B. Dhoedt, "Access/edge node supporting multiple video streaming services using a single request protocol", patent application number 05292236.6, filed on 24/10/2005.

References

[1] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Robberts and S. Wolff, "The past and future history of the Internet", Communications of the ACM, 40(2), February 1997.

[2] Internet World Stats, Usage and population statistics, http://www.internetworldstats.com.

[3] J. Postel, "Internet protocol", IETF RFC 791, September 1981.

[4] J. Wroclawski, "The use of rsvp with ietf integrated services", IETF RFC 2210, September 1997.

[5] S. Blake, D. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service", IETF RFC 2475, December 1998.

[6] A. Vakali and G. Pallis, "Content delivery networks: Status and trends", IEEE Internet Computing, 7(6):68–74, November 2003.

[7] D. C. Verma, "Content Distribution Networks: An Engineering Approach", John Wiley & Sons, Inc., New York, 2002.

[8] M. Day, B. Cain, G. Tomlinson, and P. Rzewski, "A model for content internetworking (cdi)", IETF RFC 3466, February 2003.

[9] J. Kangasharju, J. Roberts, and K. Ross, "Object replication strategies in content distribution networks", In Proceedings of Sixth International Workshop on Web Caching and Content Delivery, June 2001.

2 Overview on content distribution networks[1]

2.1 Introduction

Only a decade ago, the main goal of the Internet was to provide network connectivity for exchanging e-mail and browsing static web pages. Since then an enormous growth in Internet applications has occurred. Delivery of rich multimedia content through file sharing applications, video conferencing or online gaming has become possible due to the ever-increasing backbone capacity, the rising number of broadband access lines and a steady evolution in network technologies. Where at first only client-server models were present, different content delivery architectures have now been studied and deployed.

This chapter gives a brief overview of commonly used content delivery techniques and architectural evolutions. Section 2.2 describes the characteristics of streaming media and commonly used protocols. In Section 2.3 the classical

[1] In literature, the term content distribution networks is used as a general description for such networks, as well as for the specific technology described in Section 2.3.2.d. In the latter case it is abbreviated to "CDN" in this book.

client-server model using the Internet Protocol (IP) stack is described. Afterwards, the progress in network architectures is detailed in two distinct branches: a server based and a peer-to-peer based evolution. The first branch introduces distributed servers and caches in the network (section 2.3.2), while a second branch focuses on P2P techniques (section 2.3.3). Section 2.4 presents commonly used network support mechanisms, such as multicasting, broadcasting and traffic engineering. The underlying access network technologies, such as DSL and HFC, are explained in Section 2.5. An overview of the service specific solutions presented further on in this book is given in Section 2.6.

2.2 Streaming media characteristics and protocols

In the past years, multimedia content had to be downloaded entirely at the clients computer before it could be viewed. Streaming media, which can be viewed while it is being delivered, become more and more wide-spread in today's Internet. The popularity of multimedia websites, online video streaming of live events and listening to music or radio increases continuously. The most important characteristics of streaming media are its high bandwidth requirements and its sensitivity to packet loss and jitter.

Despite the growing market penetration of broadband Internet connections such as cable and DSL, delivery of video streams in uncompressed form is impossible. Bit rates of raw video can be larger than 1 Gbps. Therefore compression techniques are needed to reduce bandwidth consumption. Commonly used video codecs such as H.263 (compressed bitrates from 28.8kbps up to 768kbps), MPEG-1 (400kbps up to 1.5Mbps), MPEG-2 (1.5Mbps up to 15Mbps), MPEG-4 (28.8kbps up to 500kbps), DivX, WMV and RealVideo provide different mechanisms to reduce the size of the video content, at the price of quality degradation [3]. One of the most promising new codecs is H.264 (or MPEG-4 Part 10 or AVC, Advanced Video Coding). Table 2.1 shows some example H.264 data rates for typical use scenarios.

Trade-offs between different aspects such as video quality, codec complexity, (de-)compression processing power, robustness and end-to-end delay have to be balanced. Compressed content is then assembled in media containers such as MOV (for the QuickTime player), MP4 (MPEG-4), RM (RealMedia) and AVI (Microsoft Windows). A container is a file format containing various types of synchronized data, such as audio, video, subtitles and meta data, compressed by means of standard codecs.

Use scenario	Resolution, frame rate	Example data rates
Mobile	176x144, 10-15 fps	50-60 Kbps
Internet (standard def.)	640x480, 24 fps	1-2 Mbps
High def.	1280x720, 24p	5-6 Mbps
Full high def.	1920x1080, 24p	7-8 Mbps

Table 2.1: Data rates of the H.264 video codec

Since media streaming is sensitive to packet loss, packet reordening and jitter, best-effort IP is insufficient for multimedia streaming over the Internet. Therefore a transport layer protocol is required to ensure end-to-end data transfer and integrity across the network. A possible transport layer protocol is TCP, which is connection oriented, reliable and provides flow control. However, the TCP protocol asks for the retransmission of lost packets, which introduces delays during playback of the live stream. A better choice is to use the connectionless user datagram protocol (UDP), which is simple and efficient, but cannot avoid packet loss or reordening. When UDP is encapsulated in the application layer real-time transport protocol (RTP) [12], packets can be put in the right order using the RTP packet sequence numbers and timestamps. The RTP / RTCP / RTSP protocol suite is commonly used for delivering streaming media. The real-time transport control protocol (RTCP) [12] allows for scalable monitoring and QoS control of the data streams, while the real time streaming protocol (RTSP) [13] is an application layer protocol used to control the delivery of the streams themselves by means of VCR-like commands such as play, pause, rewind and fast-forward. Stream Control Transmission Protocol (SCTP) [49] is another transport layer protocol, similar to TCP, as it ensures reliable transport of data with congestion control. The main difference is that TCP transports byte streams, where SCTP transports multiple message streams, where a message is a group of related bytes, such as an image or a video file. Moreover, these messages can be sent in parallel over the same SCTP connection, e.g. several images from one web site.

A client-side solution to reduce delay and jitter is to buffer part of the streamed content. Once enough data is buffered locally, the stream can be started or continued smoothly with only a small delay.

2.3 Architectural evolutions and protocols

This section describes the classical client-server model and details the evolution in server based and peer-to-peer based network architectures.

2.3.1 Traditional client-server model

In a client-server model, the end user has the client software, e.g. a Web browser or e-mail client, and connects to the server running the service, through a network. As shown in Figure 2.1, the content has to go through the client access network, the Internet backbone and the server access network. This model is still used by many Internet applications such as Web browsing, e-mail, news groups and File Transfer Protocol (FTP) access.

In order to establish a connection for the data transfer between the client and the server, a standard communication protocol for the Internet has been defined. The ISO (International Standards Organization) has created the layered OSI (Open Systems Interconnect) model, to describe and define layers in a network operating system [1]. Each layer only uses the functionality of the layer below, and only exports functions to the layer above. The main goal of this protocol stack is to provide interoperability across various vendor platforms.

Table 2.2 lists the different layers of the ISO OSI model and gives some examples of protocols for each layer.

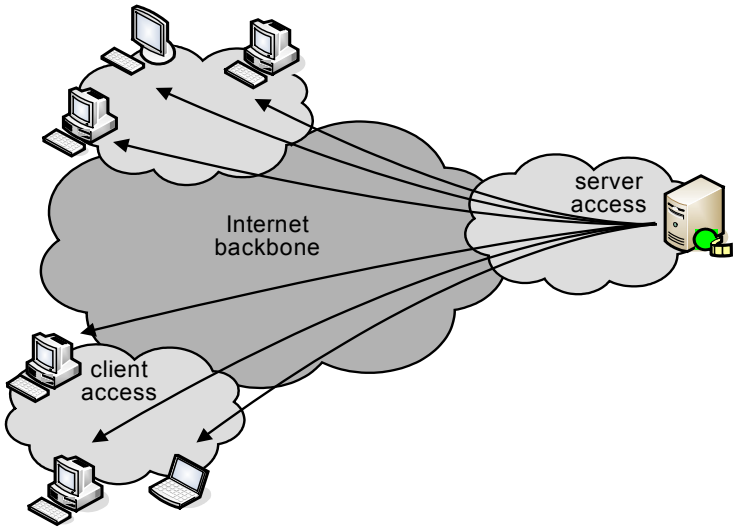


Figure 2.1 Classical client-server model

The application layer contains the applications protocols, such as HyperText Transfer Protocol (HTTP) [2] for Web browsing. The presentation layer performs data transformations such as MPEG [3] compression for video streaming applications. The session layer controls the dialogues between local and remote applications. The transport layer ensures end-to-end data transfers and integrity across the network. Transmission Control Protocol (TCP) [4] and User Datagram Protocol (UDP) [5] are important example protocols in this layer. Routing protocols such as IP [6] are situated in the network layer, responsible for routing packets across the network. The data link layer transfers data units between individual network entities over a transmission circuit, assuring data integrity. Asynchronous Transfer Mode (ATM) [7] and Ethernet [8] are well-known data link layer protocols. The physical layer is responsible for the actual bit stream on the physical medium, defining electrical and procedural formats.

TCP and IP are the two most important protocols in the Internet protocol stack (also called TCP/IP stack), only defining the OSI layers 2, 3, 4 and 7. Networks based on these protocols, such as the Internet, have gained wide social acceptance. The IP protocol is a connection-less network layer protocol used to transfer data across a packet-switched network of IP routers between source and destination hosts.

Layer	Protocol	Examples
7	Application	HTTP, FTP, SMTP, RTP
6	Presentation	MPEG, SSL, TLS
5	Session	NetBIOS, SAP, SDP
4	Transport	TCP, UDP, SPX
3	Network	IP, ICMP, IPX
2	Data link	ATM, Ethernet, Token ring
1	Transmission	10BASE-T, ISDN, SONET, DSL

Table 2.2: OSI model

The data is cut in fragments and sent over the network in packets. Every router in the IP network examines the IP header in each packet and sends each packet to the next hop based on its internal routing table. Since no fixed connection has to be set up between the source and destination host prior to sending the first

packet, the IP protocol is inherently unreliable. Only a best-effort service is provided, so that packets may arrive damaged, out of order or not at all. If reliability is needed, it has to be provided by the upper layer, e.g. by TCP.

Figure 2.2 shows an example of an e-mail application. End users use an e-mail client such as Pine or Eudora, which connects to a local e-mail server through TCP/IP. This server listens on TCP/IP port 25 for incoming Simple Mail Transfer Protocol (SMTP) [9] messages (from e-mail clients wishing to send an e-mail) and on port 110 for incoming Post Office Protocol version 3 (POP3) [10] messages (from e-mail clients wishing to retrieve an e-mail). A more advanced alternative to POP3 is the Internet Message Access Protocol (IMAP) [11]. E-mail messages are forwarded between intermediate e-mail servers using SMTP.

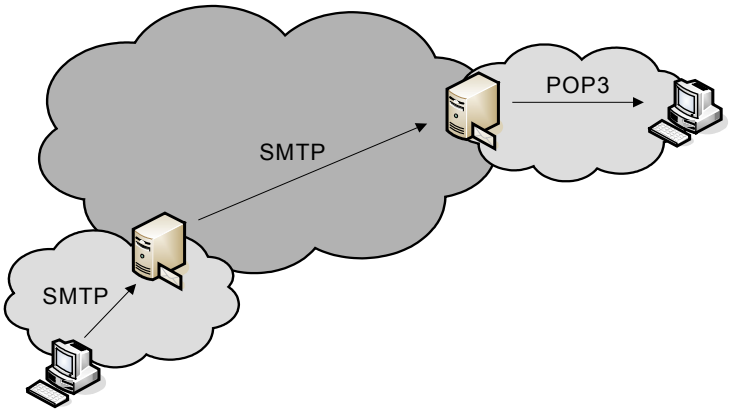


Figure 2.2 E-mail application

2.3.2 Server branch

This section describes traditional solutions to increase the quality of service, based on replication of content. The storage of the multimedia content is not restricted anymore to the central origin server, but relocated or replicated over different servers or caches. Each of the solutions studied in this section tries to reach one or more of the goals listed below:

- Reduction of the central server load
- Reduction of the client-perceived latency
- Reduction of the network bandwidth usage
- Balancing of the network bandwidth usage

2.3.2.a Server farm

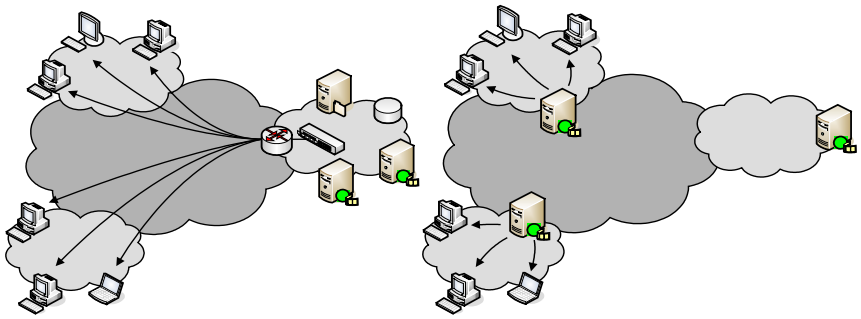
In a server farm (Figure 2.3a), the content is spread and/or replicated over multiple servers, interconnected by a local area network (LAN). Client requests can be directed to one of these servers or the content can be retreived in parallel from multiple servers (e.g. different files on a multimedia website). While server farms do not reduce network latency or bandwidth usage, the service scalability is increased significantly at the server side. Various techniques to direct a specific request to one of the servers are possible.

- A front-end load balancer is often used as an application layer switch to direct the client requests to the appropriate LAN server. Important characteristics for load balancing are the server load and available LAN resources. Client redirection can be accomplished using network address translation (NAT) [26] or shared IP addresses.
- Broadcast and filter techniques can be introduced as well. In that case, an access router sends an incoming client request to all content servers and a collaborative selection protocol ensures that only one server handles the request, e.g. based on the clients IP address.
- Smart directory servers use Domain Name Server (DNS) redirection, with addressing based on the IP address and the name of the server machines.
- Smart selection by the client is possible when a set of addresses of possible servers located in the farm is handed to the clients, who make their own choice for a content server.

Some applications, such as an e-commerce site using shopping histories, require a client to be directed to the same server for every request, so that load balancing techniques on server farm are not applicable as such. These applications balance sessions instead of single requests through session tracking, which can be accomplished using cookies or Uniform Resource Locator (URL) rewriting.

2.3.2.b Replicated servers

A similar concept is to replicate the entire central server to multiple locations in the network (Figure 2.3b). This way content can be located much closer to the end users, e.g. at the edge of the backbone network, so that network latency and bandwidth usage can be reduced effectively. The storage costs however are much higher than in case of a solution using a single server or server farm. Especially in case of unpopular content, the increase in storage costs does not compensate the reduction in transport costs. Therefore the solution using replicated or distributed servers is only beneficial for very popular services or services that have very stringent QoS requirements.



(a) *(b)*

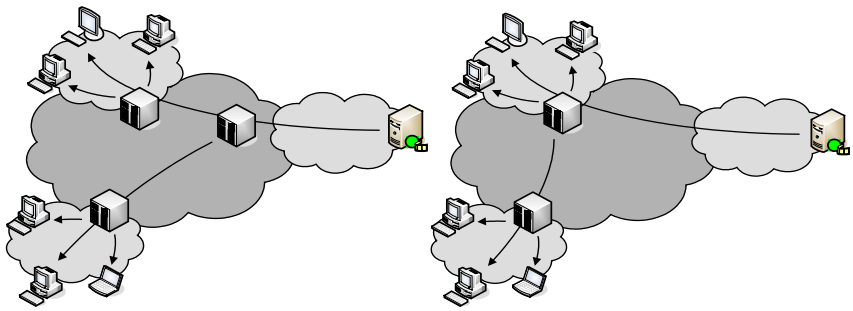
Figure 2.3 Server farm with front-end load balancer (a) vs replicated servers (b)

2.3.2.c Proxy caching

Proxy caches are placed at strategically located positions in the network to cache certain popular content in order to reduce the response times for future requests for the same content [27]. When a client sends a request for a certain file, its local proxy cache checks whether the file is stored locally. If that is the case (a cache hit), the request is served locally, resulting in a very low response time. Otherwise (a cache miss), the request is forwarded to and handled by the central server. The proxy cache then decides, according to a specific caching strategy, whether it is beneficial to store the passing file locally to serve later requests. Commonly used replacement strategies are least recently used (LRU), least frequently used (LFU) and first in first out (FIFO) [28]. Caches can typically only cover a relatively small group of users, e.g. the users of a single Internet Service Provider (ISP), university or company. Cache misses, in case of large service deployments, can result in very high response times. Two complementary mechanisms can be brought into play to further minimize these delays, as well as the bandwidth on the network between the caches and the origin server.

- Hierarchical caching [29] (Figure 2.4a) attempts to solve this problem through a tree-based hierarchical configuration with additional regional and/or national caches. Caches misses are forwarded to the parent caches before they reach the origin server. Lower level caches typically store the (locally) most popular content, while higher level caches serve less popular content. The central server is thereby offloaded of most client requests.
- Co-operative caching (Figure 2.4b) is a technique where neighbouring proxy caches co-operate to serve each others cache misses. Co-operative Web caching was first introduced with the design of the internet cache protocol

(ICP) [30-32]. The ICP supports the discovery and retrieval of documents from neighbouring caches and allows caches to query other caches for content. More recent protocols, such as the Summary Cache, increase the scalability of the caching network.



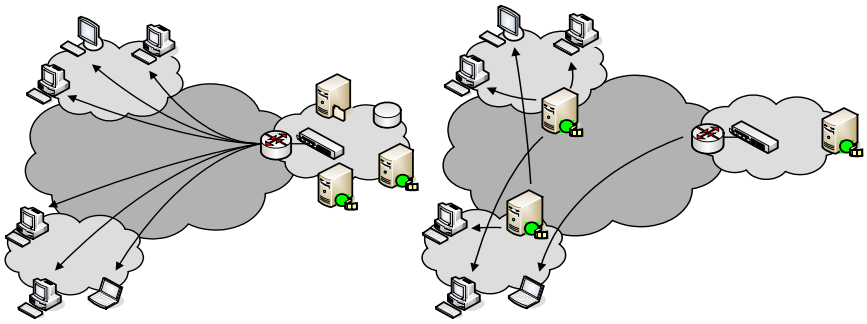
(a) *(b)*

Figure 2.4 Hierarchical (a) versus co-operative (b) proxy caching

As mentioned before, the major advantage of proxy caching is the reduction in client-perceived latency, network bandwidth usage and server load, especially for popular content. It is however difficult to fill the caches efficiently, since content popularity changes regularly, new content might be added frequently and collaboration between caches on different levels is not straightforward. Furthermore, content providers have no influence on what is cached, so that proxies that are not properly updated, might deliver stale data to the end users. Expire header fields or cache control mechanisms to avoid this are present in the HTTP protocol, but have never been widely used. Another important disadvantage is that caching is a passive pull mechanism, so that flash crowds cannot be handled adequately. Since during flash crowds many requests are made for popular but still recent content, such as live streamed events, which has not been stored close to the end users yet, the origin server gets overloaded.

2.3.2.d Content distribution networks

Essentially, a CDN can be described as a distributed server farm, where content servers are located on geographically distributed locations at the edge of the backbone network (Figure 2.5).



(a) *(b)*

Figure 2.5 Server farm with front-end load balancer (a) versus Content Distribution Network (b)

Although similar to the replicated server solution, CDNs do not just duplicate the content on the origin server to all CDN servers, often called surrogate servers. The content is replicated according to a replica placement strategy [33-35], based on popularity and distance metrics, taking bandwidth and storage costs into account. At this point, CDNs use a push strategy to store replicas, where a proxy caching solution would follow a passive pull strategy. Actively pushing content to surrogate servers results in more optimal replica placements and makes CDNs robust against flash crowds (for services where they can be predicted), but obviously requires a more sophisticated management platform. To provide robustness against server or network failures, content is replicated to multiple surrogate servers. Therefore clients have to be redirected to the best of the available surrogate servers, based on criteria such as bandwidth availability, server proximity, server load and/or content availability [36]. This choice depends on the design options of the CDN: minimal client perceived latency, optimal network resource usage, maximal service scalability, etc.. Different redirection schemes for request routing are possible:

- Similar to the front-end load balancer in a server farm, a switch at the origin server site can be used to redirect the client to the appropriate surrogate server. The actual content can be streamed from the surrogate server through the load balancer to the client, or directly from the surrogate server to the client. The latter case is called triangular routing and could be implemented through mobile IP [37].
- Request routing by means of DNS [38,39] routing is shown in Figure 2.6. The client's request sent to the host name of the CDN origin server is intercepted by a local DNS, e.g. from its ISP. This DNS needs to resolve the

host name and queries the CDN's DNS, who responds with the IP address of the appropriate surrogate server. The client then retrieves the requested content directly from the selected replica server. Due to the ubiquity of the DNS system in the Internet [40], this technique is very widespread.

Well-known CDN service providers are Akamai [50], Mirror Image [51] and Edgestream [52]. Akamai is also one of the co-authors of Edge Side Includes (ESI) [53], a simple markup language used to define Web page components for dynamic assembly and delivery of Web applications at the edge of the Internet. ESI provides a mechanism for managing content transparently across application server solutions, content management systems and content delivery networks. As a result, ESI enables companies to develop applications once and choose at deployment time where the application should be assembled – on the content management system, the application server or the content delivery network, thus reducing complexity, development time and deployment costs. A summary of the main tags is given in Table 2.3.

Tag	Purpose
<esi:include>	Include a separately cacheable fragment.
<esi:choose>	Choose among multiple alternatives based on cookie value or user agent (conditional execution).
<esi:try>	Specify alternative processing when a request fails (e.g., the origin server is not accessible).
<esi:vars>	Permit variable substitution (for environment variables).
<esi:remove>	Specify alternative content to be stripped by ESI but displayed by the browser if ESI processing is not done.
<!--esi ... -->	Specify content to be processed by ESI but hidden from the browser.
<esi:inline>	Include a separately cacheable fragment whose body is included in the template.

Table 2.3: ESI tags

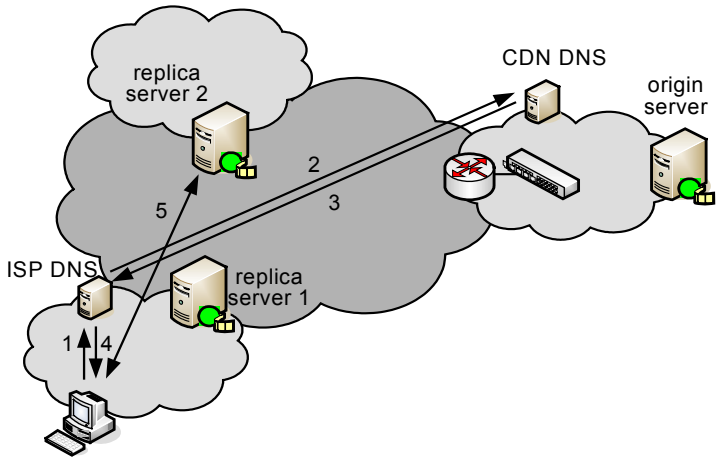


Figure 2.6 Request routing by means of DNS routing

2.3.2.e Centralized versus distributed management architectures

Traditional network management architectures are controled by a centralized system, in which most management communications are routed to one major server, typically located at the origin server site. Such a system has the benefit of concentrating most functionality in the systems main server. A benefit of centralization is the ease of maintaining accurately updated lists of data that can be easily accessed from all points. A major weakness is the single point of failure at the central component. When the system's server is put out of operation or becomes unreachable, either accidentally or through hostile action, the whole service management system fails.

In distributed architectures, the decision-making is dispersed closer to the actual point of service. The main reason why many content distribution services, particularly the popular services, migrate to more distributed management planes is the scalability issue. The manageability of the system has to remain feasible, independent of the volume of the user demand, the geographical dimensions or the number of independent organizational components involved in the service delivery model.

Due to the legal controversy on the introduction of peer-to-peer (P2P) technologies at the client side (see next section), only non-commercial file sharing applications make use of this concept. In the management plane however, P2P mechanisms are more wide-spread since they can offload the central entity from decisions on content placement (on servers or caches), server or cache selection, etc. In a next stage, P2P technologies can omit the need for an

origin server site completely, by also distributing the origin content over the different peers (distributed servers or caches) as well.

2.3.3 Peer branch

This section gives an overview of peer-to-peer (P2P) networks for content distribution. In P2P networks, the content is transferred between end users, each acting as server and/or client ("servent"). This concept has become very popular in file sharing networks like Napster [41], Gnutella [42], Kazaa [43], eDonkey [44] and BitTorrent [45].

An important goal in peer-to-peer networks is that all clients provide resources, including bandwidth, storage space, and computing power. Thus, as nodes arrive and the demand on the system increases, the total capacity of the system also increases. This is not true for server based architectures, in which adding more clients could mean slower data transfer for all users. To avoid free riding (by selfish nodes which only utilize other peers' resources without contributing to the network), different incentive mechanisms based on payments, reputation, score and trust have been studied [54].

The distributed nature of peer-to-peer networks also increases robustness in case of failures by replicating data over multiple peers, and, in pure P2P systems, by enabling peers to find the data without relying on a centralized index server. In the latter case, there is no single point of failure in the system.

Both pure and hybrid architectures build an overlay network over the existing IP network. In most cases this overlay is constructed arbitrarily, however it has been shown [46] that this generates a lot of expensive inter-domain traffic that can be reduced by intelligently building the overlay.

Since the actual data transfer is always done peer-to-peer, the classification of P2P networks is based on their control architecture: mediated by a central server, purely P2P or using a hybrid solution. The control architecture is responsible for different tasks:

- Maintenance of the network that consists mainly of ad-hoc connections
- Indexing of the content to allow searches
- Establishing connections between peers (using IP addresses, but taking care of firewall and Network Address Translation (NAT) issues)
- Providing authentication, autorization and accounting (AAA) or anonymity.

2.3.3.a Mediated peer-to-peer

Mediated P2P networks are managed by a central server (or cluster of servers). Clients have to connect to this server and upload a list of their shared content. Search requests can then be answered immediately by the central server.

Advantages of this architecture are the high level of control over the network and, due to the global network view, the fastest download speeds and shortest hopcount connections of all file sharing applications (see Appendix A).

The central server however is a single point of failure and the network is therefore also more vulnerable to law suits regarding transfer of copyright material. This is the reason why once very popular P2P networks like Napster and AudioGalaxy have ceased to exist[2].

2.3.3.b Pure peer-to-peer

Pure peer-to-peer networks do not make use of a central server, but rely on the peers to manage the self-organizing network. This means that all control messages and queries for files have to be flooded through the network. They are sent to all (or most) neighbour peers, with a decreasing time-to-live (TTL). The resulting overhead traffic, sometimes more than 10kB/s (see Appendix A), has made these pure P2P networks, like the early Gnutella network, very unpopular. Searches for less popular content are less optimal than with mediated architectures, due to the local view of the network (determined by the TTL). Advantages are the lack of a single point of failure and the increased user anonymity. FreeNet even succeeds in offering very strong anonymity using key based routing.

2.3.3.c Hybrid peer-to-peer

Most current file sharing applications use a hybrid, two-tier control architecture in their network. Some peers are dynamically promoted to ultrapeers, acting as a "mediated server" for a large number of leaf clients. The ultrapeers themselves are connected by a pure P2P network. There is still no single point of failure, the transfer speeds increase due to the global network view and the overlay traffic is limited to the ultrapeers with powerful PC's and high bandwidth Internet connections (see Appendix A).

Example networks using ultrapeers are Gnutella (in its current version) and FastTrack. On the eDonkey network, servers acting as communication hubs are

[2] Napster has been resurrected as a client-server file sharing application, much like Apple's iTunes.

present, allowing the users to locate files in the network. Likewise, BitTorrent clients contact trackers (servers keeping track of the clients sharing a particular file) using torrents (metadata files describing the file) listed on websites. Files are typically broken down into small pieces and transferred between all users of a "swarm", identified by the tracker.

2.4 Network support mechanisms

Besides the architectural evolutions, different network support mechanisms have been introduced to enhance the QoS or to reduce the bandwidth usage in the network.

2.4.1 Multicasting and broadcasting

Historically, broadcasting technologies have been widely used for radio (transmitted through the air) and television (transmitted through the air and through cable networks). Community Antenna Television (CATV) systems originally transmitted radio frequency (RF) signals from a headend to the subscribers over coaxial cable. In more recent cable TV deployments, the analogue signals are converted to compressed digital signals and transmitted over optical fiber before they reach the local coax trunks.

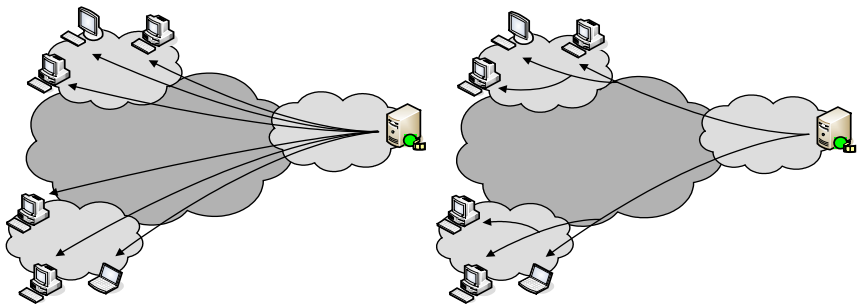


Figure 2.7 Unicast (a) versus multicast (b) delivery

Nowadays, most Internet data communication is unicast. In unicast content delivery, data is sent from a server to one client (with one destination IP address) over a network connection (Figure 2.7a). In broadcast [14] and multicast [15] content delivery, data is send to multiple clients (with one broadcast or multicast IP address) simultaneously (Figure 2.7b). While broadcasting is done to all nodes in a network, multicasting is directed to a selected set of destination nodes only.

Packets are sent only once over each link on the common path to all destination nodes. Only when the path splits for one or some of the destination nodes, packets are duplicated. Although the transmission of packets is still done using best-effort IP, a substantial reduction of the server load and network bandwidth usage can be attained. The membership of multicast groups is managed using the Internet group management protocol (IGMP) [16]. This communication protocol is used to exchange IP multicast group memberships among multicast routers. Authentication and accounting can be provided using the Internet group membership authentication protocol (IGAP) [17].

Most on-demand streaming media content, such as Video on Demand (VoD), is delivered as unicast traffic. For very popular content however, multicasting and broadcasting techniques can provide a more scalable delivery service. The requested video streams can be multicast periodically [18, 19], e.g. every few minutes, or after a sufficient number of requests have arrived, so that a near VoD service can be offered. As a consequence, clients may experience starting delays in case of periodical multicast sessions or so-called batching delays in the latter case. Several solutions to reduce these delays, as well as combinations of them [24], have been proposed and studied.

- Chaining [20] uses client-side storage capacity for the initial part of the stream and forwards this data to new clients entering the multicast group.
- Using the caching multicast protocol (CMP) [21] allows the routers on the multicast tree to intercept and cache video streams as they pass through.
- Patching [22] uses client-side storage capacity to keep up with an existing multicast channel and forwards this data using unicast patching streams.
- Piggy-backing [23] merges later multicast group members with existing members by slightly increasing the playback rate of the later users.

2.4.2 Traffic engineering

Traditionally, IP packets are routed over the shortest path, as determined by the routing tables in the intermediate network routers. In case of network congestion, the shortest path might not be the optimal path in terms of network QoS. Traffic engineering mechanisms, e.g. based on network monitoring, can choose longer paths, routing packets around congested areas (Figure 2.8b). Traffic engineering can be enforced using multiple routing tables. An example is multiprotocol label switching (MPLS) [25], where packets entering a router are first classified before they are forwarded. Packets are grouped into a forwarding equivalence class (FEC), based on their source and/or destination IP address, labeled and sent to the corresponding next hop. The labeling is done through additional shim headers (Ethernet) [8] or reuse of existing headers (ATM) [7].

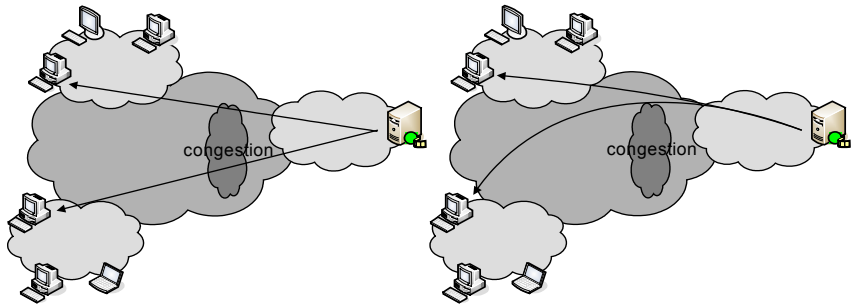


Figure 2.8 Shortest path routing (a) versus traffic engineering (b)

2.5 Supporting network technologies

Due to the growing popularity of the next-generation services, storage of multimedia content in the access network becomes more and more important. This section therefore describes the two main access network technologies: Digital Subscriber Line (DSL) and Hybrid Fiber Coax (HFC).

2.5.1 DSL-based access network architecture

DSL enables fast data transmission over copper telephone lines. Nowadays, Asymmetric DSL (ADSL) is the most common form, providing a greater bandwidth in the download direction than in the upload direction. Current download speeds vary from 512 Kbps up to 4 Mbps, while upload speeds are limited to 128 Kbps or 256 Kbps. There are both technical and economical reasons for this choice for bandwidth asymmetry. On the technical side, there is likely to be more frequency crosstalk from other circuits at the network side, which can be handled more effectively by the access multiplexer (DSLAM), than at the customer premises. The economical reason is that end users typically download more than that they upload, especially while Web browsing.

Common ADSL deployments are based on Asynchronous Transfer Mode (ATM) technology. ATM is a cell relay network protocol which encodes data traffic into small fixed-sized (53 byte; 48 bytes of data and 5 bytes of header information) cells instead of variable sized packets. Because of the relatively low data-rate (compared to optical backbone networks), ATM is an appropriate technology for multiplexing time-critical data such as streaming media (traditionally voice traffic) with less time-critical data such as web traffic. In a triple play scenario, different ATM virtual circuits (VCs) may be allocated for different services.

More recently however, network operators are increasingly moving away from ATM towards Ethernet-based solutions. Important reasons for this migration are

cost savings and the possibility of removing the older and more expensive ATM network.

2.5.2 HFC-based access network architecture

Broadband Internet access is not only available through DSL, using available bandwidth on the copper telephony network, but through the unused bandwidth of the cable television network as well. Download speeds (1 to 10 Mbps) and upload speeds (192 to 512 Kbps) are generally slightly higher than for DSL based Internet access. The available bandwidth for one neighbourhood however is shared on a single coaxial cable line, therefore connection speeds can vary depending on how many people are using the service simultaneously. The term HFC is derived from the combination of coax trunks, each with 100 up to 2000 homes, with fiber optical equipment. Optical nodes convert the optical signals to electrical and vice versa. Fiber optic cables then connect these optical nodes to distant head ends.

The communications and operation support interface requirements for HFC access networks are defined by the data over cable service interface specification (DOCSIS) [47] from CableLabs [48]. A DOCSIS architecture includes two primary components: a cable modem (CM) located at the customer premises, and a cable modem termination system (CMTS) located at the Community Antenna Television (CATV) headend. A typical CMTS is a device which hosts downstream and upstream ports, which is functionally similar to the DSLAM used in DSL systems.

2.6 Service specific solutions

The previous overview of content distribution architectures and networking mechanisms shows a variety of technological alternatives for streaming content delivery. Depending on the offered service, existing network infrastructure and network architecture, different solutions can be deployed.

2.6.1 Streaming services

In this book the network design and replica placement for several next-generation streaming services is presented. This section describes these services and identifies their most important characteristics, from a network perspective as well as from the user's point of view.

2.6.1.a Video on demand

Similar to traditional video rental stores, a video on demand (VoD) service offers films or television programs to end users. As this service typically supports user interactivity (pause, fast forward, rewind), videos are sent over individual

(unicast) connections to each user. To enhance the quality of experience (QoE) for the end user, i.e. to reduce the delay and jitter, content is typically located close to the end user. Less popular videos however are often stored deeper in the network (thus requiring fewer replicas), to limit the associated storage costs for the content provider.

Some content providers make a distinction between standard interactive video on demand (iVoD) and near video on demand (nVoD). nVoD services deliver the video content through periodic multicast, e.g. every 5 minutes. This way the end users may experience startup delays. The major benefit of nVoD is the reduced network traffic, especially in case of popular content. User interactivity for nVoD services is very limited compared to the unicast iVoD.

Network design for video on demand is discussed in Chapters 3 and 4.

2.6.1.b Broadcast television

Recent broadcast television services are delivered digitally over the Internet. Since TV channels are broadcasted in the network to all end users, IPTV servers are typically located close to the clients to limit the bandwidth usage. Besides standard broadcast mechanisms, switched broadcast techniques are used as well. TV channels that are delivered through switched broadcast are broadcast over a part of the network, e.g. up to the edge of the access network, and then forwarded locally as unicast traffic, only if those TV channels are requested locally.

Network design for broadcast television is discussed in Chapter 6.

2.6.1.c Time-shifted television

To support interactivity for broadcast television, a time-shifted television (tsTV) service can be deployed. tsTV enables the end user to watch a broadcasted TV program with a time-shift, i.e. he can start watching the TV program from the beginning although the broadcasting of that program has already started are is already finished. Pause or rewind commands can be supported as well. While user interactivity for iVoD is supported through unicast connections to each user, our tsTV solution uses storage space on proxy caches to provide this service.

Network design for broadcast television is discussed in Chapter 5.

2.6.1.d Multimedia production and collaboration

Recently, multimedia production companies have shifted from tape based to harddisk based content storage and aim at sharing storage space and computational resources between multiple multimedia production sites and corporate users. Very high bandwidth streaming and stringent QoS requirements

imply the need for specialized Grid technologies and bandwidth management solutions.

Network design and bandwidth management for MediaGrids are discussed in Appendix D.

2.6.2 Overview on service solutions

In this section, the different technological alternatives applied in the service specific solutions are discussed. Tendencies towards the decentralization of the management plane and the introduction of access network service enablers confirm the need for more scalable solutions with enhanced QoS.

	Video on Demand **(Ch. 3 & 4)**	Broadcast / time-shifted TV **(Ch. 5 & 6)**	Fast multimedia content retrieval **(App. D)**
Distributed servers	X	X	X
Caches	X	X	
Proxy components		X	
Peer-to-peer collaboration	X	X	

Table 2.4: Overview of the technological alternatives for different use cases.

Table 2.4 gives an overview of the service specific solutions presented in this dissertation and the technological choices made.

While the network design for video on demand (in Chapter 3) still focuses on the metro and DSL-based access network part, the time-shifted television service (in Chapter 5) is presented as a pure DSL access network solution. More detail on the access network architecture for time-shifted television can be found in Appendix B.

Chapter 4 focuses on the metro and HFC access network design for video on demand. Gigabit Ethernet (GbE) over Wavelength Division Multiplexing (WDM) technologies are brought into play. Broadcast IP television services for

HFC access network deployment are studied in Chapter 6, where switched broadcast technologies are used to reduce the bandwidth requirements from traditional standard broadcast mechanisms.

As the project on bandwidth management for fast multimedia content retrieval on MediaGrids is still ongoing, the current work is detailed in an internal document in Appendix D.

References

[1] A. S. Tanenbaum, "Computer Networks", fourth edition. Prentice Hall, 2003.

[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, Hypertext transfer protocol - http/1.1, IETF RFC 2616, June 1999.

[3] D. Austerberry, "Technology of Video and Audio Streaming", Focal press, 2003.

[4] J. Postel, Transmission control protocol - darpa internet program protocol specification, IETF RFC 793, September 1981.

[5] J. Postel, User datagram protocol, IETF RFC 768, August 1980.

[6] J. Postel, Internet protocol, IETF RFC 791, September 1981.

[7] M. Laubach and J. Halpern, Classical ip and arp over atm, IETF RFC 2225, April 1998.

[8] C. Hornig, Standard for the transmission of ip datagrams over ethernet networks, IETF RFC 894, April 1984.

[9] J. Postel, Simple mail transfer protocol, IETF RFC 821, August 1982.

[10] J. Myers, C. Mellon and M. Rose, Post office protocol version 3, IETF RFC 1939, May 1996.

[11] M. Crispin, Internet message access protocol version 4rev1, IETF RFC 3501, March 2003.

[12] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, Rtp: A transport protocol for real-time applications, IETF RFC 3550, July 2003.

[13] H. Schulzrinne, A. Rao, and R. Lanphier, Real time streaming protocol (rtsp), IETF RFC 2326, April 1998.

[14] J. Mogul, Broadcasting internet datagrams, IETF RFC 919, October

1984.

[15] S. Deering, Host extensions for ip multicasting, IETF RFC 1112, August 1989.

[16] B. Fenner and D. Meyer, Multicast source discovery protocol (msdp), IETF RFC 3618, October 2003.

[17] T. Hayashi, D. Andou, H. He, W. Tawbi, and T. Niki, Internet group membership authentication protocol (igap), Internet Draft, February 2004.

[18] C. Aggarwal, J.Wolf, and P. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems", IEEE International Conference on Multimedia Systems, June 1996.

[19] J. Paris, S. Carter, and D. Long, "Efficient broadcasting protocols for video on demand", SPIEs Conference on Multimedia Computing and Networking, January 1999.

[20] S. Sheu, K. Hua, and W. Tavanapong, "Chaining: a generalized batching technique for video-on-demand", the International Conference On Multimedia Computing and System, June 1997.

[21] K. Hua, D. Tran, and R. Villafane, "Caching multicast protocol for ondemand video delivery", SPIEs Conference on Multimedia Computing and Networking, January 2000.

[22] Y. Cai and K. Hua, "Sharing multicast videos using patching streams", Multimedia Tools and Applications, 21(2):125–146, November 2003.

[23] L. Golubchik, J. Lui, , and R. Muntz, "Adaptive piggybacking: a noval technique for data sharing in video-on-demand storage servers", ACM Multimedia Systems, 4(3):140–155, 1996.

[24] Ma Huadong and Kang G Shin, "Multicast video-on-demand services", ACM SIGCOMM Computer Communication Review, 32(1):31–43, January 2002.

[25] B. Davie and Y. Rekhter, "Mpls: technologies and applications", Academic press, 2000.

[26] K. Egevang and P. Francis, The ip network address translator (nat), IETF RFC 1631, May 1994.

[27] A. Vakali and G. Pallis, "Content delivery networks: Status and trends", IEEE Internet Computing, 7(6):68–74, November 2003.

[28] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts", 7th Edition. John Wiley and Sons, Inc., 2005.

[29] A. Mahanti, C. Williamson, and D. Eager, "Traffic analysis of a web proxy caching hierarchy", IEEE Network, 14(3):16–23, May/June 2000.

[30] D. Wessels and K. Claffy, Internet cache protocol (icp) version 2, IETF RFC 2186, September 1997.

[31] D. Wessels and K. Claffy, Application of internet cache protocol (icp) version 2, IETF RFC 2187, September 1997.

[32] D. Wessels and K. Claffy, "Icp and the squid web cache", IEEE Journal on Selected Areas in Communication, 16(3):345–357, April 1998.

[33] M. Karlsson and M. Mahalingam, "Do we need replica placement algorithms in content delivery networks?", In Seventh International Web Content Caching and Distribution Workshop, August 2002.

[34] J. Kangasharju, J. Roberts, and K. Ross, "Object replication strategies in content distribution networks", In Proceedings of Sixth International Workshop on Web Caching and Content Delivery, June 2001.

[35] M. Karlsson, C. Karamanolis, and M. Mahalingam, "A framework for evaluating replica placement algorithms", In Technical Report, HP Laboratories, July 2002.

[36] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, Known content network (cn) request-routing mechanisms, IETF RFC 3568, July 2003.

[37] G. Montenegro, Reverse tunneling for mobile ip (revised), IETF RFC 3024, January 2001.

[38] P. Mockapetris, Domain names - concepts and facilities, IETF RFC 1034, November 1987.

[39] P. Mockapetris, Domain names - implementation and specification, IETF RFC 1035, November 1987.

[40] D. Eastlake and A. Panitz, Reserved top level dns names, IETF RFC 2606, June 1999.

[41] Napster. http://www.napster.com/.

[42] Gnutella. http://www.gnutella.com/.

[43] Kazaa. http://www.kazaa.com/.

[44] eDonkey. http://www.edonkey.com/.

[45] Bittorrent. http://www.bittorrent.com/.

[46] M. Ripeanu, A. Iamnitchi and I. Foster, "Mapping the Gnutella network", IEEE Internet Computing, February 2002.

[47] DOCSIS. http://www.docsis.org/.

[48] CableLabs. http://www.cablelabs.com/.

[49] R. Stewart et al, Stream Control Transmission Protocol, IETF RFC 2960, October 2000.

[50] Akamai, http://www.akamai.com.

[51] Mirror Image, http://www.mirror-image.com.

[52] Edgestream, http://www.edgestream.com.

[53] Edge Side Includes, http://www.esi.org.

[54] P. Chen, T. Li and C. Laih, "On the performance evaluation in P2P networks with free riders", in proceedings of TFIT 2006, Nancy, France, March 2006.

3 Network design and replica placement for video on demand

3.1 Introduction

As stated in the previous chapter, the quality of service offered by single server (or single site cluster) architectures is often insufficient for popular multimedia websites or streaming services such as video on demand. Overloaded servers and congested transport networks degrade the user experience noticeably. A growing number of content providers therefore benefits from content distribution services offered by companies like Akamai [1]. They use high capacity overlay networks in combination with surrogate servers at the edge of the Internet to deliver their bandwidth-intensive content. Consequently, the central server is offloaded, and the latency and network traffic reduced.

The development of static (*offline*) placement strategies for content replicas further enhances CDN performance [4, 5, 6, 7, 13]. These algorithms decide where to replicate specific content, in order to reduce bandwidth consumption and latency at low infrastructure usage costs. Content server selection algorithms then direct users to the appropriate surrogate server, offering the best achievable quality of service.

Inspired by the extremely popular peer-to-peer file sharing applications, P2P technologies at the level of the surrogate servers have been introduced. This makes direct communication between them possible, so that information on local traffic patterns can be exchanged. By using these P2P architectures in CDNs [8], a more robust, scalable and efficient service can be provided. The distributed replica placement and retrieval algorithms, to be executed by all cache nodes independently, can now dynamically (*online*) adapt to new content supplied to the CDN, to a changing user request pattern or to varying network conditions.

This chapter studies offline as well as online replica placement strategies, applicable on general network topologies. In order to be able to compare both approaches to an analytical model, the presented experiments are performed on ring topologies, which are widely used in recent CDN deployments. The efficiency of the proposed RPAs is also validated on more general topologies.

This chapter is divided into two main parts: sections 3.3, 3.4 and 3.5 present a centralized and static (*offline*) approach for the replica placement problem, while sections 3.6, 3.7, 3.8 and 3.9 propose a distributed and dynamic (*online*) replica placement strategy. The centralized and static solution is used for the network design: it determines the server placement and the capacity required on these servers and on the network links. In case a prediction on the demand pattern from the end users is available, an initial content placement can be calculated using this approach. The distributed and dynamic solution is brought into play when the service is active. It dynamically adapts the content placement to changing network conditions and variations in the user demand, based on local information. The simulations for the centralized ILP solution have been performed using Cplex [19], the distributed algorithms have been evaluated using a self-made discrete event simulator (written in C++). Cplex implements a variety of branching and node selection techniques, including cuts and heuristics. For the simulations presented in this work, the Cplex Mixed Integer Programming option was used, combining integer and binary variables.

We start the discussion with an overview of related work on replica placement in CDNs in section 3.2. Section 3.3 introduces an analytical solution technique and an ILP formulation for the centralized approach on networks without topology constraints. The main cost factors to be minimized are the overall network bandwidth consumption and storage costs. Both models are compared for a ring based CDN topology in section 3.4. An approximation of the analytical solution will lead to basic design rules for ring based CDNs. The analytical solution can also be used for a larger network topology, including a tree-like access network as in typical DSL based deployments, as studied in section 3.5.

Two distributed heuristics are proposed in section 3.6. Contrary to the computationally heavy centralized solutions (this problem is NP-complete, as proven in [7]), these algorithms can be executed on networks with more complex topologies. Furthermore, they are able to dynamically adapt the replica placement to changing user demands, varying network occupations or new content added to the network. Simulation results for these heuristics on a network with a ring topology are compared to results from the static ILP formulation. In section 3.7, these distributed algorithms are extended to dynamically support load balancing and avoid congestion in the network. Simulations on a more general network topology are presented in section 3.8, while section 3.9 compares our algorithms to standard heuristics on the same topology. The last section presents some general conclusions.

3.2 Related work

The advantages of replica placement algorithms over typical caching strategies have been studied in [5] and various RPAs have been proposed in recent studies [4, 13]. A detailed overview of the available models and algorithms as well as a framework for evaluating them is given in [6]. While most models have a similar cost function (optimizing bandwidth and/or storage usage costs for a given request pattern), less attention has been given to network constraints (limited link or storage capacities). Furthermore, a large part of these algorithms are designed for specific network topologies only (e.g. tree topology [14]). The possible use of these algorithms to reduce the server load or to avoid network congestion has not been given much consideration either. The benefits of adding workload information to placement algorithms are studied in [4]. Although [7] and [8] show that the introduction of peer-to-peer systems in content delivery networks has a potential to further improve the network performance, few developments have been made on distributed replica placement algorithms. These studies on RPAs in CDNs [4, 11, 12] and unicast streaming CDNs [10], as well as similar work on proxy caching techniques [18] also show that greedy algorithms that take distance metrics and content popularity into account perform better than more straightforward heuristics such as *LRU* (Least Recently Used) or *LFU* (Least Frequently Used).

Our work deals with the aforementioned shortcomings, in order to offer a replica placement solution that enhances the service quality at low network costs, for different topologies.

3.3 General static problem formulation

As stated in the introduction of this chapter, we start with handling the static problem, i.e. the content delivery system is in steady state, from a centralized point of view. This means that the set of requested objects (streams) does not change, and the request rates of these objects also remain constant. We start with the analytical formulation for general network topologies and present an ILP formulation that takes network constraints into account afterwards.

3.3.1 Analytical formulation

Let $\mathbf{O} = \{o_1, \ldots, o_F\}$ denote the object set which is offered for download to the users, and let r_i denote the total number of requests for object o_i during the period $[0,T]$. The size (measured in bytes) of object o_i equals s_i, the streaming bitrate is b_i. The infrastructure to host the content set **O** is characterized by a graph **G**, consisting of a set of vertices **V** (of size N), which are interconnected by a set of edges **E**. For the time being, we assume that the edges are not congested and able to carry the traffic generated in the content delivery network.

Given **G**, **O**, r_i, s_i and b_i, the problem now is to find the optimal set of surrogate servers $\mathbf{S_i}$ (with cardinality $|\mathbf{S_i}| = n_i$) for each object o_i by optimizing the cost associated with both transmitting and storing this particular object at the surrogate servers. When we define C_T as the cost to transmit one unit over one link and C_S as the cost to store one unit of content, the input parameter $\alpha = C_S / C_T$ indicates the relative trade-off between storage and transport costs. Without loss of generality, C_T can be set to one, so that the cost C_i, incurred by storing and streaming the object o_i in the CDN, is then given by

$$C_i = b_i r_i \overline{d_i(\mathbf{S_i})} + \alpha s_i n_i \qquad (1)$$

using the average distance on the shortest path between requestor and node serving the request. This average distance can either be a simple hop count or a more sophisticated sum of individual link costs, and is a function of the set $\mathbf{S_i}$. The first term in equation (1) will be referred to as *transport cost*, while the second term will be called *storage cost*. Note that no infrastructure cost is taken into account. In principle, since the server disk capacity is assumed unlimited, as well as link bandwidth, the total cost

$$C = \sum_{i=1}^{F} C_i \qquad (2)$$

can be optimized by minimizing each of the C_i independently. This minimization can be done straightforwardly by an exhaustive strategy. However, it is clear that solving the problem this way becomes computationally unfeasible for large values of N, since C_i should be calculated for each subset of **V** (excluding of course the empty set, since each object should be available at at least one location, so n_i>0). However, one can easily show $C_i(n_i)$ to have a single minimum (because the transmission part monotonically decreases as a function of n_i, while the storage contribution obviously increases), and therefore, when considering increasing values of n_i (starting with n_i=1), the search comes to an end as soon as C_i increases. This all leads to the rather simple algorithm to calculate the optimal surrogate server location sets $\mathbf{S_i}$ and associated minimal costs presented in Figure 3.1.

1. For each $i = 1 .. F$
   - 1.1. $C_i(0) = \infty$, n_i=1
   - 1.2. Find $\mathbf{S_i}(n_i)$ yielding minimal cost, i.e.

$$C_i(n_i) = \min_{\substack{|\mathbf{S_i}| = n_i \\ \mathbf{S_i}(n_i) \subset \mathbf{V}}} b_i r_i \overline{d_i(\mathbf{S_i}(n_i))} + \alpha s_i n_i$$

   - 1.3. If n_i=N or $\Delta^{-} C_i(n_i) = C_i(n_i) - C_i(n_i - 1) > 0$, let $\mathbf{S_i} = \mathbf{S_i}(n_i\text{-}1)$

     Else, increase n_i by 1 and repeat step 1.2
2. Optimal surrogate server locations $\mathbf{S_i}$ found for all i=1 .. F,

   calculate minimal cost from

$$C = \sum_{i=1}^{F} (b_i r_i \overline{d_i(\mathbf{S_i})} + \alpha s_i n_i)$$

Figure 3.1: Exhaustive strategy to calculate optimal surrogate server location sets for replica placement for a set of objects $\boldsymbol{O}=\{o_1, \ldots, o_F\}$*, characterized by request rates* r_i

Based on the results found using this procedure, the CDN can be dimensioned (surrogate server sizes and link bandwidth). Of course, if not all nodes of the physical network are eligible for storing content, one removes these nodes from the set **V**, ensuring that the value d_i is calculated correctly.

In general, solving the CDN optimization problem using this procedure is complex, mainly due to the complicated structure of the function $d_i(\mathbf{S_i})$, which both depends on network topology (**G**) and request patterns (request rates from each end user location for each file). For regular topologies and request patterns, and more specifically ring networks with uniform user behavior, it will be shown in section 4 that the optimization problem can be solved analytically.

3.3.2 ILP-problem formulation

The algorithm shown in Figure 3.1 has several drawbacks, besides of being computationally intensive for large networks, as it has an exponential complexity. No limitations on surrogate server sizes, nor on bandwidth usage are taken into account, and hence the procedure is not suited for optimizing resource usage on an already installed infrastructure. To overcome these shortcomings, an Integer Linear Programming (ILP) formulation is presented in this section.

3.3.2.a Network parameters

Every edge e from **E** has a cost parameter c_e (e.g. a delay penalty) and a maximum bandwidth capacity u_e. All nodes n from **V** have a storage capacity m_n (higher than 0 for the cache nodes $\mathbf{A} \subset \mathbf{V}$). The cost to store an object o from **O** is equal to the size s_o of the object. Apart from a size s_o, every object also has a fixed bitrate b_o. This may correspond to the constant bit rate of a streaming file for a Video on Demand service. The requests rates $r_{n,o}$ from the user nodes n out of $\mathbf{D} \subset \mathbf{V}$ for each of the objects are given. These rates also reflect the popularity of the objects to the users. We also define $\mathbf{D_o} \subset \mathbf{D}$ as all the users requesting object o.

The main variables in the objective function are the transport variables $h_{e,d,o}$ and the storage variables $z_{n,o}$:

- $h_{e,d,o}$ is 1 if edge e is used to deliver object o to destination d, 0 otherwise
- $z_{n,o}$ is 1 if node n is used to cache object o, 0 otherwise

There is one set of auxiliary variables $x_{n,d,o}$:

- $x_{n,d,o}$ is 1 if node n is used to cache object o for destination d, 0 otherwise

We define I_n as the set of incoming edges of node n, O_n as the set of outgoing edges.

3.3.2.b Objective function

Now that all symbols and variables are explained, the objective function F can be expressed as follows:

$$F = \sum_{o \in O} \sum_{d \in D_o} \sum_{e \in E} c_e b_o r_{d,o} h_{e,d,o} + \alpha \sum_{o \in O} \sum_{n \in V} s_o z_{n,o} \qquad (3)$$

The objective function has to be minimized and consists of two parts, the *transport cost* and the *storage cost*:

- The first part of formula (3) is the *transport cost*. It is the cost related to the use of bandwidth. If edge e is used to transport object o to destination node d (or in other words if $h_{e,d,o}$ is 1) then there is a cost of $c_e b_o r_{d,o}$ associated with that use.
- The second part is the *storage cost*. It defines the cost for caching object o in node n as s_o (the size of object o), if $z_{n,o}$ is 1.

In order to be able to emphasize on the importance of one of these costs, a parameter α is introduced to linearly combine both costs. If α is zero, only the transport cost is considered to be important. If α is high the storage cost is more important and the solution found will have only few cached files.

Note that equation (3) only includes general transport and storage costs. Other cost components such as operational costs, RAM versus disk storage costs, video pump costs, etc. are not taken into account to reduce the ILP complexity and calculation times.

3.3.2.c Constraints

Capacity constraints

$$\sum_{o \in O} \sum_{d \in D_o} b_o h_{e,d,o} \leq u_e \qquad \forall e \in E \qquad (4)$$

$$\sum_{o \in O} s_o z_{n,o} \leq m_n \qquad \forall n \in V \qquad (5)$$

Constraint (4) imposes a restriction on the total flow through the edges. This flow can not exceed the capacity of the edge. Constraint (5) imposes a restriction on the amount of cached content in a certain node. This cost must not exceed the capacity of that node. There are $|\mathbf{E}| + |\mathbf{V}|$ capacity constraints.

Auxiliary constraints

$$x_{n,d,o} \le z_{n,o} \qquad \forall d \in D_o, \forall o \in O, \forall n \in V \qquad (6)$$

This auxiliary constraint takes care of the relationship between $x_{n,d,o}$ and $z_{n,o}$. Constraint (6) indicates that if node *n* stores object *o* (i.e. $z_{n,o}$ is one), this can be done for multiple destinations (i.e. $x_{n,d,o}$ can be one for several destination nodes $o \in D_o$).

Flow conservation constraints

$$\sum_{e \in I_n} h_{e,d,o} = \sum_{e \in O_n} h_{e,d,o} \qquad \forall n \in V \setminus d \setminus A, \forall d \in D_o, \forall o \in O \qquad (7)$$

$$x_{n,d,o} + \sum_{e \in I_n} h_{e,d,o} = \sum_{e \in O_n} h_{e,d,o} \qquad \forall n \in A, \forall d \in D_o, \forall o \in O \qquad (8)$$

$$\sum_{e \in I_n} h_{e,d,o} = 1 \qquad \forall n \in D_o, d = n, \forall o \in O \qquad (9)$$

$$\sum_{e \in I_n} h_{e,d,o} = \sum_{e \in O_n} h_{e,d,o} \qquad \forall n \in D, \forall d \in D_o, d \ne n, \forall o \in O \qquad (7')$$

These constraints regulate the flows between source and destination nodes. Constraint (7) ensures the traffic through "normal" nodes (nodes that are not cache or destination nodes), constraint (8) takes care of cache nodes and constraint (9) is for destination nodes.

Constraint (7) indicates that node *n* should let incoming data from object *o* for destination *d* pass through to the next node on the path. Constraint (7') does the same for destination nodes acting as normal nodes (e.g. destination nodes laying on the path towards other destination nodes). Constraint (8) ensures that cache node *n* should let incoming data from object *o* for destination *d* pass through to the next node on the path, except when it is the source node for that download (then $x_{n,d,o}$=1 and $h_{e,d,o}$=0 on all incoming edges). Constraint (9) indicates that destination node *n* should receive the object he requested on one of his incoming edges.

Binary constraints

$h_{e,d,o}$ binary $\qquad \forall e \in E,\ \forall d \in D_o, \forall o \in O \qquad (10)$

$z_{n,o}$ binary $\qquad \forall n \in V,\ \forall o \in O \qquad (10')$

$x_{n,d,o}$ binary $\forall n \in V\ ,\ \forall d \in D_o\ , \forall o \in O$ (10'')

Constraint (10) imposes that all variables are binary.

Additional constraint

$$\sum_{e \in E} h_{e,d,o}\, p_e \leq p_{\max} \qquad \forall d \in D_o\ , \forall o \in O \qquad (11)$$

Constraint (11) can be used as an additional constraint to set a maximum penalty on a parameter for each object stream. Examples are restrictions on the total delay (p_e represents the delay on link e) or the hopcount (p_e = 1), for the total path of a stream.

Robustness constraint

$$\sum_{n \in A} z_{n,o} \geq f + 1 \qquad \forall o \in O \qquad (12)$$

Restriction (12) adds robustness to the content delivery service. Every file should at least have f+1 different locations in the network, with f the maximum number of simultaneously failing caches.

3.4 Network design for ring based CDNs

In this section, we make a comparison between the analytical and the ILP model for a CDN with a ring topology and determine a set of network design rules. The ring network consists of N nodes that are all candidate surrogate servers (Figure 3.2). A total of F objects $\{o_1, \ldots, o_F\}$ are available for a certain period [0, T]. During that period, a total of R requests are made, with r_i requests for object o_i. We make the additional assumption that all users are connected through an access network link to one of the N ring nodes. Since this access network is assumed given, the transport cost on these access links can not be optimized. Furthermore, we assume that all requests are equally spread over the N nodes during the given time interval. Since the number of possible replica placements or routes is limited on a ring network, the centralized approach is still scalable for larger values of N. On more complex topologies, the scalability of the ILP solution is very limited (examples are presented in [9, 17]). The distributed solution presented later on in this chapter however is very scalable, since only local traffic patterns are taken into account.

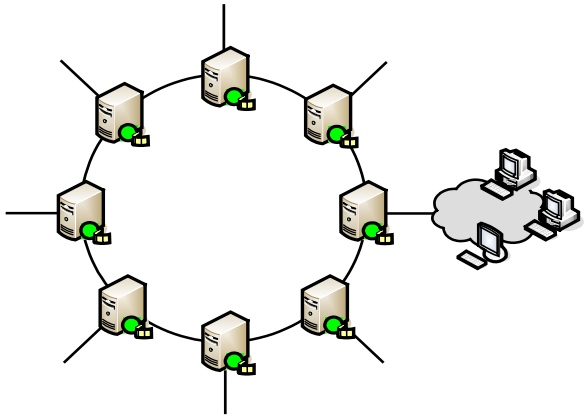


Figure 3.2: Ring network with access network links

3.4.1 Analytical solution

Due to the symmetry of the problem, it is obvious that for a given number of surrogate servers, an optimal location is achieved by maximizing the distance between individual surrogate servers. For N nodes and n surrogate servers, and q the remainder of dividing N by n, such that

$$N = \left\lfloor \frac{N}{n} \right\rfloor n + q = d_0 n + q \,, \tag{13}$$

we have n-q nodes serving requests aggregated by d_0 ring nodes, and q nodes serving d_0+1 nodes (of course assuming shortest path routing). Note that this observation yields an optimal set $\mathbf{S_i}$ for a given n_i, thereby avoiding the optimization step of the algorithm presented in Figure 3.1 (more specifically step 1.2). The following analytical expression for $\overline{d_i(n)}$ can easily be derived:

$$\overline{d_i(n)} = \left\lfloor \frac{d_0 + 1}{2} \right\rfloor \left(1 - \frac{n}{N} \left\lfloor \frac{d_0 + 1}{2} \right\rfloor \right) \tag{14}$$

Given this expression, the procedure given in Figure 3.1 can be considerably simplified as follows (see Figure 3.3), observing that

$$\begin{aligned}\Delta^{-}C_i(n_i) &= C_i(n_i) - C_i(n_i - 1) > 0 \\ &= b_i r_i \left[\overline{d_i(n_i)} - \overline{d_i(n_i - 1)}\right] + \alpha s_i \\ &= b_i r_i \Delta^{-} \overline{d_i(n_i)} + \alpha s_i\end{aligned} \tag{15}$$

1. For each $i = 1 \,..\, F$

   Calculate $n_i = \underset{\substack{\Delta^{-}C_i(n)>0 \\ 0<n\le N}}{\mathbf{max}}\ (n)$

2. Optimal server locations found for all i=1 .. F, calculate minimal cost from

$$C = \sum_{i=1}^{F} (b_i r_i \overline{d_i(n_i)} + \alpha s_i n_i)$$

Figure 3.3: Exhaustive strategy to calculate optimal surrogate server location sets for replica placement for a set of objects $\boldsymbol{O}=\{o_1, \ldots, o_F\}$*, characterized by request rates* r_i*, on a ring based CDN*

Note that this procedure is valid for *any* problem (topology and request pattern) where $d_i(n)$ is only a function of the size of $\mathbf{S_i}$, and where the optimal surrogate server location $\mathbf{S_i}$ can be found directly from n_i. This is for example not the case when the user demand is asymmetrical. In that case, this analytical solution offers an upper limit for the total cost, since the replicas of an object can be placed closer to the users requesting that object than for a symmetrical user demand.

In Figure 3.4 this analytical solution is compared to the ILP solution for a ring network with $N = 8$ surrogate servers, serving a total of $R = 10000$ requests for $F = 20$ objects. These objects have a Zipf-like popularity, i.e. the popularity of the i^{th} most popular item is proportional to $i^{-\beta}$, with typical values for β between 0.5 and 1.0 [2, 3]. Small differences in transport and storage cost are visible, but the total cost (transport cost + $\alpha \cdot$ storage cost) is identical in both cases. The reason for the small differences is the non-unique character of both terms, due to the steps in their cost functions [9]. When α is sufficiently low (relatively low storage cost), all 20 objects are replicated on all surrogate servers (storage cost = $F \cdot N = 160$ units) and the transport cost is limited to the fixed streaming cost on

the access links (transport cost = $R \cdot d(n)$ = 10000 units, since the average distance $d(n)$ on the access network is 1). When storage is relatively expensive (α is high), every object is found on only one surrogate server (storage cost = $F \cdot 1$ = 20 units) and the transport cost reaches its maximum value (transport cost = $R \cdot d(n)$ = 30000 units, since the average distance $d(n)$ is 3: 1 for the access network link plus 2 for the average distance on the ring network with 8 surrogate servers). Note that the transport and storage costs are much more sensitive to small changes to the input parameter α for lower values ($\alpha < 100$).

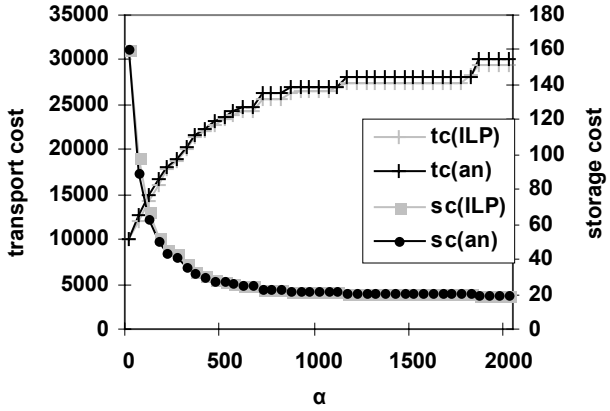


Figure 3.4: Transport (tc) and storage (sc) cost in a ring network with 8 surrogate servers (20 files available, $\beta = 0.7$), for both the analytical and the ILP solution

An approximation n_i' to the optimal value n_i can be found by solving

$$\Delta^{-} C_i(n_i') = 0 \tag{16}$$

in the interval [1,N]. If (16) has no solution in this interval, either n_i'=1 or n_i'=N, depending on the sign of $\Delta^{-} C_i(1)$, is the optimal solution. When we assume that each object has the same size s_i and streaming bandwidth b_i, we can set s_i and b_i to one without loss of generality. Solving (16) is then clearly equivalent to solving

$$\Delta^{-} \overline{d_i(n_i')} = -\frac{\alpha}{r_i}. \tag{17}$$

In the specific case of a ring based CDN, we can find an additional estimate n_i'' for n_i, by approximating the function $\overline{d_i(n)}$ (equation (14)) by

$$\overline{d_i'(n)} = \frac{N^2 - n^2}{4Nn} \tag{18}$$

which is obtained by replacing all integer divisions by their real valued counterparts. This allows to find

$$\Delta^- \overline{d_i'(n)} = -\frac{1}{4N} - \frac{N}{4n(n-1)} \tag{19}$$

and solving now

$$\Delta^- \overline{d_i'(n_i'')} = -\frac{\alpha}{r_i} \tag{20}$$

(which is a quadratic equation in n_i'') gives the following approximation for n_i:

$$\frac{1}{2}\left[1 + \sqrt{1 + \left[\frac{\alpha}{Nr_i} - \frac{1}{4N^2}\right]^{-1}}\right]. \tag{21}$$

Since however n_i should be an integer value, satisfying $\Delta^- C_i(n_i) < 0$, and because approximation (21) implies $\Delta^- C_i(n_i) = 0$, we expect the approximation (21) to be systematically too large. More specifically, since we should round (21) down to the smaller integer value, this estimate is on average too large by 0.5, giving the following improved estimate for n_i

$$n_i'' = \frac{1}{2}\sqrt{1 + \left[\frac{\alpha}{Nr_i} - \frac{1}{4N^2}\right]^{-1}} \tag{22}$$

One further notices that apparently the optimal number of surrogate servers for each object is strongly dependent on the parameter α/Nr_i, and to a lesser degree the network size N itself. If the expression in the right hand side of equation (21) gives results exceeding N, of course the limit value N is taken as estimate for n_i. Similarly, since at least one copy of each object should be stored in the network, the value one is taken as approximation for n_i in case (22) yields values smaller than one. More explicitly

$$n_i'' = \begin{cases} N & \frac{\alpha}{Nr_i} < \frac{1}{4N^2} + \frac{1}{4N^2 - 1} \\ 1 & \frac{\alpha}{Nr_i} > \frac{1}{4N^2} + \frac{1}{3} \\ \frac{1}{2}\sqrt{1 + \left[\frac{\alpha}{Nr_i} - \frac{1}{4N^2}\right]^{-1}} & otherwise \end{cases} . \quad (23)$$

3.4.2 Design rules for ring based CDNs

In this section, the results obtained above for pure ring based CDNs are used to dimension both storage space and network capacity. To arrive at numerical results for these values, we again assume a Zipf-like distribution to describe the relative object popularity, i.e.

$$r_i \propto \frac{1}{i^{\beta}}, 1 \le i \le F . \quad (24)$$

Let the total amount of requests (i.e. requests from all users during the interval $[0,T]$) be R, giving

$$r_i = R\frac{i^{-\beta}}{\sum_{i=1}^{F} i^{-\beta}} = \frac{R}{A} i^{-\beta}, 1 \le i \le F \quad (25)$$

Large β values indicate a relatively small set of extremely popular objects, leading to less storage space requirements at the surrogate servers.

3.4.2.a Storage capacity

The total storage capacity needed in the ring network (s), in case the total cost is optimized, can be calculated from

$$s = \sum_{i=1}^{F} s_i n_i \tag{26}$$

Assuming no correlation between object size and popularity, and denoting the average object size as $\bar{s}$, this becomes

$$s = \bar{s} \sum_{i=1}^{F} n_i \approx \bar{s} \sum_{i=1}^{F} n_i'' \tag{27}$$

If object size and popularity are correlated (smaller files are typically downloaded more often), the problem could be approximated as a supersposition of smaller sub-problems, each dealing with fixed-size objects (music files, video files, …).

In order to calculate the latter value (where the approximation (23) is used for n_i), the object indices i_1 and i_N are derived from (23). The index i_N is the largest value for which n_i'' yields the value N, while i_1 is the smallest value for which only one object copy is stored in the ring. From (23) it follows that

$$\begin{cases} \dfrac{\alpha}{N r_{i_1}} = \dfrac{1}{4N^2} + \dfrac{1}{3} \\ \dfrac{\alpha}{N r_{i_N}} = \dfrac{1}{4N^2} + \dfrac{1}{4N^2 - 1} \end{cases} \tag{28}$$

which, using the Zipf-like popularity distribution (25) yields immediately

$$\begin{cases} i_1 = \left\lfloor \left[\dfrac{NR}{\alpha A} \left(\dfrac{1}{4N^2} + \dfrac{1}{3} \right) \right]^{\frac{1}{\beta}} \right\rfloor \\ i_N = \left\lfloor \left[\dfrac{NR}{\alpha A} \left(\dfrac{1}{4N^2} + \dfrac{1}{4N^2 - 1} \right) \right]^{\frac{1}{\beta}} \right\rfloor \end{cases} \tag{29}$$

Using these values, s can now be calculated as

$$\begin{aligned} s &\approx \bar{s}\sum_{i=1}^{F} n_i'' \\ &= \bar{s}\left(N\min(i_N,F) + \sum_{\max[\min(i_N+1,F),1]}^{\min(i_1,F)} \frac{1}{2}\sqrt{1+\left[\frac{\alpha A}{NR} i^{\beta} - \frac{1}{4N^2}\right]^{-1}} + \max(F-i_1,0) \right) \end{aligned} \tag{30}$$

To simplify this expression, the middle term (30) is replaced by

$$\sum_{\max[\min(i_N,F),1]}^{\min(i_1,F)} \frac{1}{2}\sqrt{\frac{NR}{\alpha A} i^{-\beta}}$$

which is justified in view of the i-range values (between i_1 and i_N) of interest for this expression. If now the summations are approximated by integrals, we find the following expression for s :

$$\begin{aligned} \frac{s}{\bar{s}} &\approx N\min(i_N,F) + \max(F-i_1,0) \\ &+ \frac{1}{2}\sqrt{\frac{NR}{\alpha}}\sqrt{\frac{1-\beta}{F^{1-\beta}-1}} \frac{\min(i_1,F)^{1-\frac{\beta}{2}} - \max[\min(i_N,F),1]^{1-\frac{\beta}{2}}}{1-\frac{\beta}{2}} \end{aligned} \tag{31}$$

Of course, the case $i_N > F$ is of no practical use, since this would imply that all objects are stored on all locations, and that the ring network is actually not used. Therefore, for all practical situations, (31) becomes

$$\begin{aligned} \frac{s}{\bar{s}} &\approx Ni_N + \max(F-i_1,0) \\ &+ \frac{1}{2}\sqrt{\frac{NR}{\alpha}}\sqrt{\frac{1-\beta}{F^{1-\beta}-1}} \frac{\min(i_1,F)^{1-\frac{\beta}{2}} - \max(i_N,1)^{1-\frac{\beta}{2}}}{1-\frac{\beta}{2}} \end{aligned} \tag{32}$$

3.4.2.b Link capacity

The total link capacity needed in the ring network (l), in case the total cost is optimized, can be calculated from

$$l = \sum_{i=1}^{F} b_i r_i \overline{d_i(n_i)} \tag{33}$$

Assuming no correlation between object bitrate and popularity, and denoting the average object bitrate as $\bar{b}$, this becomes

$$l = \bar{b} \sum_{i=1}^{F} r_i \overline{d_i(n_i)} \approx \bar{b} \sum_{i=1}^{F} r_i \overline{d_i'(n_i)} \tag{34}$$

Taking into account the approximation (18) and the Zipf-like distribution (25), the total link capacity is given by

$$\frac{l}{\bar{b}} = \sum_{i=1}^{F} \frac{R}{A}^{i-\beta} \frac{N^2 - n_i^2}{4Nn_i} \tag{35}$$

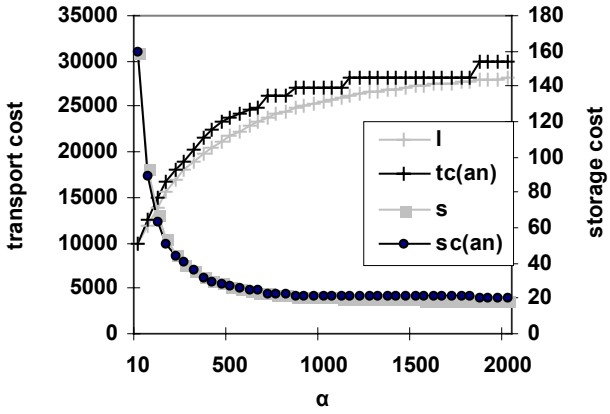


Figure 3.5: Transport (tc) and storage (sc) cost in a ring network with 8 surrogate servers, for both the exact and the approximated solution

Figure 3.5 compares the transport and storage cost for the exact and approximated analytical solution. The curves for the storage cost s and the transport cost l are given by the equations (31) and (35) respectively.

The difference between the approximated transport or storage costs and the exact solutions is always smaller than 10%. The approximated total cost (transport cost + $\alpha \cdot$ storage cost) remains within 5% of the exact total cost.

3.4.2.c Influence of request patterns

We used the analytical solution to study the influence of changes in the request pattern on the transport cost for a given network design. We assume that a ring network with 8 surrogate servers is optimally designed for distributing 20 files with a Zipf-like content popularity [2] (Figure 3.6) with parameter $\beta = 0.7$ (according to [2] and our own measurements on peer-to-peer file sharing applications [3]). In total 10000 requests are made for these files, evenly distributed over the 8 surrogate servers. The transport cost for the scenario with symmetrical user demand is given in Figure 3.7 (*tc (1:1),* (x:y) meaning that for every x requests at the first surrogate server, y requests are made at each other surrogate server).

We now look at the increase in transport cost that occurs when this optimally designed network is used for a different request pattern.

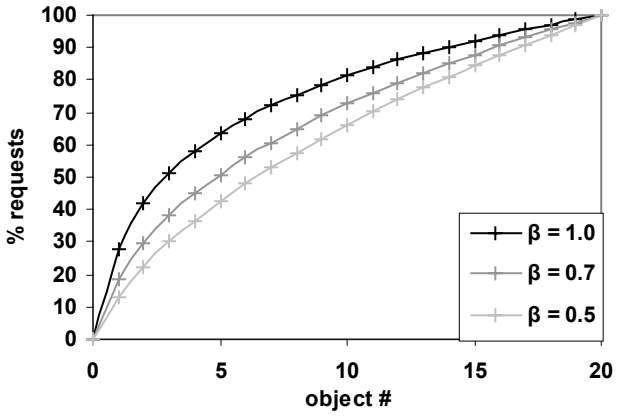


Figure 3.6: Cumulative Zipf-like distribution for the file popularity for different values of the Zipf parameter β.

The influence of the Zipf parameter β is rather limited. When the actual value of β is 0.5 (50% of all requests are made for the 7 most popular files) or 1.0 (50% requests for top 3) instead of 0.7 (50% requests for top 5), the transport cost is never higher than 2% above the optimal solution where the network would have been designed for the correct value of β (1% higher on average).

The influence of request asymmetry is more noticeable. When the storage locations are defined for a symmetrical user demand, the transport cost is given by *tc (1:1)* in Figure 3.7. When 3000 requests are made at one surrogate server and 1000 requests at each of the other surrogate servers (*tc (3:1)*), the transport costs would normally decrease according to the ILP solution for this asymmetrical design, but in the situation of a symmetrical design the transport cost will depend on the location of the surrogate server with 3000 requests, compared to the storage locations. The actual transport cost will then be somewhere between the best case (*tc (3:1) best,* the content is stored at the server with 3000 requests) and the worst case (*tc (3:1) worst*, 30% higher, the content is stored at servers far away from the server with 3000 requests). When all 10000 requests arrive at one surrogate server, the transport cost can be very high (*tc (1:0) worst*) if no optimal (ILP) design is used to take the traffic asymmetry into account.

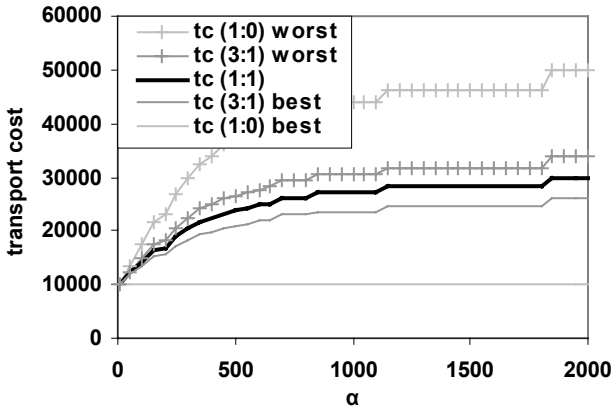


Figure 3.7: Transport cost (tc) in a ring network with 8 surrogate servers designed for a symmetric user demand, with an asymmetric user demand (X:Y means that for every X requests at the first surrogate server, Y requests are made at each other surrogate server)

3.5 Network design for ring based CDNs with a tree access topology

In a next step, we introduce surrogate servers in a tree-like access network, in addition to the ring CDN previously discussed. The analytical solution presented above is extended. We assume a tree topology consisting of L levels, each with a split x_l (the number of outgoing links for each node at level l, $l = 1 .. L$). The links in the access network are unidirectional. Similar to the previous section, we study the situation where the request pattern is symmetrical.

3.5.1 Analytical solution

The least popular objects will be stored on one or more of the N surrogate servers in the ring network (level 0), as described in the previous section. When the number of requests r_i is high enough, storage in the access network becomes beneficial. Due to the symmetry of the problem and the unidirectional access network links (no co-operation possible), an object should be stored at every surrogate server of the appropriate level.

Object o_i will be stored in the lowest level of the access network (level L, closest to the users) when the total cost (cache cost and transmission cost) at that level is lower then the total cost one level higher, or when

$$N\prod_{j=1}^{L} x_j \alpha s_i + r_i d_i(\mathbf{S_{L,i}}) b_i < N\prod_{j=1}^{L-1} x_j \alpha s_i + r_i d_i(\mathbf{S_{L-1,i}}) b_i, \qquad (36)$$

where $\mathbf{S_{l,i}}$ is the set of surrogate servers located at access network level l, for object o_i.

In general, an object o_i will be stored at level l when

$$N\prod_{j=1}^{l} x_j \alpha(s_{l+1} - 1) > r_i(d_i(\mathbf{S_{l-1,i}}) - d_i(\mathbf{S_{l,i}})) > N\prod_{j=1}^{l-1} x_j \alpha(s_l - 1) \qquad (37)$$

This means that the algorithm of Figure 3.3 has to be modified into the procedure of Figure 3.8.

1. For each $i = 1 \,..\, F$

   Calculate l out of (37)

2. If $l = 0$:

   Calculate $n_i = \max_{\bar{\Delta} C_i(n) > 0,\ 0 < n \le N} n$

   Else:

   $$n_i = N \prod_{j=1}^{l} x_j$$

3. Optimal surrogate server locations found for all $i = 1 \,..\, F$,

   calculate minimal cost from

   $$C = \sum_{i=1}^{F} (b_i r_i \overline{d_i(\mathbf{S_i})} + \alpha s_i n_i)$$

Figure 3.8: Exhaustive strategy to calculate optimal surrogate server location sets for replica placement for a set of objects $\boldsymbol{O}=\{o_1, \ldots, o_F\}$*, characterized by request rates* r_i*, on a ring based CDN with a tree access topology*

3.5.2 Experimental results

Using the strategy shown in Figure 3.8 strategy, the benefits of storage in the access network can be studied for topologies resembling the one presented in Figure 3.9. A central server is connected to the core ring network, where the edge surrogate servers are located. In the access network, multiple levels of aggregation are present, with hub surrogate servers at each level. The users are connected to level one hub surrogate servers, which are in turn grouped together by level two hub surrogate servers in a tree topology.

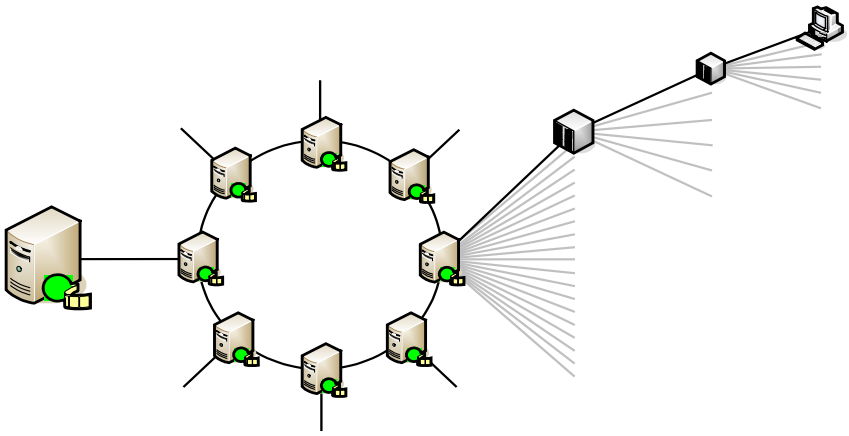
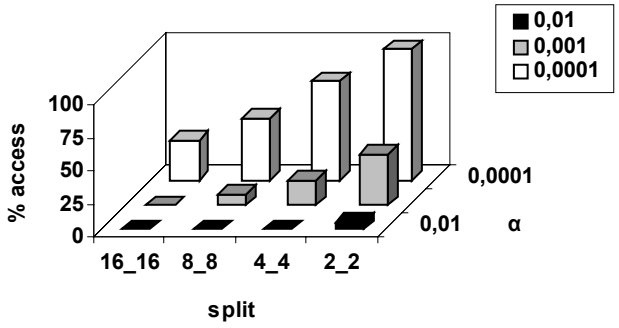


Figure 3.9: Ring network with tree access topology (2 levels)

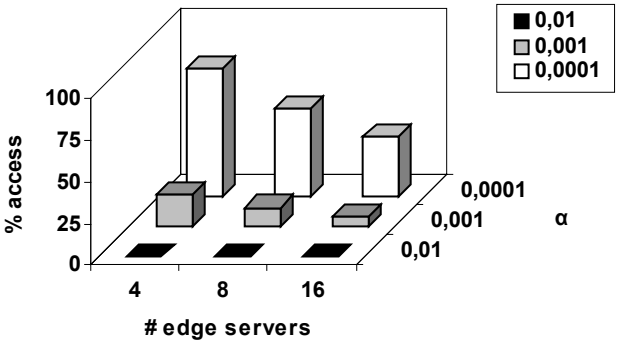
The influence of the parameter α, the split rate in the access network and the number of edge surrogate servers on the relative number of requests served by the access network servers is shown in Figure 3.10, for the analytical solution.

The total number of users and requests is kept constant: 100000 users and 2 requests per month per user, for a total number of 500 objects with Zipf-like popularity (β = 0.7). In Figure 3.10a the number of edge surrogate servers is constant (4), in Figure 3.10b the split rate is constant (4_4 or 4 outgoing links for level one and level two hub surrogate servers) and in Figure 3.10c α is constant (0.001). We notice that the efficiency of the hub surrogate servers increases for lower values of α and for denser user populations (lower split rates or less edge surrogate servers, when the number of users is kept constant).

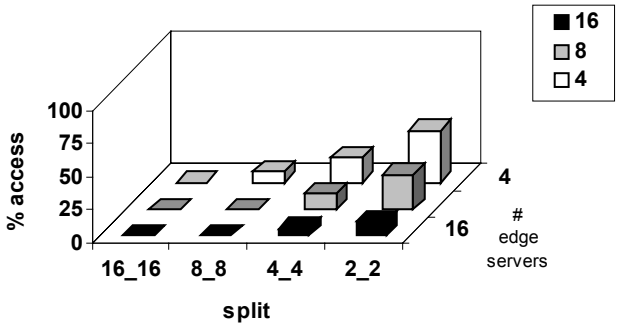
For α = 0.0001, a split rate of 2 per hub surrogate server and 4 edge surrogate servers (Figure 3.10a), all requests are served by the hub surrogate servers in the access network.



(a)



(b)



(c)

Figure 3.10: Influence of the split rate, α and the number of edge surrogate servers on the relative number of requests served by all hub surrogate servers in the access network

3.6 Dynamic heuristics for content replication

Contrary to the centralized and static solutions discussed in the previous sections, the distributed and dynamic algorithms presented in this section do not calculate global replica placements. Each surrogate server determines by itself, at run-time, which content is stored locally, depending on the traffic passing the node, and dynamically replaces stored content in case of changing request patterns.

Due to the decentralized nature of the algorithms, the results are slightly less optimal than for centralized solutions, but the CDN can now more easily adapt its replica placement to network failures or changes in user behaviour and provide a more robust content distribution service.

3.6.1 Heuristics

In this section we present two heuristics based on similar assumptions as for greedy algorithms (based on popularity and distance metrics), with a different point of view on storage costs (limited or unlimited cache sizes). Both algorithms can also introduce specific link costs, which can be used to provide load balancing on the network (see section 3.7).

3.6.1.a "Survival of the Fittest" heuristic

Every time an object passes one of the surrogate servers, this node will modify a parameter for that object. In this heuristic, this parameter $A_{n,o}$ for object o in node n only depends on the transport cost (amount of bandwidth used):

$$A_{n,o} = T_{n,o} \tag{38}$$

When an object o passes by node n, the transport cost $T_{n,o}$ is raised by the cost (number of bandwidth units on each link) to transport object o from the source node to node n (this cost would not be required if the object would have been stored in node n). We first store all the passing objects until the surrogate server is filled up (limited storage capacity) and then drop stored objects in favor of more popular or more distant objects (i.e. with higher values for $A_{n,o}$) passing by (*"Survival of the Fittest"*, *SF*). Note that this does not necessarily mean that every surrogate server stores the content that is locally most popular. $T_{n,o}$ also depends on the distance to the other nodes storing object o. Therefore it is possible that a very popular object o is not stored in a surrogate server, because another surrogate server nearby already stores a replica of it.

Figure 3.11 shows the normalized network and central server load for the core network part (with central server) of Figure 3.9. We assume that 500 objects are

available at the central server. When each of the surrogate servers in the core network can store 100 objects, the network load (occupied bandwidth) drops to less than 50% of its maximum value (when no caches are present). The central server load (number of simultaneous streams at the server) even decreases to 30%.

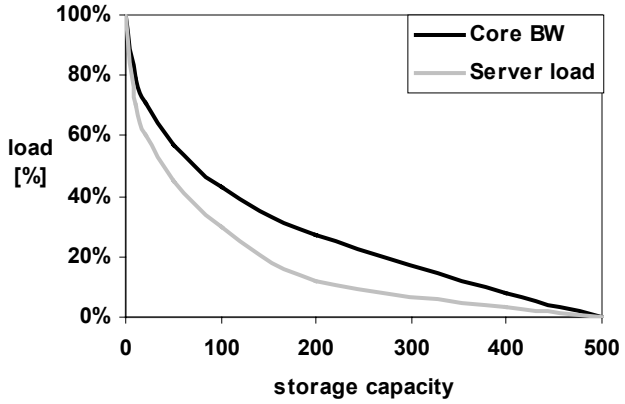


Figure 3.11: Network and central server load on a ring network with 8 surrogate servers

3.6.1.b "Storage Renting" heuristic

This algorithm is similar to the first, but now a storage cost is included in the calculation of the parameter $A_{n,o}$ (unlimited cache sizes). When this parameter is positive, the object will be cached (or stay cached), otherwise it will not be cached (or be dropped). The parameter $A_{n,f}$ for object o in node n is calculated as follows:

$$A_{n,o} = T_{n,o} - \alpha \cdot S_{n,o} \tag{39}$$

Besides the transport cost $T_{n,o}$ a storage cost $S_{n,o}$ is introduced. $S_{n,o}$ is raised by 1 every time unit object o is stored in node n (*"Storage Renting"*, *SR*). This way using a storage slot has a certain cost as well, so that this heuristic can also be used to determine the optimal size of the surrogate servers in the different parts of the network. $S_{n,o}$ is multiplied by the factor α, describing the relative cost between bandwidth and storage. If α is low, only the transport cost is considered to be important, as in the *SF* heuristic. If α is high the storage cost becomes more

important and the solution found will have only few stored replicas of the available content.

An example for this heuristic on the topology given in Figure 3.9 is shown below. We assume that the central server stores 500 objects (e.g. video streams) and that storage slots can be available on the core network as well as on the access network. First all content is only served by the central server, but after a while more objects are stored at the surrogate servers (Figure 3.12). The storage cost corresponds to the amount of used storage slots (or stored replicas) and is shown as the total cost per level (all level one hub surrogate servers, level two hub surrogate servers or edge surrogate servers).

For the given input parameters, introducing large storage facilities in the access network is not very beneficial: in steady state only 8 objects are stored in each level one hub surrogate server, 20 in each level two hub surrogate server and about 180 in each of the edge surrogate servers. When the access network servers would receive more hits (more popular content, a more dense access network, …) or when storage is cheaper (lower values for α), storage in the access network could have more advantages.

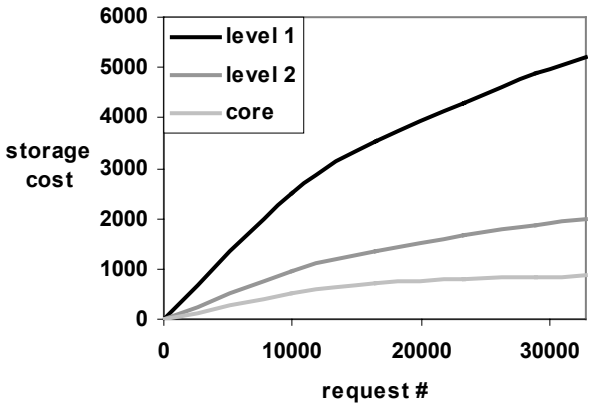


Figure 3.12: Storage cost in the core and access network (α=0.001, 500 objects, 32000 user requests, 100 level two hub surrogate servers, 600 level one hub surrogate servers, 100 users per level one hub surrogate server)

3.6.2 Comparison

In this section, the results of the *SR* heuristic are compared to the exact ILP solution for a ring network with 8 surrogate servers and 10000 requests in total ($\beta = 0.7$), as in Figure 3.4. Figure 3.13 shows the average extra network cost

(transport costs + α · the storage cost), caused by the distributed nature of the *SR* heuristic. The results are never worse than 6% above the ILP solution on average (8% for the worst case out of 10 simulations per value of α) for 20 available objects, 12% for 100 objects (15% worst case) and 18% for 500 objects (25% worst case). For very small or very large values of α, the *SR* heuristic eventually reaches the optimal solution (storing each object in every cache or in only one cache, respectively).

Note that the results for the ILP solution in Figure 3.13 for a certain value of α correspond to the results for the distributed *SR* heuristic for a value of α that is 10000 times smaller. This is because the centralized solution calculates the content placement for all 10000 requests at once, while the distributed solution adapts the content placement after each single request.

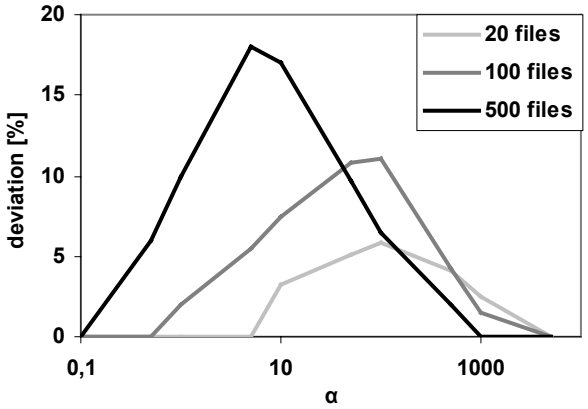


Figure 3.13: Deviation from the exact ILP solution (total network cost) for the SR heuristic on a ring network with 8 surrogate servers. 10000 requests are made for a variable number of available objects.

3.7 Dynamic heuristics for content replication with load balancing

3.7.1 Introduction

To illustrate the importance of load balancing, Figure 3.15 shows the bandwidth occupation (in number of simultaneous streams) on the different links of the core network given in Figure 3.14 (1 server and 4 surrogate servers). The *SF* heuristic is used and the surrogate servers can store 100 of all 500 available streams.

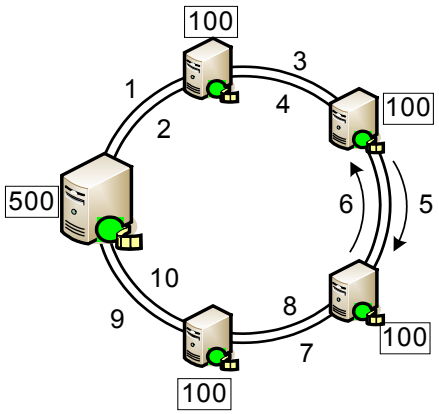


Figure 3.14: Core network topology, with uni-directional, numbered links

First the central server serves all requests, but in the steady state situation the surrogate servers are filled and serve many requests as well (see also Figure 3.4). The outgoing links of the central server (links 1 and 10) are heavily loaded, compared to the other links. Links 2, 4, 7 and 9 are not used at all.

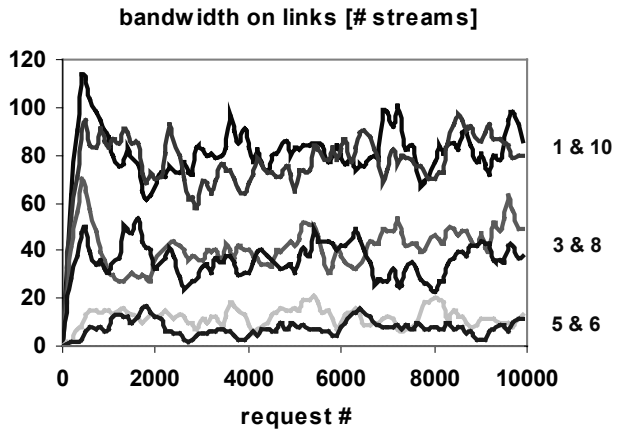


Figure 3.15: Bandwidth occupied on the core network links

When a more uniform load on each of the links could be achieved, a higher number of user requests could be supported by the network. Therefore, the goal of the following heuristics is to minimize the deviation of the actual link loads from the average link load. We assume that the link capacity is uniform on the network.

3.7.2 Heuristics

Both distributed RPAs can easily be adapted to support load balancing. The only changes in the algorithm are made in the calculation of $T_{n,o}$ as part of the parameter $A_{n,o}$. $T_{n,o}$ represents the streaming cost for object o in surrogate server n. When a object passes node n, $T_{n,o}$ is raised by the cost to transport the object from the source node to node n. Until now the cost c_e for using a link was set to 1 for each link. This means that the transport cost between two nodes is proportional to the number of hops between them. Now we change the cost c_e of a link e to

$$c_e = \left\lfloor \frac{1}{(1-l_e)^{\gamma}} \right\rfloor \tag{40}$$

with l_e the actual load on link e (relative to the link capacity). Some values for c_e are given in Table 3.1 (γ is set to 1). When the link load is at 95% of its maximum capacity, the cost for using this link for a new download is 20 times higher than the cost for using a free link. Note that when the load on the link is smaller than 50% of its capacity, no load balancing is done (c_e = 1). By introducing these link costs, the congested links will be avoided when calculating the shortest path (weighted Dijkstra algorithm) between the user and the candidate surrogate servers storing the requested object. Even when a congested link has to be used, the values for $T_{n,o}$ (and consequently $A_{n,o}$) will be higher for all nodes n after the congested link(s) on the path. Therefore more content will be stored beyond the congested link(s).

load	c_e
0	1
0,50	2
0,90	10
0,95	20
0,99	100
0,999	1000

Table 3.1: Link cost for a given load

The situation in Figure 3.15 now changes to that in Figure 3.16. The load on all links is now much closer to the average value (the variance is much lower) and links 4 and 7 also carry streams. Note that the average value of the total link load will be higher in the load balanced situation, compared to the original case, where the total bandwidth occupation on the network was minimised. Spreading the load over all the network links will therefore also slightly increase the average load.

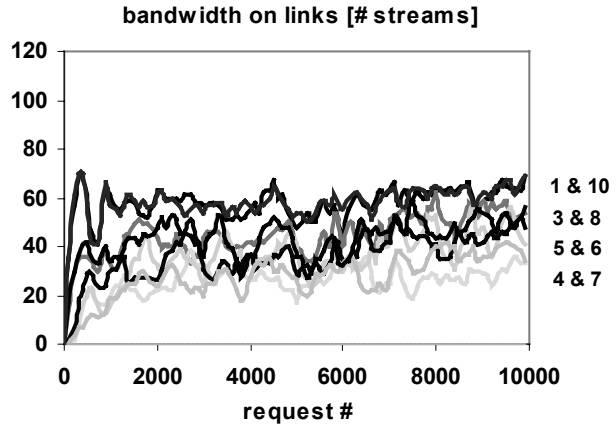


Figure 3.16: Bandwidth occupied on the load balanced core network links

3.7.3 Experimental results

To study the extended *SF* heuristics, simulations were performed on a network with a central server connected to a core ring with 8 surrogate servers (like on Figure 3.9). On average 450 streams are present on the network, 500 objects (with Zipf-like popularity distribution, β = 0.7) are available. The requests (10000 in total) are uniformly distributed over the different destination nodes and served over the least congested path.

The results for the *SF* heuristic are compared for different situations: with or without load balancing and for different cache sizes. The parameter γ is set to one. In Figure 3.17 the influence of load balancing on the average bandwidth usage on the core network links is shown. For intermediate cache sizes on the surrogate servers, the average bandwidth is up to 40% higher than in the optimal situation without load balancing. However, the deviation of the actual link load around this average is much lower, as shown in Figure 3.18.

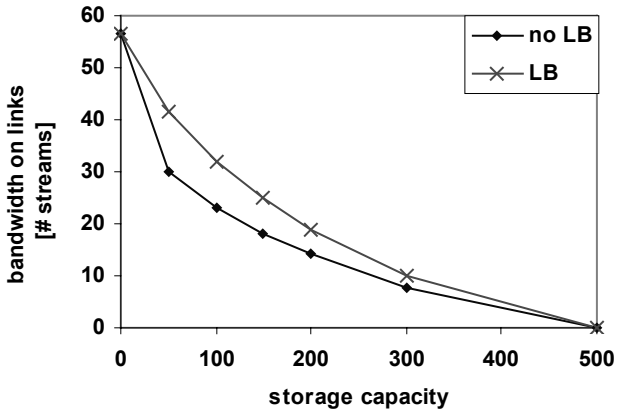


Figure 3.17: Average bandwidth usage on the core network links, with (LB) and without (no LB) load balancing (β = 0.7, γ = 1)

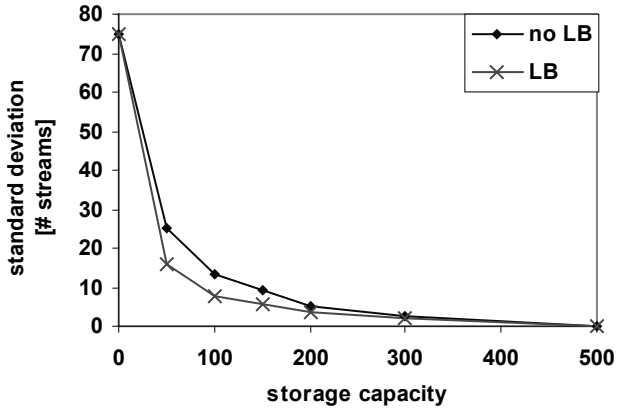


Figure 3.18: Standard deviation on the bandwidth on the core network links, with (LB) and without (no LB) load balancing

Balancing the load on the network comes at the price of a higher average link bandwidth. The influence of the parameter γ is not clearly visible on this network topology, since the content placement is already near the optimum. Simulations on more complex topologies like in section 3.8 show that larger values for γ increase the level of load balancing (higher average bandwidth and even lower values for the standard deviation).

3.8 Dynamic content replication in more complex topologies

This section presents simulations performed on a larger European core network (Figure 3.19). On average 450 simultaneous streams are present on the network, 200 objects (with Zipf-like popularity, $\beta = 0.7$) are available. The requests are randomly distributed over the different destination nodes and served over the least congested path. The results were compared for different situations, with or without load balancing ($\gamma = 1$): when only a central server is used, when a central server and 10-slot caches on all nodes are installed and when only 10-slot caches (P2P) are available.

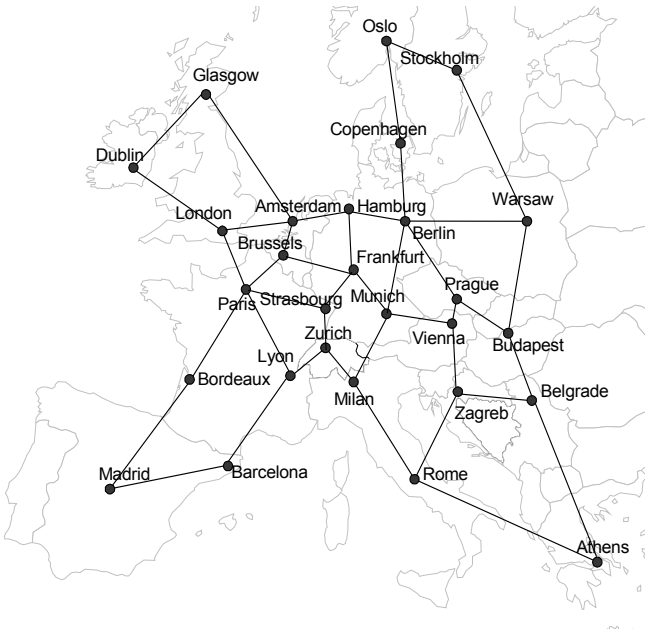


Figure 3.19: European network topology (28 nodes, 41 bi-directional links)

On Figure 3.20 the first situation, with one central server located in Vienna, is shown. The links close to the central server are heavily loaded (e.g. the outgoing links of Vienna, dark lines in Figure 3.20), while 50 out of all 82 uni-directional links carry no traffic at all (those not part of the shortest path tree).

A first solution towards a more symmetrical situation is to use caches in the network. When every node can store 10 objects, the central server will be

offloaded when these caches are filled up (*SF* heuristic, Figure 3.21). They typically store the 6 up to 8 most popular objects (since they are requested a lot locally) combined with a few less popular objects (so that at least one cache in the neighbourhood stores these objects, to avoid the long distance to the central server).

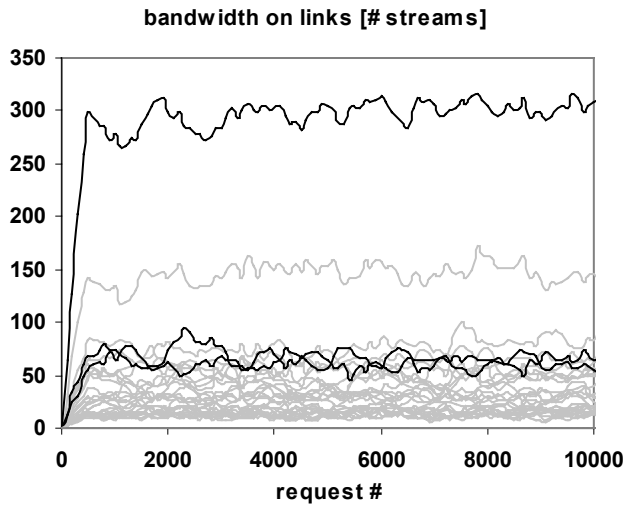


Figure 3.20: Bandwidth usage on a European network (central server only)

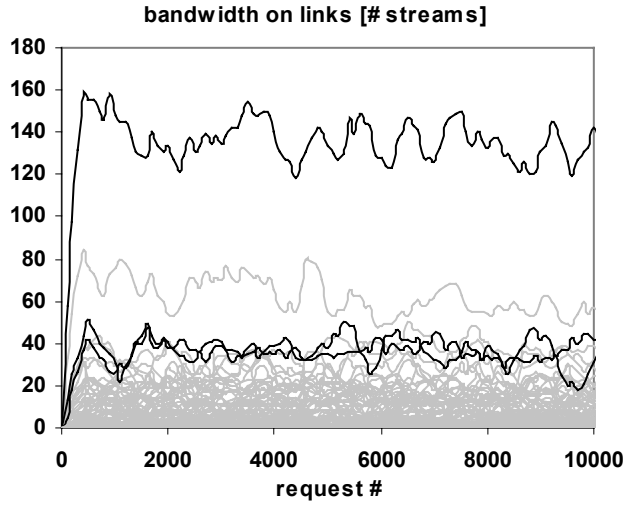


Figure 3.21: Bandwidth usage on a European network (central server and 10-slot caches)

We notice that the central server load has dropped from 450 to about 235 simultaneous streams (140+40+40 on the outgoing links and about 15 for local traffic): a gain of almost 50%. In this case, only 16 links are not in use. In Figure 3.22, the situation without a central server is shown. Only 10-slot caches are present at the network, so that 280 objects can be stored (the 200 original objects and 80 replicas).

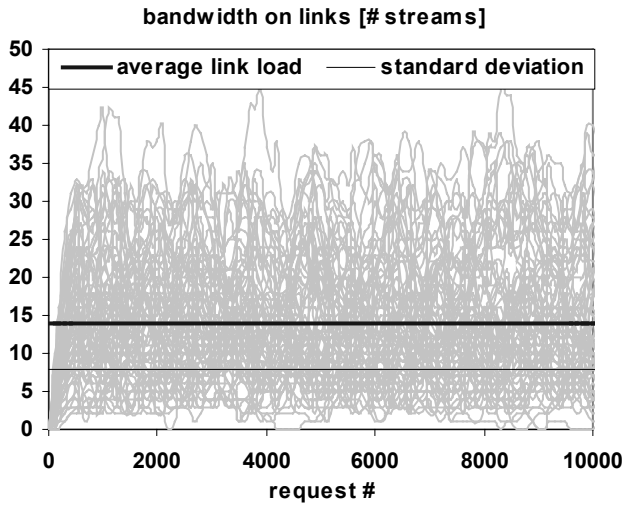


Figure 3.22: Bandwidth usage on a European network (10-slot caches only)

Now all links are in use: heavily loaded links at the center and less occupied links at the edge of the network. The average link load is 13.8 simultaneous streams, the standard deviation is 7.9 streams.

The following figures show the load balanced versions of Figure 3.20 and Figure 3.22. In Figure 3.23 every outgoing link of the central server has the same load.

The load on the other network links remains almost the same, as no load balancing is done on most of these links (see Table 3.1, the capacity of all links has been "set" to 150 simultaneous streams). If the load on these network links should also be balanced, the capacity of these links has to be set to lower values.

The effect of the load balancing algorithm is also clearly visible on Figure 3.24, for the situation with 10-slot caches only (the capacity of all links can be "set" to a value as low as 30 simultaneous streams, before streams get lost). The standard deviation for the link load has decreased from 7.9 to 4.6 streams. The average

link load however is now slightly higher: 14.7 simultaneous streams, instead of 13.8 in Figure 3.22.

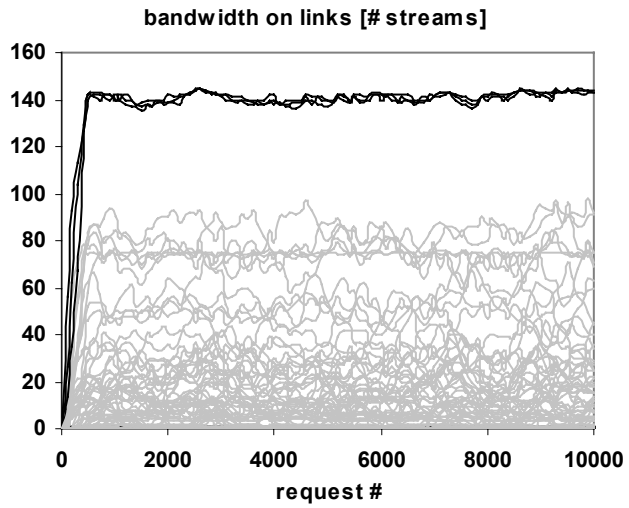


Figure 3.23: Link load on a load balanced European network (central server only)

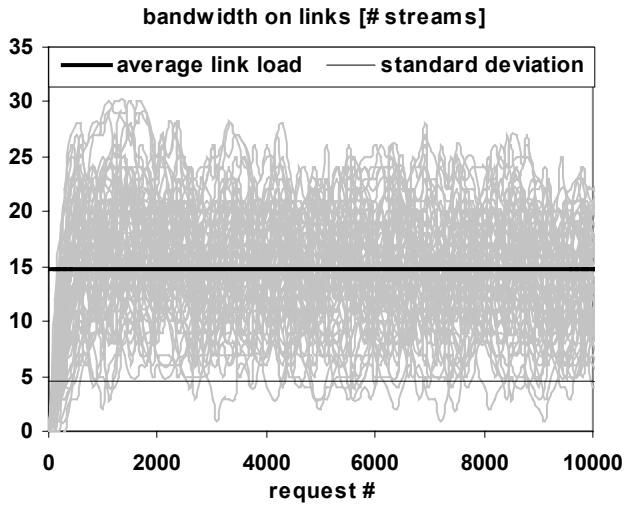


Figure 3.24: Link load on a load balanced European network (10-slot caches only)

3.9 Comparison of dynamic heuristics for content replication

In this paragraph the *SF* heuristic, with and without load balancing, will be compared to several standard RPAs, described in detail in [7]. The average link load, the standard deviation for the link load and the average distance between source and destination node will be studied, for the situation without a central server (P2P caches only). In every node an equal amount of cache slots is available. Initially the available cache slots are filled up randomly with original content and replicas. Since the standard heuristics are static (an initial replica placement is calculated for a fixed user request pattern), the results used for our *SF* algorithm are in the steady state situation. A short description of the RPAs is given below. Each object is at least stored once in the network and all caches are filled up according to these heuristics.

Random. The content is uniformly replicated over the different caches.

Popularity Local. Each cache stores the content that is most popular to its local users.

Greedy Single. Each cache calculates the cost for each object, when the content placement would be random. The global object popularity and distance to the closest cache storing the object are taken into account. It then stores as many objects, with the highest costs, as possible.

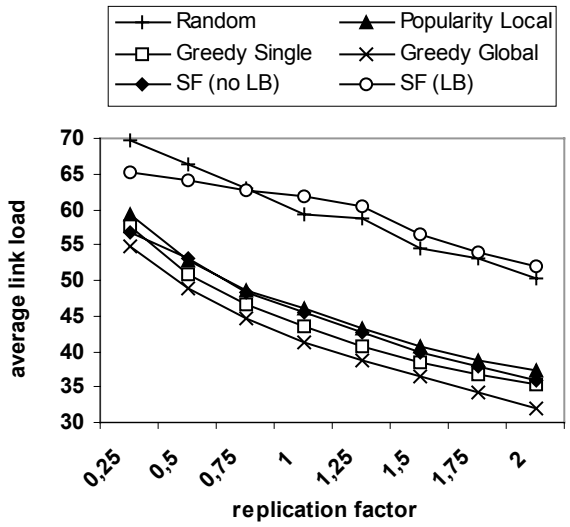
Greedy Global. The cost for each object, when the content placement would be random, is calculated for each of the caches. The global object popularity and distance to the closest cache storing the object are taken into account. The object-cache-pair with the highest cost is determined and that object is stored in that cache. The calculations are then iterated for the new content placement, until all caches have been filled.

Figure 3.25 shows the average link load (in number of simultaneous streams), the standard deviation for the link load and the average distance between source and destination cache on the network topology shown in Figure 3.19, in steady state. 2800 objects are made available (β = 0.7). The replication factor indicates the number of replicas, divided by the number of original objects (e.g. factor 1 means that the number of replicas equals the number of original objects in the network).

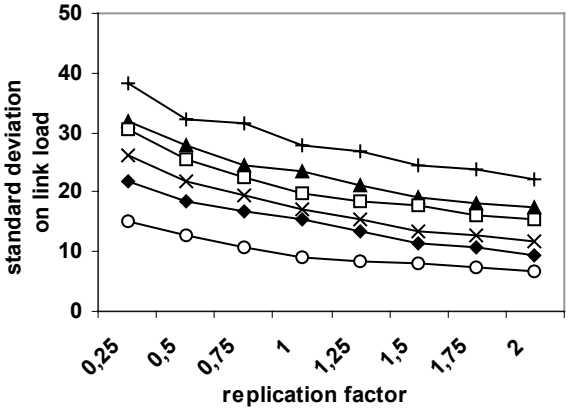
As could be expected, the average load and distance decrease when the caches become bigger. The *greedy global* heuristic provides better results, but is very computationally heavy. The steady state results for the *SF* heuristic are close to

the results for the *popularity local* and *greedy single* heuristics, which have similar approaches. The standard deviation however is a bit lower.

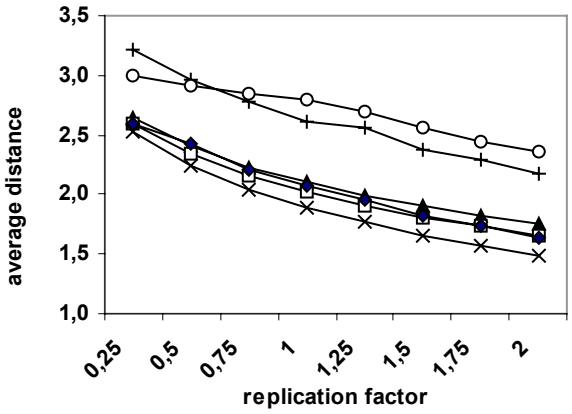
Note that the standard RPAs have to recalculate the entire content placement over all nodes, in case new objects are added to the network or when the user behaviour changes. Our *SF* heuristic dynamically and gradually adapts local content placement to these events.



(a)



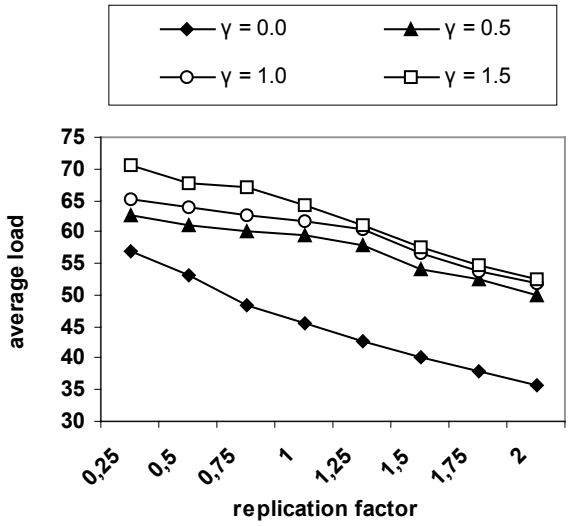
(b)



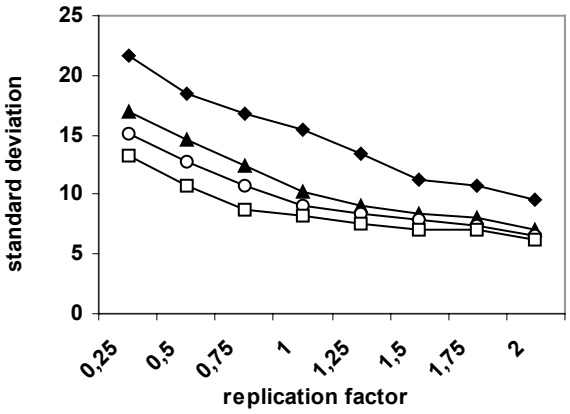
(c)

Figure 3.25: Comparison of the average link load, the standard deviation on the link load and the average hopcount between our RPA and standard algorithms

When we look at the load balanced version of the *SF* heuristic (LB, γ=1), the standard deviation has decreased considerably, at the price of a higher average load and longer distances to the source cache.



(a)



(b)

Figure 3.26: Influence of γ on load balancing

Figure 3.26 shows the results for load balancing with different values for the parameter γ in equation (3). When γ=0, no load balancing is done (c_e=1 for all links). Higher values for γ decrease the standard deviation for the link load, at the expense of an increased average load. The influence of γ is more significant for smaller values of the replication factor.

3.10 Conclusion

In the first part of this chapter we have presented a static solution for the replica placement problem for streaming media services in CDNs. We have compared an ILP formulation of the problem to the analytical solution for ring based CDNs. Both approaches showed that the load on the network and the central server can be considerably decreased. Adding storage facilities in the access network or even at the home network can be cost effective and might be interesting to study in future work.

Afterwards two distributed algorithms have been introduced. These heuristics dynamically adapt the replica placement to variations in the network load, user behaviour or available content. This way congestion in the network can be avoided and a more robust service can be provided, at the price of a slightly increased network load.

The introduction of link costs in the load balancing algorithms can also be used for different objectives. Instead of specifying the link load, they could also

represent transfer or propagation delays. Minimizing link delays together with other network resources can be interesting for future work. While the approach of storing whole objects, as presented in this chapter, is very effective for Video on Demand services, a method of storing partial objects (e.g. with sliding intervals) can be interesting for very popular content (e.g. live television), as studied in 5 on time-shifted television.

References

[1] Akamai, http://www.akamai.com

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", IEEE Infocom, 1999.

[3] P. Backx, T. Wauters, B. Dhoedt, P. Demeester, "A comparison of peer-to-peer architectures", Eurescom Summit, 2002.

[4] L. Qiu, V. N. Padmanabhan, G. M. Voelker, "On the Placement of Web Server Replicas", IEEE Infocom, April 2001.

[5] M. Karlsson, M. Mahalingam, "Do We Need Replica Placement Algorithms in Content Delivery Networks?", Seventh International Web Content Caching and Distribution Workshop, August 2002.

[6] M. Karlsson, C. Karamanolis, M. Mahalingam, "A Framework for Evaluating Replica Placement Algorithms", Technical Report HPL-2002, HP Laboratories, July 2002.

[7] J. Kangasharju, J. Roberts, K. Ross, "Object replication strategies in content distribution networks", Computer Communications 25 (4) (2002) 376-383.

[8] D. Turrini, F. Panzieri, "Using P2P Techniques for Content Distribution Internetworking: A Research Proposal", Second International Conference on Peer-to-Peer Computing, 2002.

[9] T. Wauters, J. Coppens, T. Lambrecht, B. Dhoedt, P. Demeester, "Distributed Replica Algorithms for Peer-to-Peer Content Distribution Networks", EuroMicro Conference, September 2003.

[10] J. M. Almeida, D. L. Eager, M. K. Vernon, "A Hybrid Caching Strategy for Streaming Media Files", Proceedings of Multimedia Compuing and Networking (MMCN), San Jose, CA, January 2001.

[11] X. Tang, J. Xu, "On Replica Placement for QoS-aware Content Distribution", IEEE Infocom - The Conference on Computer Communications, March 2004.

[12] M. Yang, Z. Fei, "A Model for Replica Placement in Content Distribution Networks for Multimedia Applications", ICC 2003 - IEEE International Conference on Communications, May 2003.

[13] M. Karlsson, M. Mahalingam, "Choosing Replica Placement Heuristics for Wide-Area Systems", Proceedings of the International Conference on Distributed Computing Systems (ICDCS), March 2004.

[14] I. Cidon, S. Kutten, R. Soffer, "Optimal Allocation of Electronic Content", Proceedings of IEEE Infocom, April 2001.

[15] J. Coppens, T. Wauters, F. De Turck, B. Dhoedt, P. Demeester, "Evaluation of a Monitoring based Architecture for Delivery of High Quality Multimedia Content", Conference proceedings of 10th IEEE Symposium on Computers and Communications ISCC 2005, June 27-30, 2005, Cartagena, Spain.

[16] J. Coppens, T. Wauters, F. De Turck, B. Dhoedt, P. Demeester, "Evaluation of Replica Placement and Retrieval Algorithms in Self-Organizing CDNs", Conference proceedings of IFIP/IEEE International Workshop on Self-Managed Systems & Services SelfMan 2005, May 19, 2005, Nice, France.

[17] T. Wauters, J. Coppens, B. Dhoedt, P. Demeester, "Load balancing through efficient distributed content placement", Conference proceedings of NGI 2005, April 18-20, 2005, Rome, Italy.

[18] J. Liu, J. Xu, "Proxy caching techniques for media streaming over the Internet", IEEE Communications Magazine, vol. 42, no. 8, August 2004, pp. 88-94.

[19] Cplex, http://www.ilog.com/products/cplex/

4

Optical metro and HFC access network design for video on demand

4.1 Introduction

In the previous chapter, the network design and content placement for a video on demand service was presented for a CDN core network with a tree-like access network structure, as often used in DSL based deployments. This chapter presents the VoD design for an optical metro network and HFC based access network. As in this study, the goal was to avoid any traffic on the core network, an architecture with independent distributed servers is proposed. The network structure is shown in Figure 4.1. At each head end in the metro network, multiple optical nodes, where the optical signals are converted into electrical signals, are present. The end users are connected to these nodes through coax cable. As the video streams are transported over an optical metro network, Gigabit Ethernet (GbE) over Wavelength Division Multiplexing (WDM) technologies are introduced, which offer high bandwidth streaming opportunities for VoD services. When designing the network that supports these services, it is important to decide where to place the video servers, the WDM equipment, and the GbE switches on the metro ring network. The installation cost for network elements

on the HFC access network, such as quadrature amplitude modulation (QAM) devices, also has to be taken into account. These devices support the additional radio frequency (RF) channels needed to transport the video streams from the head ends to the optical nodes.

As mentioned previously, one of the most important VoD services is interactive VoD (iVoD), also called real VoD. Customers can select any available movie or program at any time on their TV screen and pause, fast forward or rewind as they please. This approach is different from near VoD (nVoD), where movies only start at specific times and no interaction from the customer can be supported. Where nVoD can be broadcast to the users, the more user-friendly iVoD service requires bandwidth-intensive unicast streaming.

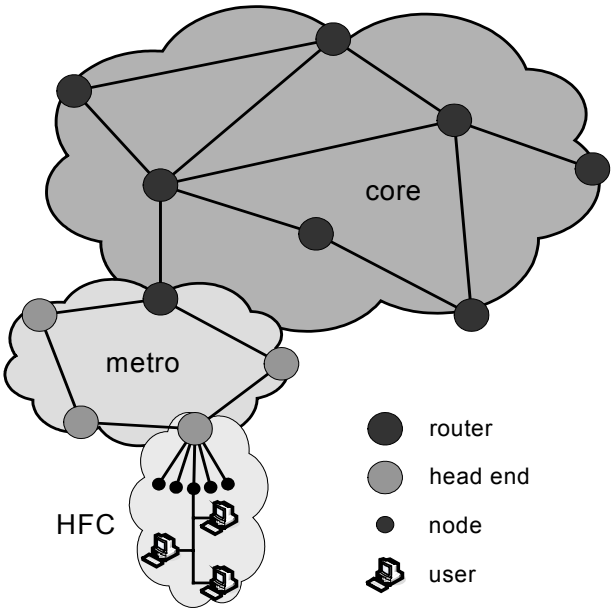


Figure 4.1: General network structure (the network is divided into a core network with local metro networks and HFC access networks)

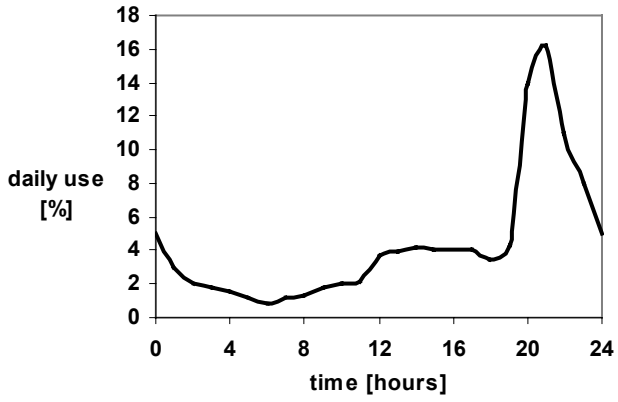
Our network design model presented in this chapter determines the optimal number and location of video servers on the head ends in the metro access network. The installation of additional switches and WDM network equipment in these head ends will be investigated as well. In discovering the most optimal design, issues like viewing behavior, grooming strategies, statistical multiplexing and Erlang modeling are brought into play.

The rest of this chapter is organised as follows. First some issues on the traffic model are discussed in Section 4.2. Section 4.3 presents an Integer Linear Programming (ILP) model for the design of the metro network for iVoD services only. A linear network model is built and the necessary GbE and WDM restrictions are included. Finally a network design tool, based on heuristics derived from the ILP model, is introduced in Section 4.4. Simulations with this tool allow us to study different VoD services and the influence of network and user parameters.

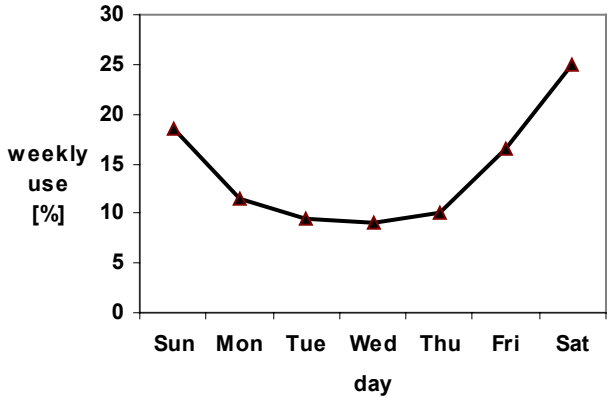
4.2 Traffic model

Daily user behavior for video on demand is shown in Figure 4.2a, characterised by peak values between 8 and 9 PM. The weekly VoD behavior [2] shows that Saturday is the most popular day (Figure 4.2b). Combining both figures learns that about 5% of all weekly download requests are made during peak hour on Saturday night [2]. Currently about 1.5 to 3 movies are watched per month, per subscriber. In our model, we assume that 5% of all VoD subscribers watch a video simultaneously during peak hour. The network should therefore be designed to cope with this peak traffic.

In a typical HFC access network architecture (Figure 4.1), about thousand users ("homes passed", HP) are grouped together at an optical node (on a coax trunk), while several tens of nodes are combined at one head end (on a fiber network). To determine the exact number of extra RF channels and corresponding QAM devices needed at the HE, a traffic model similar to the Erlang model for telephony has been studied.



(a)



(b)

Figure 4.2: Viewing behavior for video on demand: daily (a) and weekly (b) (peak traffic occurs on Saturdays, between 8 and 9 PM)

4.2.1 Erlang model

In the Erlang model for telephony, the traffic intensity is defined as the average number of calls simultaneously in progress during a particular period of time. It is measured in units of Erlang. The assumptions for the traffic model for telephony are also valid for VoD services:

- Poisson arrivals: the arrivals of user requests are independent.
- statistical equilibrium: statistics do not change during peak hour.

The traffic intensity per optical node can easily be calculated. Of all 1000 HP at one node, only a fraction will be VoD subscribers. When eventually one third of all users become VoD subscribers and 5% of them watches a movie at peak hours simultaneously, about 17 simultaneous video sessions will be present on average. A linear approximation of the Erlang lost-call formula, determining the number of required video slots N, when 99% of all requests have to be served successfully, is

$$N = 6 + A/0.85\,,\ A < 75 \qquad (1)$$

where A is the traffic intensity in Erlang. This means that according to Erlang's model 26 video channels have to be available, for an average number of 17 simultaneous requests.

Depending on the QAM modulation techniques used, about 38 (64-QAM) to 51 Mbps (256-QAM) is available per RF channel. This way 10 to 13 MPEG-2 streams at 3.8 Mbps can be carried. In case of 64-QAM modulation, Erlang's model asks for 3 RF channels per node. Therefore, QAM devices with on average 150 RF channels should be installed additionally on every head end (with 50 optical nodes) for iVoD services. We assume that QAM devices with one GbE input ports and a fixed number of RF output ports are available.

On the metro network, traffic from different nodes can be aggregated at the head end, so that a statistical multiplexing gain can be achieved [3]. While the necessary capacity at the node level is more than 50% higher than the average value (26 video channels needed for an average value of 17 simultaneous streams), the aggregated capacity is only 8% higher at the head end level (903 channels needed for an average of 833 requests, for 50 nodes). The reason for this is that, for large values (where the Erlang model can be approximated by the normal distribution), the variance follows a square-root dependency, while the average traffic volume grows linearly with the number of nodes [4].

4.2.2 Traffic grooming

Another key issue in network design is to groom the traffic in such a way that a good compromise between capacity efficiency and node cost can be achieved. Two extreme strategies exist: end-to-end (E2E) and link-by-link (LbL) grooming. In E2E grooming, a dedicated logical link is used for each traffic demand, while in LbL grooming, each network node terminates all logical links entering that node.

In the Erlang model, we have indicated the variable nature of the aggregated video traffic. Therefore, we will use E2E grooming only for all completely filled GbE links. The GbE links that are not always completely filled, due to the traffic variability, are combined and sent LbL (i.e., through the switches). This strategy is called hybrid grooming [4].

An example is given in Figure 4.3. The demands from the left nodes (e.g., the head ends) to the right node (e.g., the server) require on average 2.5 circuits (e.g., GbE signals), but only a capacity of $2.5C$-A is (almost) always needed, while up to $2.5C$+A is sporadically required. We assume that in 99% of all cases, the actual demand is in the interval [$2.5C$-A, $2.5C$+A].

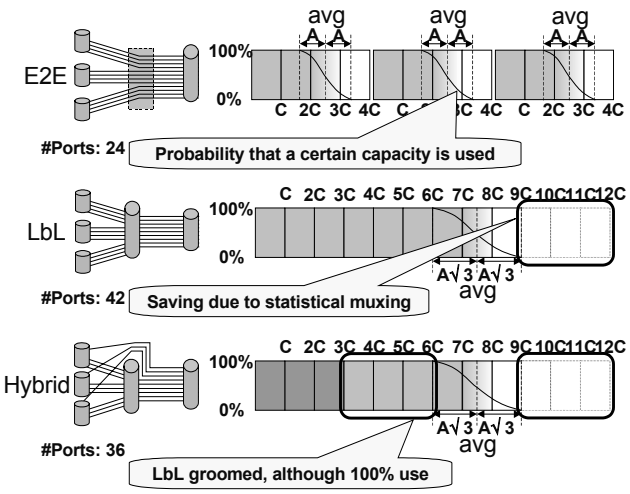


Figure 4.3: Different grooming strategies (the hybrid strategy combines the benefits of end-to-end and link-by-link grooming)

In case of E2E traffic, $3 \cdot 4 = 12$ circuits are cross-connected in the middle node. In the LbL grooming case, only $7.5C+\sqrt{3}A = 9$ circuits are needed (if $A \leq 1.5/\sqrt{3}$), but 18 extra GbE ports are required in the middle node. In the hybrid strategy, the completely filled circuits are cross-connected, while the partially filled circuits are sent link-by-link. This way six GbE ports are saved in the middle node. Note that cross-connecting a second circuit per demand would probably also make sense, since these circuits are also nearly completely filled. This requires an additional circuit between the middle and the right node, but it saves six GbE ports in the middle node. The optimal case will then depend on the costs of GbE ports in the middle node (e.g., switch ports) compared to GbE ports in the right node (e.g., server ports). A more detailed study on traffic grooming is presented in Appendix C.

4.3 ILP model

To determine the optimal placement of video servers, switches, and WDM equipment on the local nodes of each metro network in Figure 4.1, an ILP model for this dimensioning problem has been formulated. This model will describe the iVoD traffic on the metro network only.

First some assumptions about the different network technologies are given. Afterwards the objective function and the network restrictions for the actual ILP

formulation are presented. Finally, a standard network configuration will be simulated using CPLEX [8] and examined as a use case.

4.3.1 Network model

4.3.1.a Network layers

The modeling of the Ethernet over WDM technology is done in a multi-layer structure. Every network link consists of a number of fibers, each with a fixed number of wavelengths on it. On these wavelengths, GbE signals carrying the video streams can be transported. We assumed a CWDM technology, for the benefits of reduced hardware costs and low power dissipation on these short-haul metro networks. In our simulations eight wavelengths per fiber are supported. Each of these wavelengths can carry two GbE signals. The top layer is responsible for the transport of the individual streams (e.g. 300 MPEG-2 streams per GbE signal).

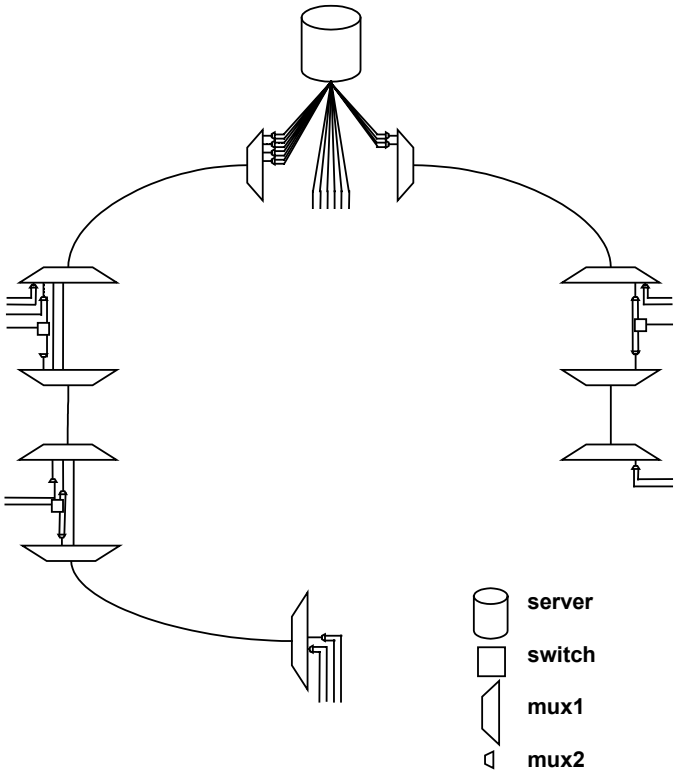


Figure 4.4: Metro network configuration with network elements in the GbE and WDM layer

Each of the layers has its own equipment, with different equations describing them. We assume that the servers and the switches have GbE ports. Different CWDM network elements are used for multiplexing: mux1 combines the wavelengths on one fiber (WDM layer), mux2 combines the GbE signals in one wavelength (GbE layer). Figure 4.4 shows these network elements on a possible metro network configuration.

4.3.1.b GbE layer

Before the ILP formulation is given, extra server and client nodes are added to the network and the links are split up into GbE layer signals (Figure 4.5). A server node represents a location where a server could possibly be placed. A client node represents all end users in the access network at the head end. Each of the links at the bottom side in Figure 4.5 can now transport one GbE signal. GbE signals coming from the server are either sent to one of the switch ports (one of the n_{max} "switch nodes") or not (one of the n_{max} "non-switch nodes") and further on to the client nodes.

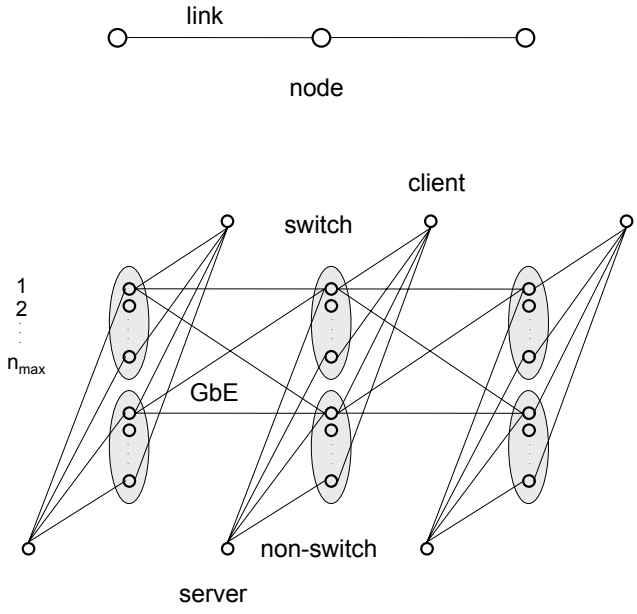


Figure 4.5: The network links (top) are split up into GbE level links (bottom). Each network node (top) is split up into one server node, one client node, n_{max} switch nodes and n_{max} non-switch nodes accordingly (bottom)

The value of n_{max} depends on the number of fibers, wavelengths and GbE signals per link. In our simulations (one fiber with eight wavelengths, each with two GbE signals) n_{max} is 16. Video signals can now be streamed over these GbE links. The maximum number of streams per GbE link is given (e.g. 300).

4.3.2 ILP formulation

The objective function and the restrictions for the ILP formulation of the problem are given in this section. First the necessary symbols are introduced.

4.3.2.a Symbols

Each network node $n \in N$ is now split up in one server node $s_n \in S$, one destination node $d_n \in D$, switch nodes $x^1_n \in X^1_n$ and non-switch nodes $x^0_n \in X^0_n$.

Input parameters	
v_d	number of streams for destination d
i_s	installation cost for server s
c_s	cost for a server port
c_{m1}	cost for a mux1
c_{m2}	cost for a mux2
c_x	cost for a switch port
Solution	
p_s	number of ports used at server s
m_{1n}	number of mux1 used at node n
m_{2n}	number of mux2 used at node n
g_n	number of switch ports used at node n

Table 4.1: Symbols for the ILP formulation

Each GbE link $e \in E$ carries v_e simultaneous video streams, of which $v_{e,d}$ are for destination node d. If v_e is larger than zero, the link e is in use: $b_e = 1$ (else $b_e = 0$).

If a server s_n has to be placed in node n, $b_s = 1$ (else $b_s = 0$). The input parameters and the variables describing the final solution are explained in Table 1.

I_n contains all incoming links of node n, O_n all outgoing links.

4.3.2.b Objective function

The objective function that has to be minimised represents the installation cost of all network elements. It is given by

$$\min\left(\sum_{s\in S}(i_s b_s + c_s p_s) + \sum_{n\in N}(c_n g_n + c_{m1} m_{1n} + c_{m2} m_{2n})\right) \quad (2)$$

Equation (2) consists of two parts. The first part determines the installation costs and the costs for the GbE ports at the servers. The second part gives the costs of the WDM equipment and the GbE ports at the switches.

4.3.2.c Restrictions

While minimising Equation (2), several restrictions have to be taken into account. These constraints, explained below, describe the traffic flow, the GbE and WDM technology limitations and network equipment.

Capacity restrictions. The maximum number of streams per GbE link and the maximum number of server ports are given by (e.g., $v_{max} = 300$):

$$\sum_{d\in D} v_{e,d} \le v_{\max} b_e\,,\ \forall e \in E \quad (3)$$

$$\sum_{e\in O_s} b_e \le s_{\max} b_s\,,\ \forall s \in S \quad (4)$$

The binary variables b_e and b_s are now automatically forced to 1 if traffic is present on link e or out of server s respectively.

In/out restrictions. In/out restrictions make sure that the streams reach their destination through the network. Equation (5) takes care of server nodes, Equations (6) to (8) describe the behavior of switch nodes, Equations (9) to (11) are used for non-switch nodes and Equations (12) and (13) for destination nodes.

$$\sum_{s\in S}\sum_{e\in O_s} v_{e,d} = v_d\,,\ \forall d \in D \quad (5)$$

$$\sum_{x\in X_n^1}\sum_{e\in I_x} v_{e,d} = \sum_{x\in X_n^1}\sum_{e\in O_x} v_{e,d}\,,\ \forall d \in D\,, \forall n \in N \quad (6)$$

$$\sum_{e\in I_x}\sum_{d\in D} v_{e,d} \le v_{\max},\ \forall x \in X_n^1, \forall n \in N \tag{7}$$

$$\sum_{e\in O_x}\sum_{d\in D} v_{e,d} \le v_{\max},\ \forall x \in X_n^1, \forall n \in N \tag{8}$$

$$\sum_{e\in I_x} v_{e,d} = \sum_{e\in O_x} v_{e,d},\ \forall d \in D, \forall x \in X_n^0, \forall n \in N \tag{9}$$

$$\sum_{e\in I_x} b_e \le 1,\ \forall x \in X_n^0, \forall n \in N \tag{10}$$

$$\sum_{e\in O_x} b_e = \sum_{e\in I_x} b_e,\ \forall x \in X_n^0, \forall n \in N \tag{11}$$

$$\sum_{e\in I_d} v_{e,d} = v_d,\ \forall d \in D \tag{12}$$

$$\sum_{e\in I_d} v_{e,d'} = 0,\ \forall d \in D, \forall d' \ne d \tag{13}$$

Equation (5) states that every destination node has to be served from (any of) the potential servers. In Equation (6) it is made sure that all incoming traffic in the switch nodes has to leave on the outgoing links of one of the switch nodes. Per switch node, the total amount of in- and outgoing video streams has to be limited by v_{max} through Equations (7) and (8). Similar restrictions can be found at the non-switch nodes, but the incoming traffic on a non-switch node has to leave on the outgoing links of the same non-switch node, because of Equations (9), (10) and (11). Equation (12) ensures that every destination node receives the requested streams on its incoming links (and no other traffic: Equation (13)).

Equipment restrictions. The equations for the WDM equipment are described below. The parameters n_1 and n_2 indicate the maximum number of wavelengths per fiber (e.g. 8) and the maximum number of GbE signals per wavelength (e.g. 2) respectively.

$$\sum_{x\in X_n^1}\sum_{e\in L_x} b_e + \sum_{x\in X_n^0}\sum_{e\in L'_x} b_e \le n_2 m_{2n}^l,\ \forall n \in N \tag{14}$$

$$\sum_{x\in X_n^1}\sum_{e\in R_x} b_e + \sum_{x\in X_n^0}\sum_{e\in R'_x} b_e \le n_2 m_{2n}^r,\ \forall n \in N \tag{15}$$

$$\sum_{x\in X_n}\sum_{e\in L_x} b_e \le n_1 n_2 m_{1n}^l,\ \forall n \in N \tag{16}$$

$$\sum_{x \in X_n} \sum_{e \in R_x} b_e \leq n_1 n_2 m_{1n}^r, \ \forall n \in N \quad (17)$$

Equations (14) and (15) determine the number m_{2n} of GbE layer multiplexers (mux2) at both sides of node n (undicated by the upper index l, for left, and r, for right). The total number of GbE signals that have to be (de-)multiplexed in this layer can be found by counting all signals going through the switch nodes (L_x at the left side, R_x at the right side) and all signals passing through non-switch nodes, coming from the server or going to the destination node (L'_x at the left side, R'_x at the right side). GbE signals that are just passing through the node (end-to-end signals) remain in the WDM layer. Since all signals have to be counted for WDM layer multiplexing, Equations (16) and (17), determining the number m_{1n} of mux1 at both sides of node n, are more straightforward.

Type restrictions. The types of the different variables are described in Equations (18).

$v_{e,d}$ integer, $\forall e \in E, \ \forall d \in D$ (18)

b_e, b_s binary, $\forall e \in e, \ \forall s \in S$ (18')

$m_{1n}^l, m_{1n}^r, m_{2n}^l, m_{2n}^r$ integer, $\forall n \in V$ (18'')

4.3.2.d Solution

The number of server ports, switch ports, and WDM elements for each of the network nodes are given by the following equations:

$$\sum_{e \in O_s} b_e = p_s, \ \forall s \in s \quad (19)$$

$$\sum_{x \in X_n^1} \left(\sum_{e \in L_x} b_e + \sum_{e \in R_x} b_e \right) = g_n, \ \forall n \in N \quad (20)$$

$$m_{2n}^l + m_{2n}^r = m_{2n}, \ \forall n \in N \quad (21)$$

$$m_{1n}^l + m_{1n}^r = m_{1n}, \ \forall n \in N \quad (22)$$

Equation (19) determines the number of server ports by counting the occupied outgoing GbE links of each server. Equation (20) gives the switch port count. The number of multiplexers for the WDM and GbE layers is shown in Equations (21) and (22).

4.3.3 Case study

Simulations based on this ILP model were performed on a metro ring network with six head ends and an average user population. Before the results are studied, the input parameters are given.

4.3.3.a Input parameters

As indicated before, the available movies are MPEG-2 coded, so that 300 videos can be transported in one GbE signal. Two GbE signals are combined in one wavelength and eight CWDM wavelengths are multiplexed in one fiber. Possible equipment costs (in base units u) and number of HP on the head ends are summarised in Table 4.2.

Equipment cost	
installation server	25 u
1 server port	25 u
1 switch port	1 u
1 mux1	4 u
1 mux2	2 u
Number of HP	
head end A	100k
head end B	45k
head end C	20k
head end D	55k
head end E	15k
head end F	65k

Table 4.2: Input parameters

4.3.3.b Results

The design for the ring network with the above-mentioned parameters is shown in Fig. 6. At each of the head ends, the amount of expected traffic in GbE signals (99% interval from the Normal distribution), the maximum number of GbE

signals and the number of HP are given. The upper border of the 99% interval is used as the input parameter for the ILP model (v_d in Equation (5)). We notice that all GbE signals that are completely filled with video streams are sent end-to-end (E2E), while not completely filled GbE signals are combined at the server and split at several switches at the head ends (LbL). This corresponds to the hybrid grooming strategy proposed in our traffic model.

In this case, only one video server is installed on the network, in the head end with most users. In three head ends, a 3-port switch for link-by-link traffic has to be installed. The total cost on the metro network for this design is 560 u (25 u server installation cost, 450 u server port cost, 9 u switch port cost and 76 u CWDM equipment cost) to offer iVoD services to 100k subscribers (33% of all 300k HP). An additional cost for QAM devices with three RF channels per node on the HFC network, or 900 RF channels in total, also has to be taken into account. The cost for one RF channel can be estimated at about 0.5 u, so the total cost for the access network part is 450 u. The total installation cost is therefore 1010 u, about 0.01 u per subscriber.

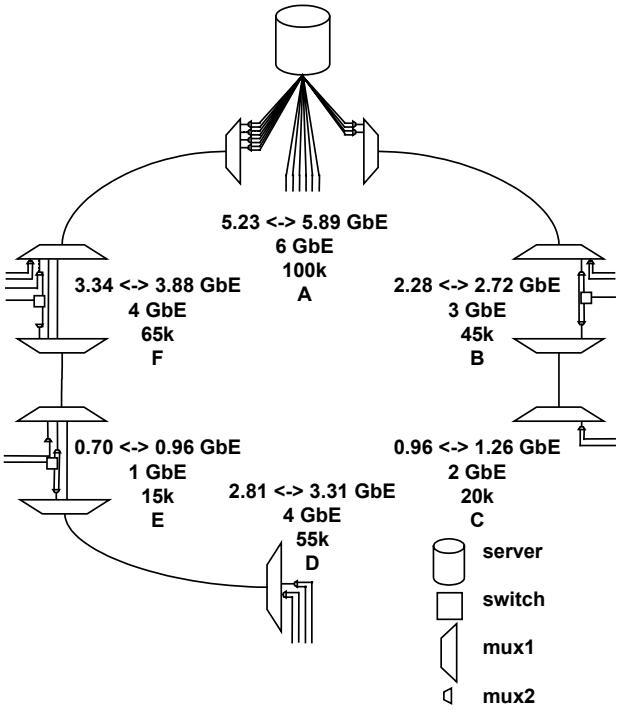


Figure 4.6: ILP solution for a ring network with 6 head ends (installation of servers, switches and WDM equipment)

4.4 Network design tool

Since this network design problem is an NP-complete problem, calculation times for the ILP model are increasing exponentially for larger networks or growing user demands. Besides that, no (straightforward) linear equations can be introduced to include regeneration of optical signals in this model. Furthermore, the ILP model does not take statistical multiplexing on the metro network itself into account. As input parameters for the demand per head end, the upper border of the 99% interval is used and the aggregated traffic for multiple head ends is determined as the sum of those values. According to our Erlang model however, the variance for the aggregated traffic (and thus the upper border of the 99% interval) is relatively smaller than for the individual demands.

Therefore, a network design tool, based on a simplified version of the ILP model, has been developed. The main focus of the heuristic for this tool is on minimising the major costs: number of server ports and number of RF channels. The only differences with the exact ILP solution can be found in the placement of the CWDM equipment. As a result, the heuristic is maximum 1% less optimal than the ILP solution (if only iVoD is considered), as the simulations show (see below).

Other VoD services than iVoD will also be discussed, as well as their impact on the installation cost on the access network.

Calculation times are never longer then several minutes for the simulations presented in this chapter, while the ILP model sometimes needed more than a day.

4.4.1 Heuristic

The tool makes use of an exhaustive strategy to find the optimal design of the network. All possible combinations of server placements and choices for VoD services are calculated. For each of these combinations the optimal installation of CWDM equipment and switch ports is determined, as described below. Of all possible configurations, the cheapest one is chosen as the final design. First we describe how traffic for the different VoD services is handled.

4.4.1.a Unicast

A first part of the algorithm describes how it deals with unicast traffic, like iVoD. Totally filled GbE signals are sent end-to-end (E2E) from the server to the head ends. For the partially filled GbE signals, sent link-by-link, a similar approach as in the ILP model is used.

From the simulations of the ILP model, we learned that for similar input parameters as described in Table 4.2, the number of switch ports needed per head end is either 0 (no GbE signals switched), 3 (one GbE signal switched), 4 or 5 (two GbE signals switched), never more. Therefore, we combine the partially filled GbE signals in groups of one or two GbE signals. The optimal combination (minimal number of GbE signals required at the server) is found through a brute force calculation. An example is shown in Figure 4.7: at the right side, one GbE signal is sufficient (0.2 + 0.7 = 0.9), while two GbE signals are sent link-by-link at the left side (0.6 + 0.5 + 0.8 = 1.9), together with one end-to-end signal (0.8 GbE). This way four server ports are enough to handle the LbL traffic.

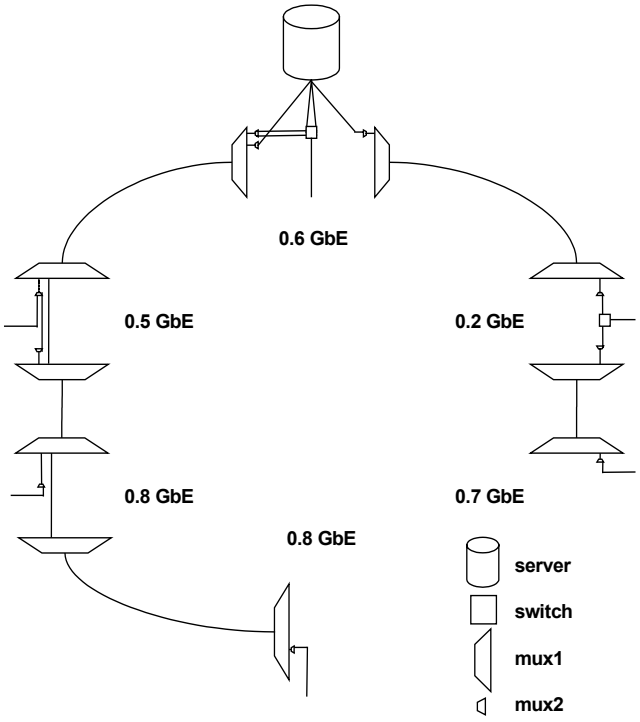


Figure 4.7: Strategy for link-by-link iVoD traffic (the partial GbE signals at each head end are given)

When the route of all GbE signals is determined, the necessary CWDM equipment is added accordingly. This equipment now also includes elements for regeneration of the optical signals (e.g. every 80 km), at a cost of about 2 u per wavelength.

4.4.1.b Broadcast

Unicast services, like iVoD, are much more user-friendly than broadcast services, since users can pause, fast forward and rewind video streams at any time. iVoD streams are sent on different unicast RF channels to all nodes. Broadcast traffic (nVoD) however requires much less bandwidth on the network, since all user requests during a certain period are served at once after a fixed "stagger time" (e.g., 15 minutes). This way only six copies of each nVoD video of 90 minutes are present on the network at any given moment, probably much less than the number of simultaneous requests for that video (see also Figure 4.9). A problem here is that broadcast channels have to be available at the HFC network. RF channels carrying nVoD traffic can then be split at the head ends and sent over broadcast channels to all nodes connected to that head end, as shown in Figure 4.8.

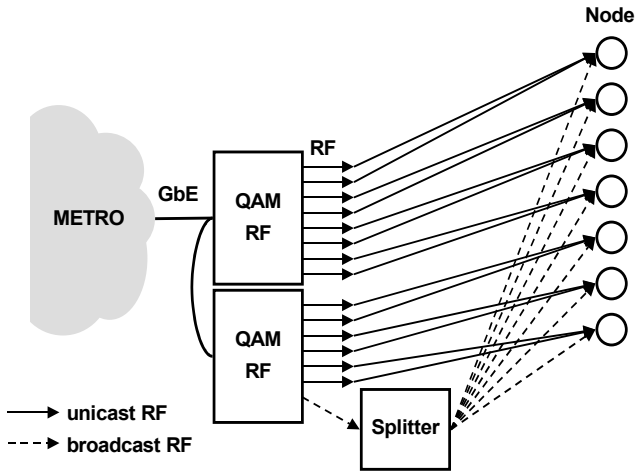


Figure 4.8: Difference between unicast and broadcast traffic on the HFC access network

Other solutions, like virtual VoD (vVoD, a VoD solution similar to Switched Broadcast [7]), try to combine the benefits from both worlds. The videos are still broadcast on the WDM network, but with a shorter stagger time (e.g., five minutes, this means 18 copies of each 90 minutes video), and only the locally requested videos are streamed on the access network, so that no broadcast channels have to be available. This however requires intelligent routing at the head ends.

In the design tool broadcast traffic is integrated by dedicating a certain number of GbE signals on the metro network to broadcast traffic (nVoD or vVoD). The optimal number is again found through an exhaustive strategy.

Wavelength adapters (WLA) are used to drop-and-continue one or two broadcast GbE signals at the head ends. The cost for one WLA can be estimated at 1 u.

4.4.1.c Personal Video Recorder

When VoD customers have a personal video recorder (PVR) at their set-top box (STB) at home, part of its hard disk could be used by the content provider to store popular videos before they are requested. This way a significant amount of network traffic can be avoided during peak hours.

4.4.1.d Combination of VoD services

When different VoD services are available, a choice has to be made for each video to determine which service has to be used.

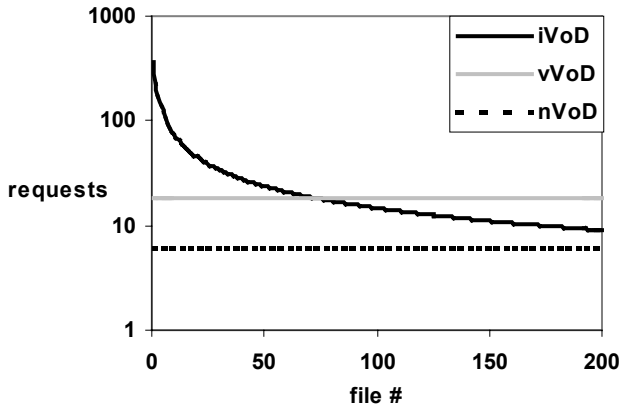


Figure 4.9: Choice between iVoD and vVoD for video files (the videos are ranked according to popularity)

The solution that results in the cheapest solution (the lowest number of outgoing RF channels at the head ends and server ports on the metro network) is shown below:

1) The most popular videos should be pushed to the STBs (if a PVR service is available) at the subscribers home, so that traffic for these videos can be avoided.

2) The next most popular videos should be sent as nVoD videos, if any broadcast channels are available at the HFC network. nVoD causes the least traffic on the metro network (only six simultaneous streams per video) and at the HFC network (broadcast RF channels can be split at the QAM devices at no extra cost to all optical nodes at one head end).

3) For the rest of the videos, the choice between vVoD and iVoD depends on the load on the metro network, since both services have an almost equal load on the HFC network. Videos that are requested more than 18 times (for a stagger time of five minutes) on the total ring network are sent using vVoD, the rest using iVoD. This is also demonstrated in Figure 4.9 for a total of 5000 simultaneous requests (present on an average network with 300k HP): the videos in the top 74 that are not yet stored on the PVRs or sent using nVoD, should be transported using the vVoD service (if available).

4.4.2 Simulations

First the network design for a standard configuration, with the same input parameters as given in Table 4.2, is determined and compared to the one shown in Figure 4.6. This design will be the starting configuration to compare the other results to.

4.4.2.a Standard configuration

In this configuration only iVoD services for the 200 available MPEG-2 videos are offered. Our network tool then gives exactly the same results as shown in Figure 4.6.

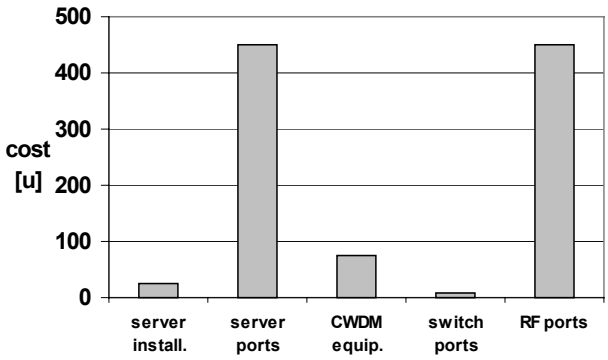


Figure 4.10: Total installation costs for the standard configuration (server ports and RF ports generate the major costs)

The total cost is again calculated as 1010 u. Note that the actual cost for CWDM equipment will be a bit higher, due to costs for the CWDM shelfs, client plugins, etc. that are not included in this model. The main cost however is still caused by the server ports and the RF devices, as Figure 4.10 indicates.

4.4.2.b Server installation costs

When the server installation costs are set to less than 8 u, a server is placed in every head end of the ring network. In this case, no costs for switches or CWDM equipment is required, since all traffic is sent directly from the local server to the QAM devices. The total number of server ports has now increased to 20 (instead of 18), because of the loss in statistical multiplexing. For higher values than 8 u, only one server is installed.

4.4.2.c Broadcast VoD services

The introduction of vVoD decreases the total cost to 853 u (15% gain). Now the 33 most popular videos are sent with vVoD and the rest with iVoD. nVoD can only be used if broadcast channels are available on the HFC network. Even then offering nVoD is not always profitable, since it increases the number of required RF channels on the devices. Only if six or more broadcast channels are present, introducing nVoD becomes beneficial, because the number of unicast RF channels per node decreases (from 3 to 2 in case of six broadcast channels). The total number of RF channels per average head end is then 106 (50·2 unicast + 6 broadcast) instead of 150 (50·3 unicast + 0 broadcast).

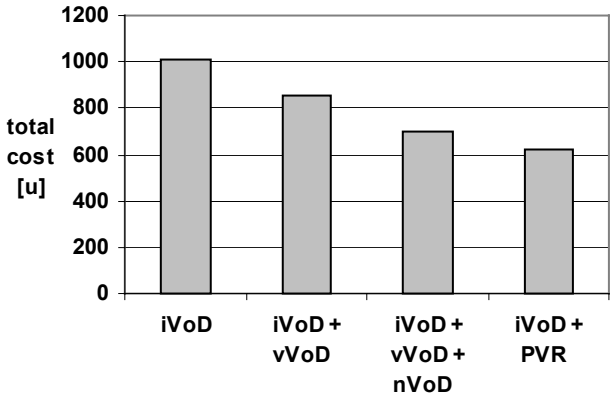


Figure 4.11: Total installation costs for different VoD services (iVoD, nVoD, vVoD, PVR)

In that case the 10 most popular videos are transmitted using nVoD, 46 with vVoD and the other 144 videos with iVoD. The total cost is than 698 u (30% gain). The different situations are compared in Figure 4.11.

4.4.2.d PVR service

If all subscribers have a PVR (100% penetration) and hard disks are 100GB in size (about 37 MPEG-2 videos, representing 52% of all traffic), a gain of almost 40% can be obtained (total installation cost: 625 u), compared to the standard configuration (see also Figure 4.11).

4.4.2.e Video codec

In case MPEG-4 video streams are used, the total cost is reduced by almost 60% to 428 u. This is because 800 streams can be carried in one GbE signal (instead of 300) and 30 in one RF channel (instead of 10). MPEG-4 streams have a bandwidth of 1.5 Mbps, while 3.8 Mbps has to be reserved for an MPEG-2 video.

4.4.2.f Network size

Changing the network size to nine head ends (50000 HP per head end on average) still does not increase the number of servers. The total cost is higher (1523 u).

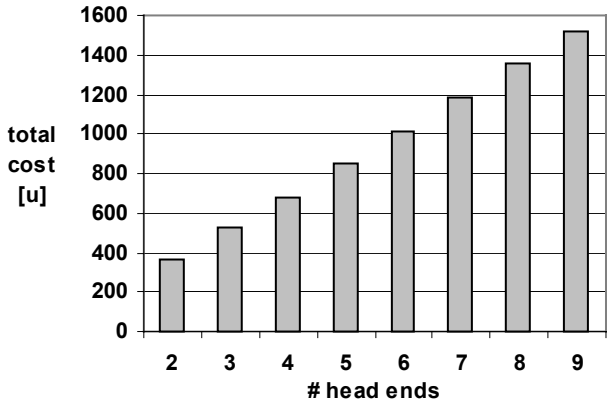


Figure 4.12: Total installation costs for different network sizes (the installation cost per subscriber remains constant)

A smaller network (three head ends) also only requires one server (total cost 534 u). The installation cost per subscriber remains about 0.01 u (Figure 4.12).

4.4.2.g Video popularity distribution

When we change the video popularity, no differences occur when only iVoD is available. In case of broadcast services, a gain can be achieved when the most popular videos become even more popular, because the amount of network traffic for those broadcast videos remains the same anyway. When we set the Zipf parameter to 1.0 instead of 0.7, half of the requests are made for the top ten videos instead of the top 33. This causes a decrease of 10% in the total installation cost (628 u instead of 698 u).

4.5 Conclusions

A decentralized network design for VoD services on Ethernet-based WDM networks has been presented in this chapter. By dividing the network into regional metro networks with a ring topology, the core network can be offloaded. The installation of a single local server per metro network appears to be sufficient in most cases.

Introducing PVR or broadcast VoD services besides interactive VoD can further decrease the installation costs. The gain in deployment costs might have to be evaluated against other economical perspectives, such as feasibility and user-friendliness. The influence of other parameters, such as video codec, content popularity and equipment costs, also have an important influence on the network design.

References

[1] J. Robadey, "Planning of content delivery networks for video on demand", Swisscom Innovations, Proc. of NOC 2004 (Eindhoven, The Netherlands, July 2004), pp. 352-358.

[2] J. Shreeram, "VoD everywhere! Considerations in Transport Methods for Scalable VoD/SvoD Deployment", Scientific Atlanta white paper, October 2002.

[3] T. Wauters, D. Colle, E. Van Breusegem, S. Verbrugge, S. De Maesschalck, J. Cheyns, M. Pickavet, P. Demeester, "Virtual topology design issues for variable traffic", IEICE Electron. Express, vol. 1, no. 12, pp. 328-332, September 2004.

[4] R. Guérin, H. Ahmadi, and M. Naghshineh, "Equivalent Capacity and Its

Application to Bandwidth Allocation in High-Speed Networks", IEEE JSAC vol. 9, no. 7, pp. 968-981, September 1991.

[5] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", Proc. of IEEE Infocom 1999, pp. 126-134, March 1999.

[6] P. Backx, T. Wauters, B. Dhoedt, P. Demeester, "A comparison of peer-to-peer architectures", Proc. of Eurescom Summit (Heidelberg, Germany, October 2002), pp. 215-222.

[7] N. Sinha, R. Oz, "The Statistics of Switched Broadcast", Proc. of SCTE Conference on Emerging Technologies (Huntington Beach, CA, January 2005).

[8] Cplex, http://www.ilog.com/products/cplex/

5 Access network design and replica placement for time-shifted television

5.1 Introduction

In the previous chapters, network solutions for video on demand services were presented for different network technologies. The available content, such as video films or archived television programs, is offered through an interactive service, that supports VCR-like commands such as pause, fast forward and rewind. Traditional services for live television do not offer such interactivity, since they are not supported by the broadcast technologies used. Some degree of personalization can be introduced through Private Video Recorders (PVRs), but these storage devices are difficult to use, have to be programmed in advance and can be expensive. Furthermore, the throughput capacity on this storage device and on the access link to the end-users's home is limited.

We therefore propose a network PVR solution, where the user interactivity is supported through a proxy based time-shifted television (tsTV) service. Time-shifted TV is a service that enables the end-user to watch a broadcasted TV program with a time shift, i.e. the end-user can start watching the TV program

from the beginning although the broadcasting of that program has already started or is already finished. They can also pause, and rewind live TV programs (possibly only within a relatively small window). In this work we show that only small, diskless caches are required to support this service.

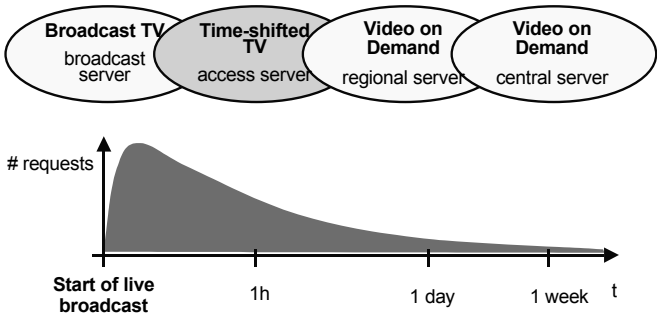
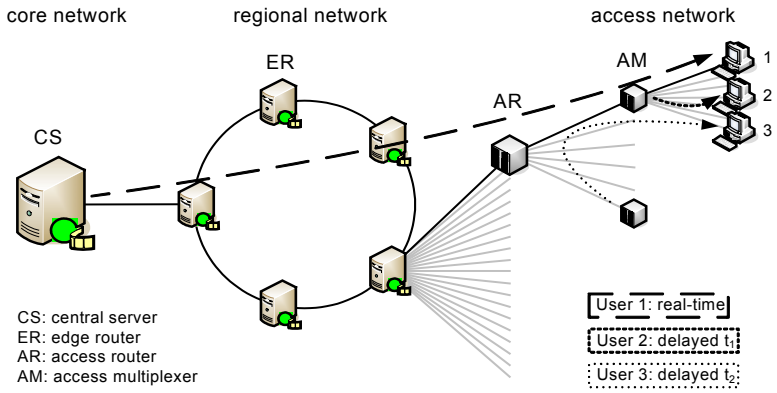


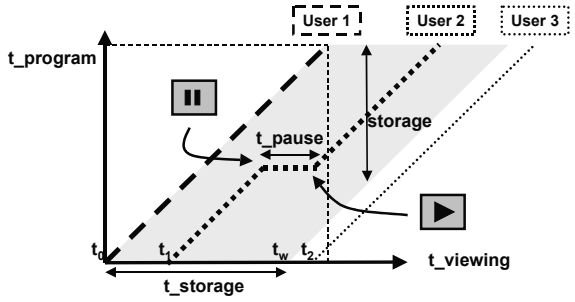
Figure 5.1: Delivery mechanisms for IPTV

As shown in Figure 5.1, the popularity of a television program typically reaches its peak value within several minutes after the initial broadcast of the program and exponentially decreases afterwards. This means that caching a segment with a sliding window of several minutes for each current program can serve a considerable part of all user requests for that program. Therefore, our new network based time-shifted television (tsTV) solution uses low cost distributed streamers with limited storage capacity. These streamers can be located at the proxy caches and store segments of the most popular content (TV programs), so that all requests arriving within these intervals can be served by the cache from start to finish.

In Figure 5.2a and Figure 5.2b, user 1 is the first to request a live program on a certain TV channel and gets served from the central server. Afterwards, other requesting users (e.g. user 2) can be served by the proxy, as long as the window of the requested program is still growing. After several minutes, the window stops growing and begins sliding, so that user 3 cannot be served anymore and will be redirected to the (central or regional) server or, in case of co-operative caching, to a neighbour proxy with the appropriate segment, if present. Pausing (parallel to the horizontal axis) can also be supported within the segment window, as well as fast forward or rewind (parallel to the vertical axis).



(a)



(b)

Figure 5.2: Time-shifted television: (a) typical network topology and (b) tsTV streaming diagram

The remainder of this chapter is structured as follows. Related research work is discussed in section 5.2. Section 5.3 presents an analytical model of the sliding-interval caching problem with fixed window sizes, for comparison with our caching algorithms and to have an initial estimate of the required storage space. The next section 5.4 introduces and evaluates our sliding-interval caching algorithms, for both stand-alone and co-operative caching. The location and the size of the different segments at the proxy caches are determined. Experimental results are obtained using a discrete event simulator. In section 5.5, the RTSP implementation is briefly discussed, while conclusions are presented in section 5.6.

5.2 Background and related work

As stated in Chapter 3, previous studies on proxy caching techniques [2] or distributed replica placement strategies for CDNs [6,7] show that greedy algorithms that take distance metrics and content popularity into account perform better than more straightforward heuristics such as LRU (Least Recently Used) or LFU (Least Frequently Used). Similar metrics are also of importance for storage techniques for sliding windows.

Segment-based caching techniques have been studied extensively for streaming media, due to the huge size of multimedia streams compared to traditional web objects. A survey on different segment-based strategies such as prefix caching, segment caching, rate-split caching and sliding-interval caching has been presented in [2]. The main goal of prefix caching is to reduce the start-up delay by caching the initial portion of the stream at the proxy. This paradigm is generalized by segment caching, where cache decisions are made for a series of segments of the stream. In rate-split caching, the partitioning is done along the rate axis, instead of along the time axis. This way, the cache takes care of the peak rates in VBR streaming, while the backbone only has to cope with the lower constant rate. Of particular interest for this study is sliding-interval caching [3], where the cached portion of the stream is initially a growing prefix, but afterwards a dynamically updated sliding interval. This way, consecutive requests can be served from start to finish within this window. A more advanced aspect is the use of co-operative proxy caching [4], where a better performance than with independent proxies can be achieved through load balancing and improved system scalability. In this case it is important to continuously keep track of cache states. Note that contrary to standard co-operative proxy caching, there is no need to switch to segments on other proxies when using co-operative proxy caching with sliding intervals. Similar peer-to-peer caching techniques have also been introduced in streaming CDNs, where whole files are stored instead of segments [5]. Several studies such as [8] have been investigating the implementation of segment-based caching techniques on proxies using the RTP / RTCP / RTSP protocol suite.

In our work, we present a novel sliding-interval caching technique and combine it with various collaboration schemes: stand-alone caching, hierarchical caching and co-operative caching. Our caching algorithms are included in a transparent proxy implementation. A demonstrator of an IP aware multi-service access network, including this prototype tsTV setup, has been presented in [9].

5.3 Analytical approach

Before presenting our sliding-interval caching algorithm, we introduce an analytical model of a tsTV solution based on sliding-interval caching with fixed window sizes, offering a method to estimate the required storage space in the network.

For each available TV program, a sliding interval of several minutes is stored on a cache between the server and the end users. This way, all user requests made during these first minutes of each program can be served from start to finish by the proxy. All other requests are redirected to the server. Our goal is to determine the cache's hit rate (the number of requests served by the cache, divided by the total number of requests) analytically.

5.3.1 Model parameters

Consider a model where each TV program is characterized by a start time τ_i, a duration T_i and a function $\lambda_i(t)$, representing the request arrival rate for this program (the total number of requests per second). $N(t)$ denotes the total number of programs with $\tau_i \leq t$. The proxy cache I, placed between the server and the clients, contains the first X minutes of any currently streaming file with $t - T_i \leq \tau_i \leq t$.

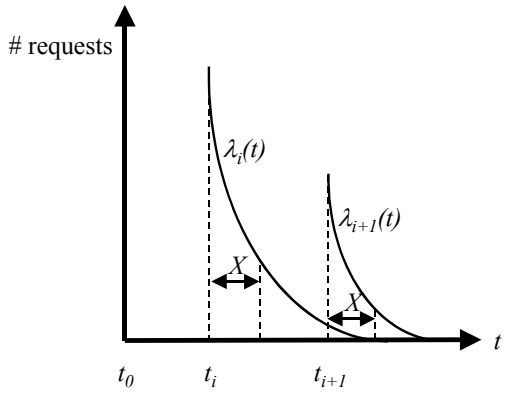


Figure 5.3: Parameters in the storage model for TV programs

5.3.2 Cache hit rate

We derive an expression for the hit rate of cache I, $h_I(t)$. Consider further the time period $|t, t + \Delta t|$, then the total number of requests is given by

$$\sum_{i=1}^{N(t)} \lambda_i(t)\Delta t \tag{1}$$

To find the total number of successful requests (i.e. requests that can be served by the cache) for the currently broadcasted program j in a single channel situation, we assume a uniform distribution for τ_j and make the following observations:

- these requests have to arrive at most X minutes after τ_j
- only a fraction X / T_j of the requests is served from cache I

Therefore the total number of successful requests is given by

$$\lambda_j(t)\Delta t \frac{X}{T_j} \tag{2}$$

Averaging over all programs j for which $t - X \leq \tau_j \leq t$, multiplying by the total number of channels K and supposing that popularity and duration are uncorrelated, we obtain the following expression:

$$h_I(t) = K \frac{<\lambda_j(t)>^*}{<T>} \frac{X}{\sum_{i=1}^{N(t)} \lambda_i(t)} \tag{3}$$

with <>* denoting averaging, on the condition that $t - X \leq \tau_i \leq t$. Supposing further that λ_i is a separable function of i and t, such that $\lambda_i(t) = \lambda_i f(t - \tau_i)$, with $f(t)$ a normalized function such that $f(t) = 0$ for $t < 0$ and

$$\int_0^\infty f(t)dt = 1 \tag{4}$$

we can write:

$$\begin{aligned} &<\lambda_j(t)>^* = <\lambda_j><f(t-\tau_j)>^* \\ &= \frac{<\lambda_j>}{X} \int_0^\infty f(t)dt \end{aligned} \tag{5}$$

as long as $X << T >$. Hence,

$$h_I(t) = K \frac{< \lambda >}{< T >} \frac{\int_0^X f(t)dt}{\sum_{i=1}^{N(t)} \lambda_i(t)} \tag{6}$$

Further consider a time period P, then the total number of broadcasted programs is $N(P) = KP/<T>$. Suppose a user group of size G, each requesting r programs per second on average, then the total number of requests is given by GrP. Therefore, the average number of requests for a long enough period of time will satisfy

$$< \lambda >= \frac{GrP}{N(P)} = \frac{GrP}{KP / < T >} = \frac{Gr < T >}{K} \tag{7}$$

On the other hand, the total number of requests per time unit is given by

$$\sum_{i=1}^{N(t)} \lambda_i(t) = Gr \tag{8}$$

simplifying our expression for the cache I hit ratio to

$$h_I = \int_0^X f(t)dt \tag{9}$$

Taking for $f(t)$ an exponentially decreasing function $b\exp(-bt)$ (for $t>0$), we get

$$h_I = 1 - e^{-bX} \tag{10}$$

as long as $X << T >$. The size of cache I is simply KX.

Figure 5.4 shows the server load for different values of the cached segment size. If the content popularity only decreases slowly (e.g. by 10% after each interval, $b = -\ln(0.9)/\Delta$), the server load cannot be reduced significantly. When the content

popularity is halved after each interval Δ (b = -ln(0.5)/Δ), the server load is halved as well when the segment size is Δ. It is then given by

$$1 - h_I = \left(\frac{1}{2}\right)^a \tag{11}$$

if $X = a\Delta$.

When the content popularity is halved after each interval Δ (b = -ln(0.5)/Δ), the server load looks like presented in Figure 5.4. It is given by ($X = a\Delta$)

$$1 - h_I = \left(\frac{1}{2}\right)^a \tag{11}$$

Similar results for the server load can be found using the sliding-interval caching algorithm presented in the following section (comparable to the "s -> c1" curve in Figure 5.10a, for stand-alone caches at level 2).

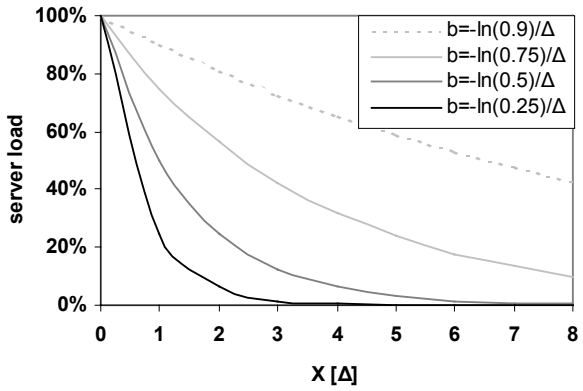


Figure 5.4: Analytical solution for the server load, for different values of the segment size

We can conclude that in case of an exponentially decreasing temporal content popularity, the server load decreases proportionally, for increasing segment sizes.

5.4 Sliding-interval caching algorithm

Our caching algorithm for tsTV services is presented in this section. Since we assume that in general only segments of programs will be stored, cache sizes can be limited to a few gigabyte (corresponding to a few hours of streaming content). This way smaller, diskless streaming servers can be deployed closer to the users, without increasing the installation cost excessively.

The storage metrics (popularity and distance) used in this caching algorithm are similar to those used in the *SF* algorithm used for VoD (Section 3.6.1.a).

5.4.1 Basic principle

We propose that the cache is virtually split up in two parts: a small part *S* and a main part *L*. Part *S* will be used to cache the first few (e.g. 5) minutes of every newly requested (or broadcasted) program, mainly to determine its initial popularity. Its size is generally smaller than 1 GB (typically 1 hour of streaming content). Part *L* will be used to actually store the appropriate segments (with growing or sliding windows). The actual size of each segment in part *L* will be determined and, if necessary, adapted after each interval Δ (e.g. 5 minutes). After Δ, one of the following decisions has to be taken:

- let the segment grow (for very popular programs);
- let the segment slide (to finish the current requests, for less popular programs);
- drop the segment (for unpopular programs, with no current requests to be served).

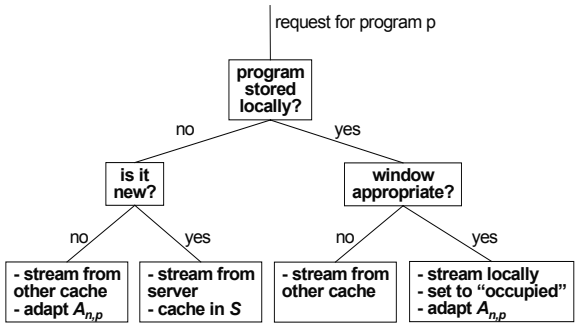


Figure 5.5: Basic principle of the tsTV caching algorithm at each proxy

Figure 5.5 shows the basic principle of the tsTV caching algorithm. During each interval Δ, program requests arrive at the different proxies. Each time, a

parameter $A_{n,p}$ will be updated in proxy n, for program p. In general, this parameter tries to determine the popularity of the program, while taking distance metrics into account. This means that a (segment of a) popular program might not be cached, because a nearby proxy already stores that (segment of the) program. $A_{n,p}$ is calculated as follows:

Each time a request for program p arrives at proxy n, $A_{n,p}$ is increased by 1 (only taking popularity into account) or by the hopcount between proxy n and the serving node (also taking distance into account).

After each interval Δ, first all segments (sliding or growing) with status set to "occupied" are stored in L. Afterwards L is filled with segments with growing windows for the most popular programs (i.e. with the highest values of $A_{n,p}$). All other segments are dropped, S is cleared and all values of $A_{n,p}$ are reset to 0.

Note that this basic principle should be extended in case of more fluctuating demand patterns than the exponentially decreasing popularity distributions assumed in this work.

5.4.2 Caching mechanisms

5.4.2.a Hierarchical caching

The access network part of Figure 5.2a is shown in Figure 5.6. When using hierarchical caching, every cache c1 (at the access multiplexer) forwards a request it cannot serve on to the next cache c2 (at the access router) on the path from the client to the edge server. All caches basically follow the caching scheme shown in Figure 5.5, independent of each other.

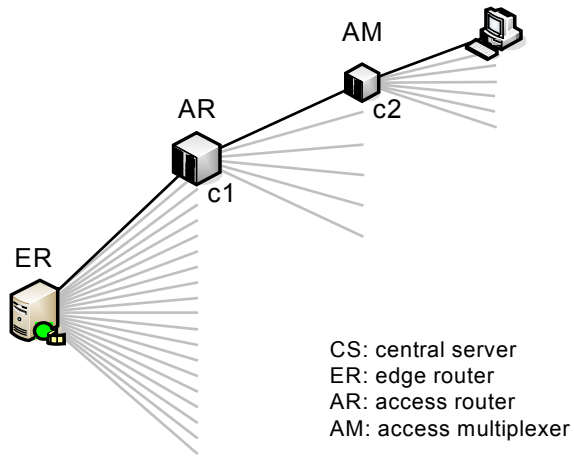


Figure 5.6: Basic access network topology

5.4.2.b Co-operative caching

Contrary to hierarchical caching, collaboration between caches on the same level, 1 or 2 in Figure 5.6 (at the access routers or at the access multiplexers), is made possible, by the introduction of peer-to-peer communication among these caches.

To benefit from this co-operation, the large part L in each cache is again virtually divided into two separate storage spaces. Part L_1 is used to store unique segments only, shared among all co-operating cache nodes. These unique segments are never duplicated: only the first cache that decides to store a segment in L_1 is allowed to do so. This way, all parts L_1 on all cache nodes represent one large cache, mainly to offload the central server. The second part L_2, if there is still storage space left, is then used to store segments that are locally most popular. The main goal of that part is to offload the access network links, used by the co-operative caching mechanism (requests served by L_1 on a neighbour cache). The basic caching scheme in Figure 5.5 is then only used for part L_2.

5.4.3 Numerical results for stand-alone caching

The proposed algorithms were implemented on a discrete event simulator and the results are discussed below.

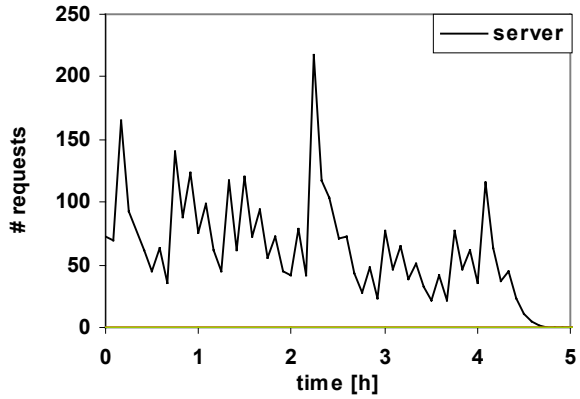
5.4.3.a Input parameters

To illustrate the stand-alone caching principle (with hierarchical caches), a first set of simulations was performed on one branch of the access network tree of Figure 5.2a: a regional server with two hierarchical caches (Figure 5.6).

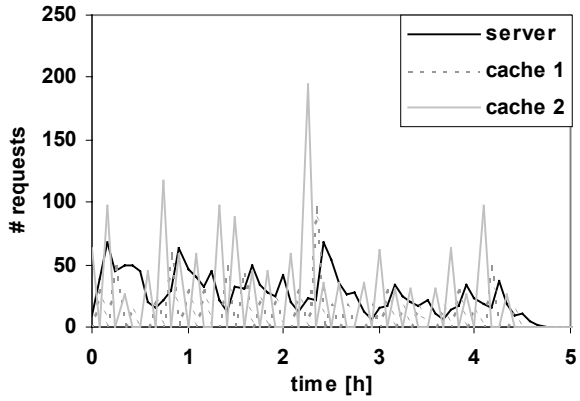
The regional server offers 20 channels: 5 very popular channels (80% of all requests), 5 less popular channels (10% of all requests) and 10 unpopular channels (10% of all requests). The top 5 channels are served as a tsTV service, the other channels through standard VoD technology on the regional server. The popularity of the programs per channel follows a Zipf-like distribution with parameter β = 0.7 (the popularity of the i'th most popular program is proportional to $i^{-\beta}$). This distribution is commonly used for content distribution [10,11] and TV viewing measurements like [12] confirm this trend. A total of 3000 requests are made during one evening, of which 200 for the most popular program on the most popular channel. The popularity of a program reaches a peak during the first interval Δ (= 5 minutes) and decreases exponentially afterwards (halved every interval Δ) (similar to Figure 5.1). Each channel offers 6 programs of 45 minutes per evening, with a streaming bandwidth of 2.5 Mbps.

5.4.3.b Server and cache load

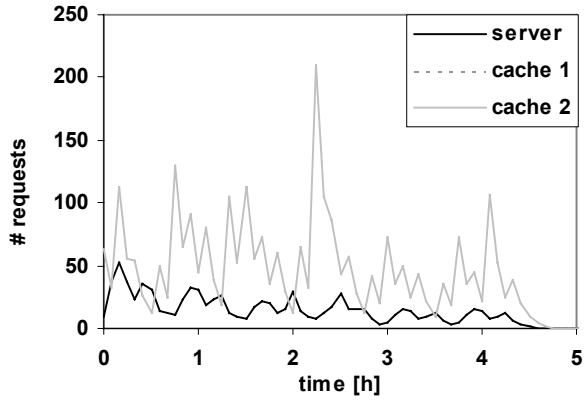
In Figure 5.7, the server and cache load are presented. Various peak values occur at the start of a new program on one of the TV channels. When both cache sizes are limited to 0.5 GB (*S* only: the number of channels times Δ or 25 minutes, Figure 5.7b), the server load is much lower than without caches (Figure 5.7a) and the caches serve most of the tsTV requests. What happens is that cache c1 (closest to the server) and cache c2 first store all 5-minute prefixes of each new program, but since cache c2 intercepts new requests afterwards, cache c1 will not receive new requests and drops these segments after Δ.



(a)



(b)



(c)

Figure 5.7: Server and cache load. All requests are made within 30 minutes. The cache sizes are 0 GB (a), 0.5 GB (b) and 4 GB (c)

Afterwards cache c1 will store the next 5 minutes of each program, while cache c2 is storing the sliding "occupied" windows from the first interval. This means that the caches serve all requests made during the first 10 minutes of each single program. For infinite cache sizes (or 4 GB or higher in this example, Figure 5.7c), the regional server only serves the VoD requests for channels 6 to 20. Cache c2 stores and serves all currently broadcasted popular programs, thereby effectively offloading the network.

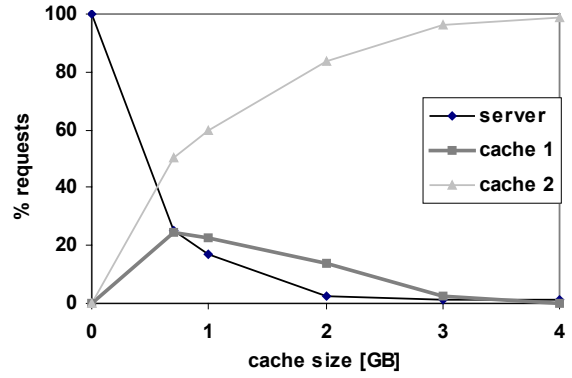


Figure 5.8: Relative server and cache load. All requests are made within 30 minutes

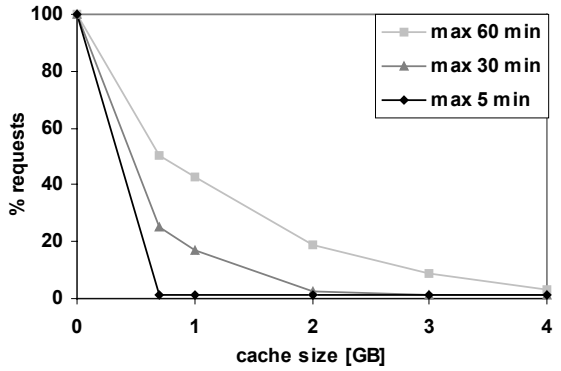


Figure 5.9: Relative server load for different values of the maximum request period

More detail on the regional server and cache load is given in Figure 5.8 (tsTV only, top 5 channels). Note that the server load never drops to 0, since at least the first request for a certain program has to be served from the regional server. In Figure 5.9 the server load is shown for different values of the maximum request period per program. Since no upstream links are used in these simulations, the bandwidth on the links can easily be determined from the server and cache load.

5.4.4 Numerical results for co-operative caching

The same caching principles can be applied for a co-operative caching mechanism, where caches on the same level of the broadcast tree can collaborate, using peer-to-peer protocols to exchange information on stored content. Contrary to stand-alone caching, where a request that cannot be served is forwarded to the next cache on the path to the central server (hierarchical caching), caches can now forward requests to caches on the same level. However, the decision on when to store a certain fragment not only depends on the value of $A_{n,p}$, but also on the source node serving the request. Two different approaches can be distinguished.

The first heuristic only takes the values of $A_{n,p}$ into account ("Cache from All sources", *CfA*). This means that the storage space $L = L_2$, so that most caches store the same fragments, since content popularity is similar for most nodes. The numerical results will therefore be comparable to the results for stand-alone caching.

The second heuristic also takes the values for $A_{n,p}$ into account, but never stores content that is already stored on another cache ("Cache from Server only", *CfS*),

so that $L = L_1$. This way the central server will be offloaded considerably, even with small caches, but many requests will have to be served by other caches over the access network links.

Both alternatives have their benefits. The first one is optimal in case of larger caches, since all content is then stored locally, which offloads the server as well as the access network. The second one minimizes the server load in case of small caches, since all content is quickly spread over all caches. The optimal heuristic however takes the best of both worlds, storing unique content segments in L_1 and locally popular segments in L_2. This heuristic is called "Cache from Elected sources" (*CfE*). This way the central server load is always minimized first. The access network load can be reduced afterwards, if the cache space is large enough.

As a consequence, Figure 5.4 can be used for capacity planning for the access network caches. The desired server load determines the size of the virtual cache that consist of all small parts L_1. By dividing the size of this virtual cache by the number of caches in the access network, taking the number of TV channels into account, the size of L_1 for each cache is found. Extra storage capacity can then be used for part L_2, to offload the access network links.

5.4.4.a Input parameters

The input parameters for the simulations are the same as in the previous section. The network topology (shown in Figure 5.6) consists of a regional server, one node at level 1 (without storage capabilities) and 6 proxy caches at level 2. The level 1 node is connected to the level 2 caches with bidirectional links, so that cache co-operation is possible.

Note that no storage space is available at the level one node so that the results of the simulations for cache co-operation are not influenced by hierarchical caching.

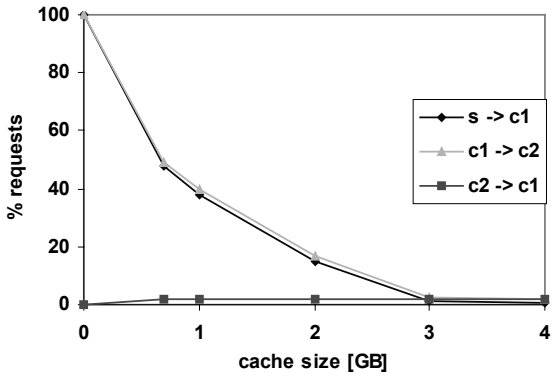
The cost of using the link from the central server to the node at level 1 has been set to a value higher than 1 (the cost of an access network link). This way the central server will be avoided when the requested segment can already be found on a neighbour level 2 cache (when calculating the shortest path using the weighted Dijkstra algorithm).

5.4.4.b Server, cache and network load

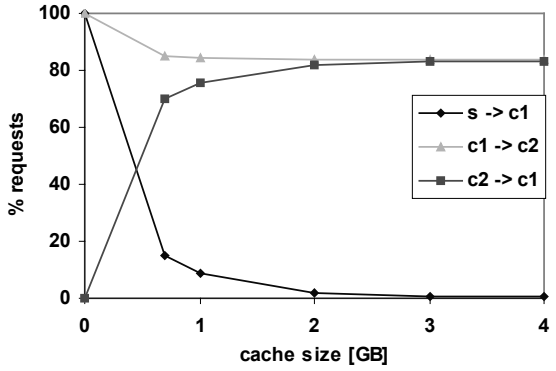
In case of stand-alone caching, the network bandwidth can easily be determined out of the cache and server load (Figure 5.8), since only downstream traffic is present on the access network. With co-operative caching, the uplinks in the access network are used as well. Figure 5.10 shows the relative load on the

network links, determined by the number of requests on each link divided by the total number of requests served.

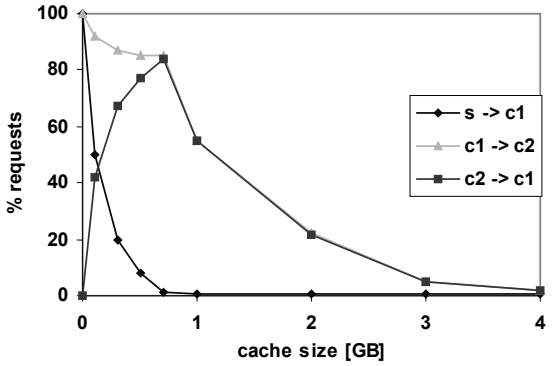
Using the *CfA* heuristic (Figure 5.10a), the server load is almost identical to the case where stand-alone caches on level 2 are used. The only difference is that the central server does not need to serve the first stream to all of the 6 proxies, but only to one of them. Again the central server load for the tsTV channels drops to (almost) zero when 4 GB caches would be used. The uplinks from the level 2 caches to node 1 are almost never used, since all caches store the same fragments. The results are therefore very similar as for stand-alone caching (remember the analytical results of Fig. 3, with 1GB = 10 minutes per channel = 2Δ).



(a)



(b)



(c)

Figure 5.10: Relative load (fraction of the total number of requests) on the links between the server and the level 1 node (s -> c1) and between the level 1 and 2 nodes (downlink c1 -> c2 and uplink c2 -> c1) for the CfA (a), CfS (b) and CfE (c) heuristics

When the *CfS* heuristic is used (Figure 5.10b), each 5-minute (Δ) fragment is only stored on one cache. This way, the central server load is already almost zero for the tsTV channels when only 0.5 GB caches are used. The total storage space is then 3 GB, therefore one could expect that the results for the central server load would correspond to the situation with 3 GB caches in stand-alone mode. This is not entirely the case, since it is possible that the first requests for a new program arrive at caches that have no storage place left in L_1. These first requests are then served by the central server. The load on the access network links ("c1 -> c2" and "c2 -> c1") is balanced.

The *CfE* heuristic (Figure 5.10c) offers the best of both worlds. The server load is reduced effectively, while, in case of larger caches, the access network is offloaded as well. The server load (link "s -> c1") is even lower then for the *CfS* heuristic. This is due to the RTSP request forwarding mechanism, allowing requests that arrive at a cache that has no storage space left in L_1, to be forwarded automatically to another cache with enough storage space. This way the virtual cache consisting of all parts L_1 is filled up in an optimal way.

5.5 tsTV service deployment

A transparent RTSP proxy for time-shifted TV has been implemented (in C++) by Wim Van de Meerssche for evaluation purposes. This section gives an

overview of the different components and protocols used and evaluates a prototype through performance measurements.

5.5.1 Functionality

In order to implement the proxy, its functionality is divided into logical parts (Figure 5.11). Each functional component is described in more detail below. The communication with the users and the central server includes messages containing data about which program or channel has to be streamed, or VCR like commands such as PAUSE and STOP. A protocol commonly used for this interaction is RTSP (Real-Time Streaming Protocol) [5]. The streams themselves are encapsulated and delivered with RTP (Real-Time Protocol), a standard protocol for live streamed media [6].

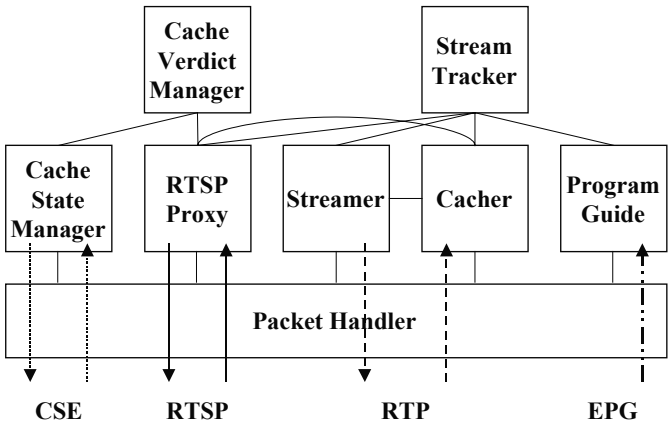


Figure 5.11: Overview of the different components in the proxy cache

5.5.1.a Cacher

The Cacher component is responsible for keeping track of the packet buffers which contain the stored streams. It receives RTP packets and stores them in the correct buffer, together with calculated parameters of the packet, such as offset in the buffer, and the time the packet was originally sent by the streamer. It also manages the removal of packets from full buffers.

5.5.1.b PacketHandler

The PacketHandler offers an interface to low level network functions, such as receiving and sending RTSP and RTP packets. It handles all low-level network interaction. Its function is to shield the other classes from operating system

specific code. Besides PacketHandler, no other class needs to know about sockets.

5.5.1.c Streamer

The Streamer component manages a list of clients and the packet buffers these clients need to receive packets from. The Streamer will run periodically, lookup which packet(s) each client needs to receive, retrieve those packets from the buffer, and send them to the correct client. In this process, IP addresses, ports and RTP sequence numbers of the packet to be sent are reset to values for this specific client.

5.5.1.d StreamTracker

The StreamTracker is basically a central storage, which gives access to data about RTSP streams. It keeps track of all RTSP URLs and the RTP media streams in each RTSP stream (for example one for audio and one for video). It uses SDP data to gather this information. SDP is received from the ProgramGuide and from RTSP describe replies.

5.5.1.e ProgramGuide

The ProgramGuide listens for announcements from the "EPG" (Electronic Program Guide) server, e.g. when a new stream is started. This data is required, since for each packet, the time it was originally sent by the streamer has to be known. The EPG currently used is in an early stage of development, and future extensions will include other functions, allowing to offer multiple programs per channel with the tsTV service. Since all program guide functions are implemented in this object only, portability to another (or even multiple) EPG is easy.

5.5.1.f RTSPProxy

The RTSPProxy processes RTSP packets originating from the clients. It generates responses, or if it cannot respond itself, forwards the request to another RTSP server. It translates the RTSP messages into requests for the CacheVerdictManager, and uses the Streamer and Cacher to execute the CacheVerdictManager's verdict. To do all this it needs to keep track of RTSP sessions. With each session information needs to be stored about which RTSP stream was requested, what the transport parameters for delivery to the client are, what the verdict was and how the streamer is streaming it.

5.5.1.g CacheVerdictManager

The CacheVerdictManager is an interface used to decide whether to cache or not. In this module, the actual implementation of the Caching algorithm is done.

Whenever a user sends a play request, it is translated into various parameters for the caching algorithm, and the algorithm will use these to decide to either cache the stream, or not.

5.5.1.h CacheStateManager

The CacheStateManager collects information about which content is stored on other caches and communicates the cache verdicts from the local CacheVerdictManager to the other caches. It updates this information through a centralized or distributed Cache State Exchange (CSE) protocol.

5.5.2 Detailed scenario

Figure 5.12 shows a detailed setup of a streaming session between the client, the proxy caches and the server. First, the client sends an RTSP request to the server, but this request is intercepted by the proxy. In a first scenario (Figure 5.12, 1a), the proxy does not store the requested fragment, forwards the request (with the destination IP address of the proxy) to the server, starts caching the stream from the server and forwards the RTP stream to the user. Afterwards, the proxy exchanges its new cache state in a distributed way to all other caches (Figure 5.12, 2a). In a second possible scenario (Figure 5.12, 1b), the proxy does not store the requested fragment and decides not to store the fragment locally. It forwards the RTSP request to another (proxy) cache, keeping the destination IP address of the client. The other proxy decides to forward the request to the server, caches the fragment locally and sends the RTP stream directly to the client. Afterwards, the new cache states are exchanged through a centralized CSE protocol (Figure 5.12, 2b). The second scenario shows how the co-operative caching algorithm (section III.B.2) can efficiently create one large virtual cache, using the "transparent RTSP request forwarding" principle.

5.5.3 Test setup and measurements

In this section, performance measurements on a prototype proxy are presented, implemented on an AMD AthlonTM 64 processor 3000+ (512MB RAM). The setup is similar to the one used for demonstration at BBE 2006 [9], shown in Figure 5.13.

Figure 5.14 shows the number of client RTSP requests that can be handled simultaneously by the proxy, already serving RTP streams (2.5Mbps) over a gigabit link (560Mbps throughput measured with Iperf [13]). The proxy uses high-priority RTP threads and low priority RTSP threads. We observe that the RTSP handling decreases linearly and fails at 190 simultaneous RTP streams (480Mbps), due to limited system resources.

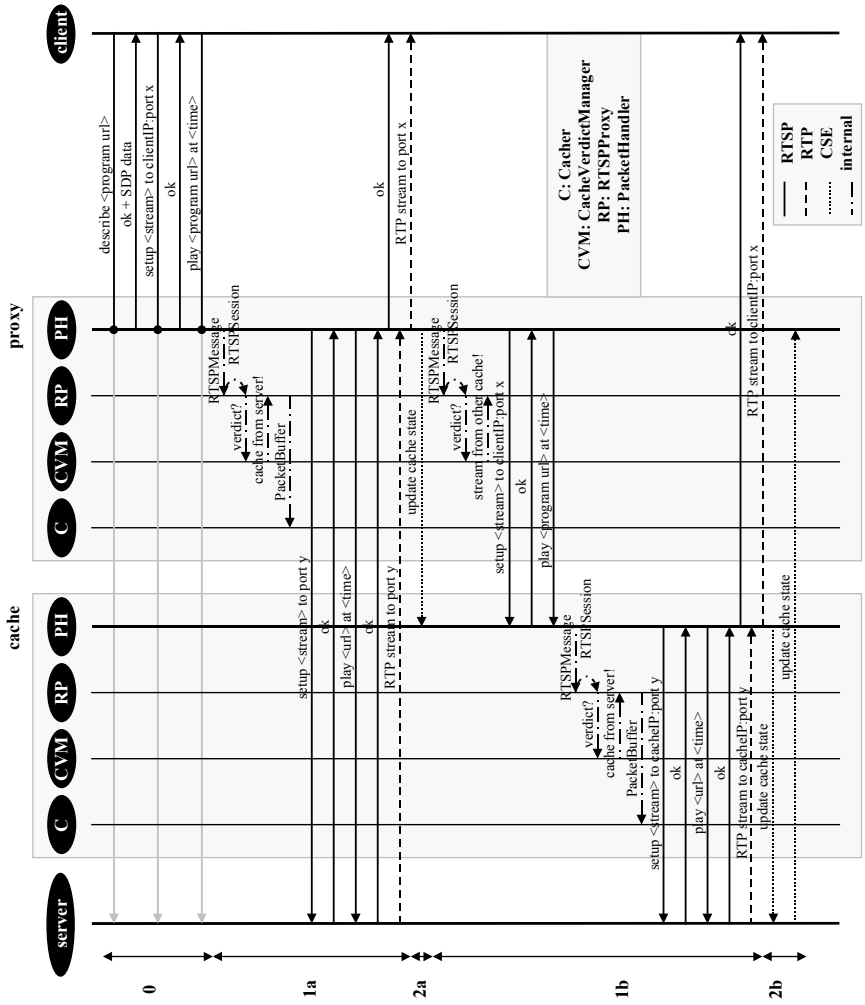


Figure 5.12: Detailed setup of a streaming session between client, proxy, any other cache and the server. The proxy caches the requested program from the server (a) or forwards the RTSP request transparently to another cache (b).

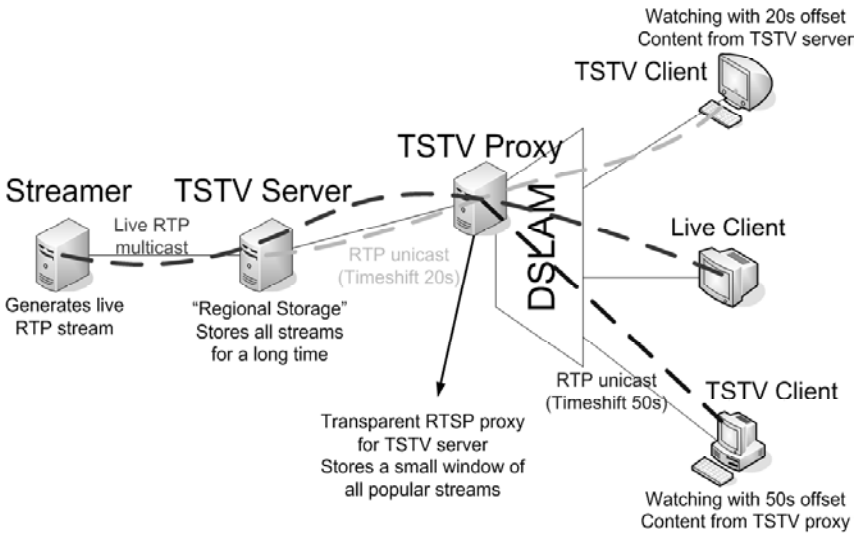


Figure 5.13: Demonstrator setup for tsTV

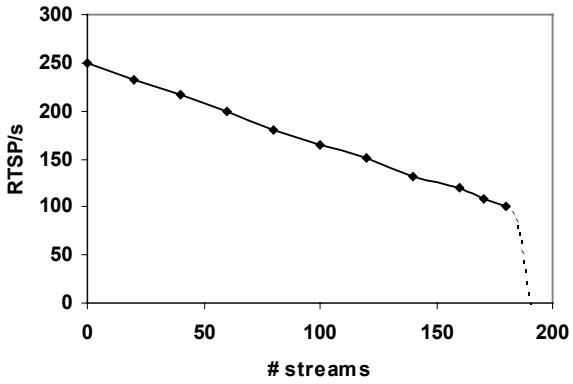


Figure 5.14: RTSP requests handling (AMD AthlonTM 64 processor)

Figure 5.15 shows the delay between a PLAY request sent by a PC client and the arrival of the first RTP packet at the PC client, for different configurations. Even when the proxy has to fetch the content from the server, the delay is never higher than 35 ms (1000 measurements per configuration). When the proxy acts as a mere router, the delay caused by the server (Darwin streamer [14]) is less than 1 ms. The delay on the network links between server, proxy and client is negligible.

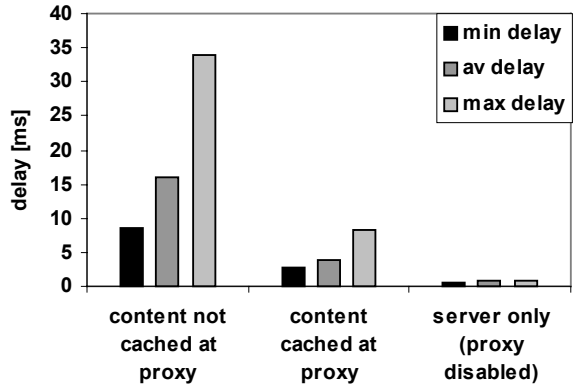


Figure 5.15: Delay between a client request and the actual start of the RTP stream on a client PC

5.6 Conclusions

In this chapter a novel architecture for a time-shifted television service is presented, as well as a sliding-interval caching algorithm for efficient storage. Cache decisions (on segment size, stored programs, …) at low cost distributed streamers are made after fixed learning intervals, based on popularity and distance metrics. Experimental results for a basic access network topology showed promising results in terms of server and network load, especially for co-operative caching. An RTSP proxy implementation has been introduced as well. The transparent RTSP request forwarding principle for co-operative caching further reduces the server load. A prototype integrating the caching algorithms has been built and evaluated through measurements.

References

[1] Akamai. http://www.akamai.com.

[2] J. Liu, J. Xu, "Proxy caching for media streaming over the internet", *IEEE Communications Magazine*, vol. 42, no. 8, August 2004, pp. 88-94.

[3] S. Chen et al., "SRB: Shared running buffers in proxy to exploit memory locality of multiple streaming media sessions", *24th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2004.

[4] Y. Chae et al., "Silo, rainbow, and caching token: Schemes for scalable, fault tolerant stream caching", *IEEE Journal on Selected Areas in*

Communications, vol. 20, no. 7, September 2002, pp. 1328-1344.

[5] D. Turrini, F. Panzieri, "Using p2p techniques for content distribution internetworking: a research proposal", *2nd IEEE International Conference on Peer-to-Peer Computing*, September 2002.

[6] M. Karlsson, C. Karamanolis, M. Mahalingam, "A Framework for Evaluating Replica Placement Algorithms", Technical Report HPL-2002, HP Laboratories, July 2002.

[7] T. Wauters, J. Coppens, B. Dhoedt, P. Demeester, "Load balancing through efficient distributed content placement", *NGI 2005*, April 2005, Rome, Italy.

[8] S. Gruber, J. Rexford, A. Basso, "Protocol Considerations for a Prefix-Caching Proxy for Multimedia Streams", *Computer Networks*, vol. 33, no. 1-6, 2000, pp. 657-668.

[9] E. Gilon, et al., "Demonstration of an IP Aware Multi-service Access Network", *BroadBand Europe 2005*, December 2005, Bordeaux, France.

[10] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", *IEEE Infocom*, 1999.

[11] P. Backx, T. Wauters, B. Dhoedt, P. Demeester, "A comparison of peer-to-peer architectures", *Eurescom Summit*, 2002.

[12] Broadcasters' audience research board, http://www.barb.co.uk.

[13] Iperf, TCP/UDP bandwidth measurement tool, http://dast.nlanr.net/projects/Iperf

[14] Darwin Streaming Server, http://developer.apple.com/opensource/server/streaming

6 HFC access network design for switched broadcast television

6.1 Introduction

In the previous chapter, a DSL based access network design for an IPTV service that supports user interactivity through a proxy based approach was presented. This chapter again focuses on IPTV, but studies an HFC access network service deployment (Figure 6.1). In order to minimize the installation costs for access network elements such as QAM devices, we introduce switched broadcast techniques [1], in a combination with traditional broadcast mechanisms. A similar network structure as in Chapter 4 is studied. At the edge of the metro network, gigabit ethernet (GbE) signals reach the head ends (HE) and are split at the QAM devices. Similar to nVoD, standard broadcast TV channels are broadcast on the metro network and sent through a splitter at the HE to all nodes. Switched broadcast TV channels are also broadcast on the metro network, but only the nodes with at least one user watching a particular channel actually receive that channel, sent as unicast traffic on an RF channel (with a capacity of several Mbps per RF channel, depending on the QAM modulation used), very much like iVoD traffic. Switched broadcast might be a bridge from the current broadcast network paradigm to a fully switched network, providing subscribers

with greater opportunities for personalization and operators with enhanced revenue opportunities, while preserving bandwidth.

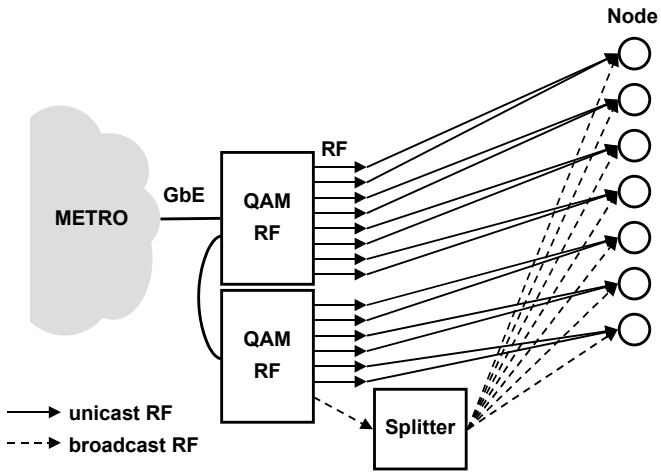


Figure 6.1: Typical HFC access network configuration

The remainder of this chapter is organized as follows. Section 6.2 presents our traffic model, based on content popularity and user behaviour, determining the number of simultaneously watched channels. In section 6.3 the access network design tool is described. Based on a set of simulations the influence of different parameters on the network design is studied in section 6.4. Each time the result is compared to a standard configuration. Section 6.5 concludes this chapter and presents ideas for future work.

6.2 Traffic model

This section presents an analytical traffic model that determines the distribution of the number of simultaneously watched TV channels during peak hour, given the number of user requests. This model can later on be used to find the number of switched broadcast and standard broadcast RF channels required at the edge.

6.2.1 User demand

A typical HFC access network configuration, as shown in Figure 6.1, consists of several tens of nodes per HE, each with about 1000 HP (homes passed). To determine the number of user requests per node during peak hour, the digital TV

market penetration is multiplied by the percentage of digital TV users active during peak hour.

As for other multimedia content, the popularity of the available TV channels is again represented by a Zipf-like distribution [2], where the popularity of the *i*'th most popular object is proportional to $i^{-\beta}$. We fit this distribution with data from a field trial in Belgium, where 118 TV channels were offered. The corresponding value for β is about 1.7 (see Figure 6.2). 50% of all requests are made for the most popular channel, 90% for the top 12.

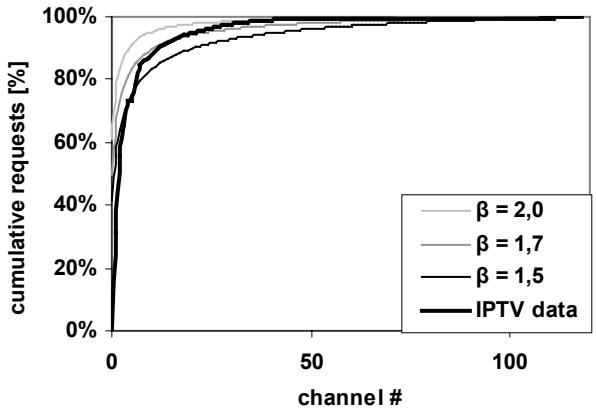


Figure 6.2: Cumulative Zipf-like TV channel popularity (ranked), compared for different values of β

6.2.2 Mathematical formulation

The goal of the traffic model is to set up a probability distribution for the number of simultaneously watched TV channels, for a given content popularity and user demand. In other words, we have to find how *R* user requests are distributed over *N* available TV channels.

6.2.2.a Variables

We define *N* variables X_n as follows: $X_n = 0$ if channel *n* is not requested, $X_n = 1$ if channel *n* is requested at least once. When $q(n)$ is the chance that a particular user request is made for channel *n*, the corresponding probabilities for X_n are:

$$\text{Prob}[X_n = 0] = \left(1 - q(n)\right)^R = p_1(n) \quad (1)$$

$$\text{Prob}[X_n = 1] = 1 - \left(1 - q(n)\right)^R = 1 - p_1(n) \quad (2)$$

since Prob[X_n = 0] is the chance that all requests are made for one of the N-1 other channels than channel n and this chance is (1-q(n)) for each of the R individual (and independent) requests. We define p_i as the chance that i particular channels are not requested:

$$p_1(n) = \left(1 - q(n)\right)^R \quad (3)$$

$$p_2(j,k) = \left(1 - q(j) - q(k)\right)^R, \ j \neq k \quad (4)$$

For a Zipf-like channel popularity, we know that $q(n)$ is given by:

$$q(n) = \frac{n^{-\beta}}{\sum_{i=1}^{N} i^{-\beta}} \quad (5)$$

6.2.2.b Solution

Our goal is to find the probability distribution for the total number of channels requested by at least one user, given by

$$Y = \sum_{n=1}^{N} X_n \quad (6)$$

Since Y is the sum of a large number of statistically independent variables (N variables in total), its distribution can be modelled by a Normal distribution (with mean μ and variance σ^2):

$$P(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y-\mu)^2}{2\sigma^2}} \quad (7)$$

To calculate μ and σ^2 we need the following formulas:

$$E[X_j] = 1 \cdot \text{Prob}[X_j = 1] + 0 \cdot \text{Prob}[X_j = 0]$$

$$= \text{Prob}[X_j = 1] = 1 - p_1(j) \tag{8}$$

$$E[X_j^2] = 1 \cdot \text{Prob}[X_j^2 = 1] + 0 \cdot \text{Prob}[X_j^2 = 0]$$

$$= \text{Prob}[X_j = 1] = 1 - p_1(j) \tag{9}$$

$$E[X_j X_k] = 1 \cdot \text{Prob}[X_j X_k = 1] + 0 \cdot \text{Prob}[X_j X_k = 0]$$

$$= 1 - (\text{Prob}[X_j = 0] + \text{Prob}[X_k = 0]$$

$$- \text{Prob}[X_j = X_k = 0]) = 1 - (p_1(j) + p_1(k) - p_2(j,k)) \tag{10}$$

Therefore we find:

$$\mu_Y = E[\sum_{n=1}^{N} X_n] = \sum_{n=1}^{N} E[X_n] = \sum_{n=1}^{N} (1 - p_1(n)) \tag{11}$$

$$\sigma_Y^2 = E[Y^2] - E[Y]^2 = E\left[\left(\sum_{n=1}^{N} X_n\right)^2\right] - \mu_Y^2$$

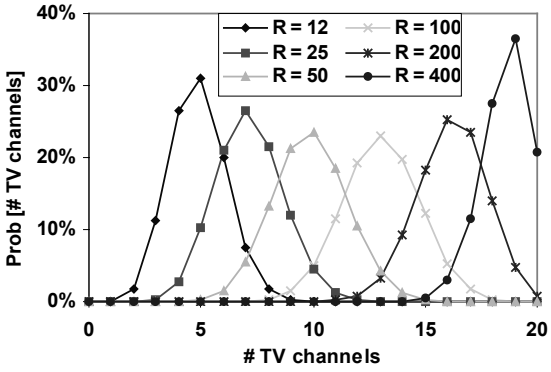
$$= E\left[\sum_{n=1}^{N} X_n^2 + 2 \sum_{1 \le j < k \le N} X_j X_k\right] - \mu_Y^2$$

$$= \sum_{n=1}^{N} E[X_n^2] + 2 \sum_{1 \le j < k \le N} E[X_j X_k] - \mu_Y^2$$

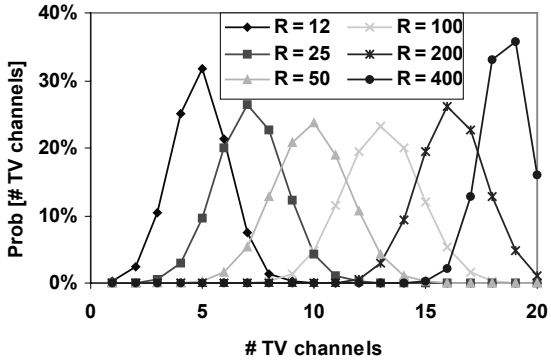
$$= \mu_Y - \mu_Y^2 + 2 \sum_{1 \le j < k \le N} \left(1 - p_1(j) - p_1(k) + p_2(j,k)\right) \tag{12}$$

6.2.2.c Example

Figure 6.3 shows that the Normal distribution is indeed a good approximation of the distribution of *Y*. The Normal distribution (Figure 6.3b) is very similar to the results of the exact solution, calculated through a brute force computation (Figure 6.3a). Since the latter method is computationally heavy, the results are presented for a total of only 20 TV channels ($N = 20$). Figure 6.3c compares the curves from the exact solution (full line) to the approximated solution (dotted line) and shows a good match (up to 1%).



(a)



(b)

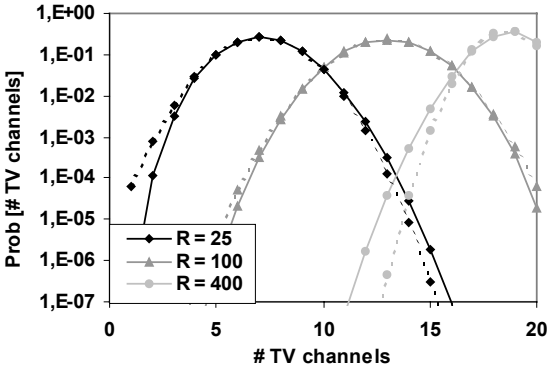


*Figure 6.3: Statistical distribution of the number of watched TV channels (*N = *20), for different values of the total number of user requests* R*; (a) exact, (b) approximated by Normal distribution, (c) comparison*

6.3 Network design

This section describes the methodology of our HFC access network design tool for TV services. The main goal is to find the minimal installation cost required to serve all requests, taking a possible restriction on the available RF spectrum at the node level into account. Through an exhaustive method, the optimal choice for each TV channel is made: send it through standard broadcast or through switched broadcast.

6.3.1 Input parameters

The most important input parameters are listed below. For each parameter a typical value is indicated in brackets, used for the standard configuration to which all other simulations are compared.

For each HE, we need the number of nodes (50), the number of users per node (1000), the digital TV market penetration (25%) and the percentage of simultaneous users during peak hour (40%), which gives us the maximum number of simultaneous requests per node R (100). Interesting parameters related to the content are the number of TV channels N (200), the Zipf parameter β (1.7), the TV channel stream bandwidth (SDTV mpeg2, 3.8 Mbps). We also need the RF channel bandwidth (in Europe 8 MHz, 64 QAM, 38 Mbps), the maximum number of available RF channels at the nodes (10) and the cost of one RF output port at the QAM RF devices (1 unit).

Since we use a Normal approximation for the number of simultaneously watched TV programs Y (average μ, variance σ^2) per node, we know that the following standard formulas for the Normal distribution are valid:

$$\begin{aligned}
&\text{Prob}[Y \le \mu] = 0.5 \\
&\text{Prob}[Y \le \mu + 1.28\sigma] = 0.9 \\
&\text{Prob}[Y \le \mu + 1.65\sigma] = 0.95 \\
&\text{Prob}[Y \le \mu + 2.33\sigma] = 0.99 \\
&\text{Prob}[Y \le \mu + 3.09\sigma] = 0.999
\end{aligned} \qquad (13)$$

This percentage of statistical events the system should be capable of serving, can be modified as an input parameter as well. A value of e.g. 99% (as in the standard configuration) means that of all 100 random events (peak hours), on average 99 can be handled by the system, as designed by the tool.

6.3.2 Methodology

The optimal solution is found through an exhaustive method that determines the optimal number of standard broadcast TV channels. First the TV channels are ranked according to popularity, then a variable number of broadcast TV channels is set ($0 < n < N$) and the corresponding design is calculated using the mathematical formulation presented above. As a result, the number of outgoing RF channels (unicast and broadcast) on the QAM RF devices at the HE is found, as well as the corresponding installation cost.

The value for n which gives the minimal installation cost, while satisfying the restriction on the number of available RF channels at the nodes, is used for the final design.

Note that if the n most popular TV channels are sent through standard broadcast, we have to take the following equations into account ($n > 0$):

$$p_1(j) = 0, \qquad \text{if } j \leq n \tag{14}$$

$$p_2(j,k) = 0, \qquad \text{if } j \leq n \text{ or } k \leq n \tag{15}$$

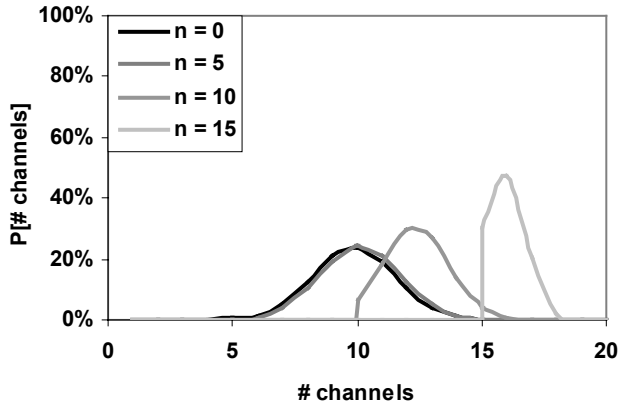


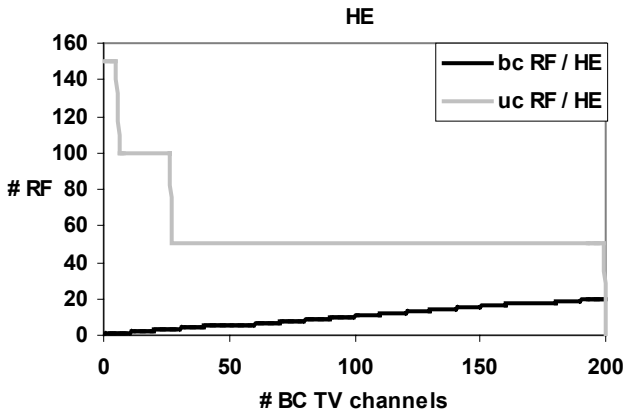
Figure 6.4: Distribution of the total number of TV channels streamed to a node, for different numbers of broadcast TV channels; N = *20,* R = *50,* β = *1.7*

Figure 6.4 shows the influence of the number of broadcast TV channels (n) on the total number of streamed TV channels (through both standard and switched broadcast), for a given number of channels (N = 20), user requests (R = 50) and content popularity (β = 1.7). On average 10 different TV channels (in 99% of the

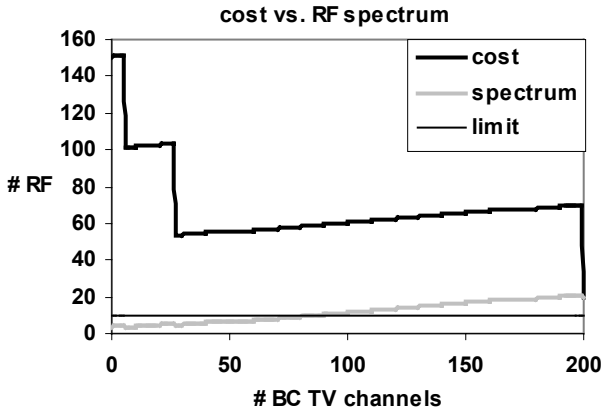
cases below 13 channels) are requested by the users through switched broadcast (n = 0). Offering more TV channels through standard broadcast obviously decreases the channels sent through switched broadcast, but the total number of TV channels increases. The variability on this total number decreases (to zero for $n = N = 20$). Note that the curves shown in Figure 6.4 are approximations based on the Normal distribution with the same values for the average and variation as the exact solution.

6.3.3 Results

Figure 6.5 presents the results for the standard configuration (Figure 1), with the input parameters given above. Figure 6.5a shows the number of outgoing switched broadcast and standard broadcast RF channels needed at the HE, for a variable n. The number of standard broadcast RF channels increases by one every 10 TV channels, the number of switched broadcast RF channels decreases from 3 to 0 per node (150 to 0 per HE). Figure 6.5b shows the total cost (# switched broadcast RF + # broadcast RF at the HE) and the spectrum (# switched broadcast RF + # standard broadcast RF at each node). In this configuration, four local minima for the installation cost can be distinguished, for n = 0 (cost = 150u), n = 6 (cost = 101u), n = 27 (cost = 53u) and n = 200 (cost = 20u). Since the maximum number of RF channels at each node is limited to 10, the overall optimum for n is 27. The corresponding average number of streamed TV channels per node is 32 (27 standard broadcast, 5 switched broadcast), with a 99% limit of 37 (27 standard broadcast, 10 switched broadcast).



(a)



(b)

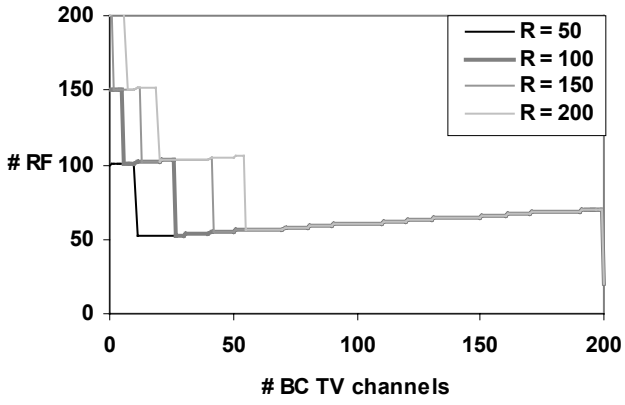
Figure 6.5: Results for the standard configuration, showing (a) the switched broadcast (uc) and standard broadcast (bc) RF channels at the HE and (b) the total installation cost at the HE and the occupied RF spectrum at the node

6.4 Numerical parameter study

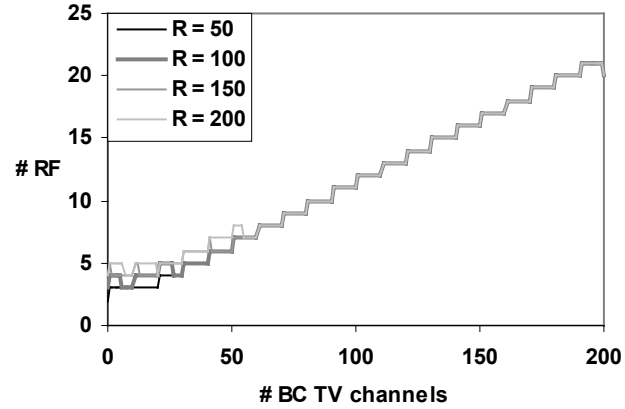
In this section, we compare the results for the standard configuration (bold curves in the figures below) to those for other configurations, with different values for one parameter at a time. The influence of the restriction on the maximum number of RF channels per node determines which local optimum has to be chosen.

6.4.1 Influence of the user demand

The influence of changes in the user demand (R simultaneous requests per node) on the total installation cost is shown in Figure 6.6a. The number of local minima for the installation cost increases from 3 (R = 50) to 5 (R = 200), but the deviation in cost at each minimum is rather small. This is because of the fact that the main cost, covered by the number of switched broadcast RF channels, remains the same (e.g. 50 · 1 = 50u at the optimum). Only the number of standard broadcast RF channels changes, but since these are sent through the splitter (see Figure 6.1), the influence on the total installation cost is much smaller. The influence of the number of standard broadcast RF channels is also visible in the RF spectrum at the nodes (Figure 6.6b), but only for lower values.



(a)

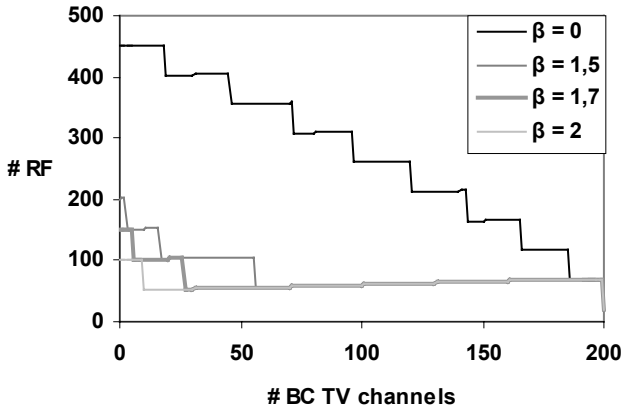


(b)

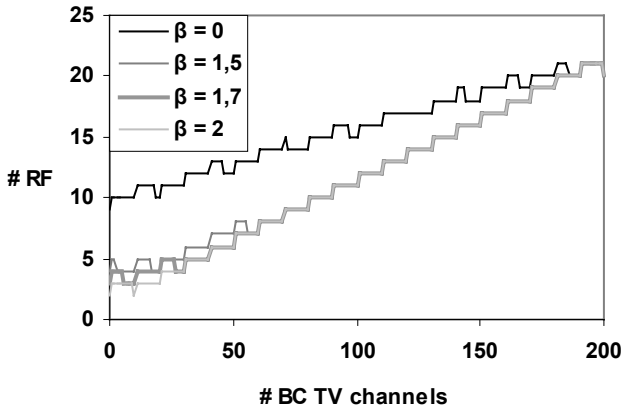
*Figure 6.6: Influence of the user demand per node (*R*) on (a) the total installation cost at the HE and (b) the RF spectrum at the node*

6.4.2 Influence of the content popularity

Changing the popularity distribution of the TV channels also has an impact on the installation cost.



(a)



(b)

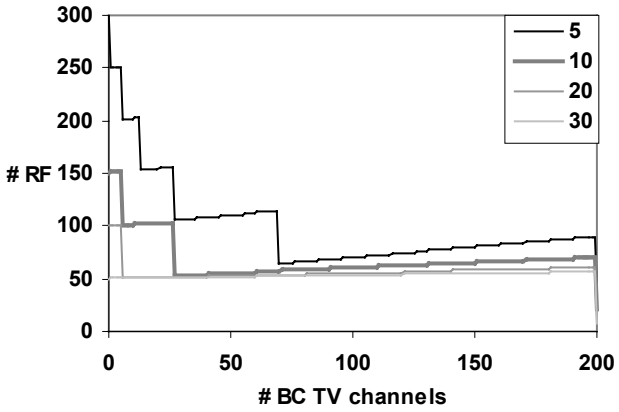
Figure 6.7: Influence of the Zipf parameter β for the content popularity on (a) the total installation cost at the HE and (b) the RF spectrum at the node

The higher the popularity of the top TV channels (high Zipf parameter β), the lower the cost, since all requests for these channels can be served through the "cheaper" standard broadcast service (if the necessary standard broadcast RF channels are present). Figure 6.7 shows the results for small variations around β = 1.7. The most popular channel then receives about 40% (β = 1.5), 50% (β = 1.7) or 60% (β = 2.0) of all requests. 90% of all requests are made for the top 27 (β = 1.5), 13 (β = 1.7) or 6 (β = 2.0) most popular programs. When β = 0 (each TV channel is equally popular), the total number of RF channels is always

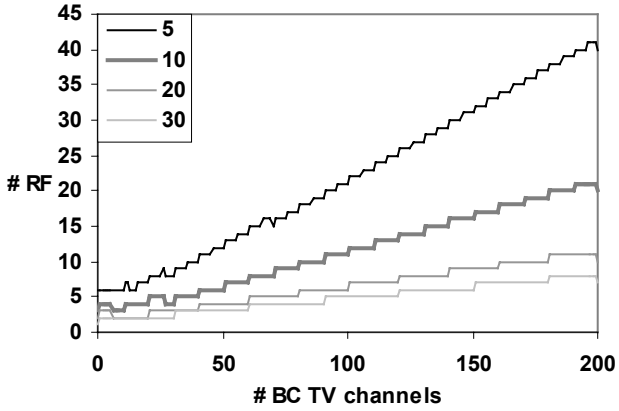
extremely high (at least 9, when $n = 0$), which increases the total installation cost from 53 to 402 units (at a maximum of 10 RF channels per node).

6.4.3 Influence of the stream bandwidth

Changing the quality or format of the streams, e.g. from SDTV mepg2 format (3.8 Mbps) to SDTV mpeg4 (1.6 Mbps) or HDTV mpeg4 (8.0 Mbps), also has a large impact on the installation cost.



(a)



(b)

Figure 6.8: Influence of the number of streams per RF channel on (a) the total installation cost at the HE and (b) the RF spectrum at the node

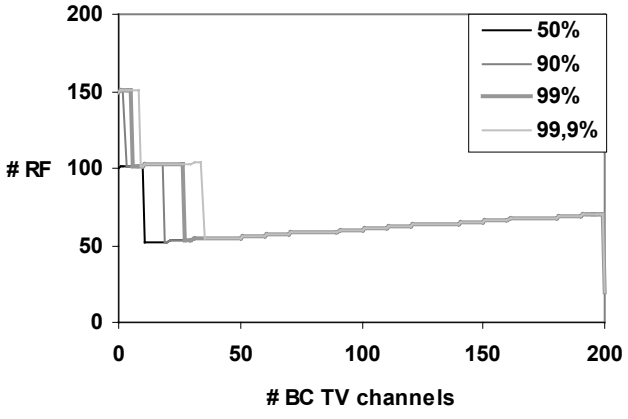
Note that the impact of doubling the stream bandwidth is much higher than doubling the user demand, since the latter does not mean that twice as many TV channels will be watched (e.g. contrary to a VoD scenario)! Figure 6.8 shows the results for different numbers of streams that can be transported simultaneously in one RF channel. The influence on the RF spectrum is now much more significant. Both the number of local minima and the total installation cost increase quickly. When 30 streams can be transported in one RF channel (e.g. 30 SDTV mpeg4 streams in one 256 QAM 8 MHz RF channel) one unicast RF channel per node is enough to serve all users (no standard broadcast TV).

6.4.4 Influence of the size of the uncertainty interval

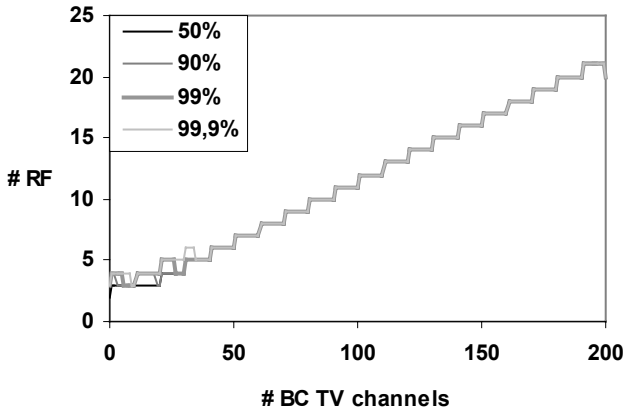
The influence of the size of the uncertainty interval (99% in the standard configuration) is shown in Figure 6.9. The results are similar to those for changes in user demand, but less noticeable.

6.4.5 Conclusion

The numerical results above show the influence of the most important parameters. The main conclusion is that the cheapest solution would normally be to stream as much TV channels as possible through standard broadcast. The restriction on the total number of RF channels at the node determines whether this solution can be reached or not. If not, one of the local minima has to be chosen. Therefore the number of unicast RF channels must be increased (typically from 1 to 2 or 3) and the total installation cost will increase almost proportionally.



(a)



(b)

Figure 6.9: Influence of the size of the uncertainty interval on (a) the total installation cost at the HE and (b) the RF spectrum at the node

6.5 Conclusions

In this chapter, an HFC access network design tool for standard and switched broadcast TV services has been presented. While very popular TV channels are offered using standard broadcast mechanisms, sending less popular channels through switched broadcast technologies reduces the network load considerably and decreases the installation cost. We identified the most important traffic and content parameters and studied their influence on different network configurations.

References

[1] N. Sinha, R. Oz, "The Statistics of Switched Broadcast", Proc. of SCTE 2005 Conference on Emerging Technologies, Huntington Beach, CA, Jan 2005

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", Proc. of IEEE Infocom, March 1999, pp. 126-134

7 Conclusions

As the popularity of bandwidth intensive streaming content has increased significantly during the last few years, advanced content distribution network architectures have been proposed in the recent past, to replace traditional client-server systems. In this work, distinct network solutions were presented for various next-generation multimedia streaming services. By replicating the content to surrogate servers or proxy caches closer to the end user, the quality of service can be significantly increased. The clients experience reduced latency and jitter, while the backbone network and origin server are relieved of most traffic. As a consequence, a more robust and scalable streaming service can be offered.

For each of the proposed solutions the network design and content placement problem was tackled, by optimizing the cost trade-off between network bandwidth and storage. The network design depends on the underlying network technologies and elements used. Analytical and ILP solutions of the specific design problems were formulated and heuristics were applied to provide a scalable solution for larger networks. Once the network design was handled, replica placement algorithms were proposed to determine the optimal location of the available content in the network. To provide a scalable and yet close to optimal solution, distributed RPAs were worked out and evaluated through simulations on various network topologies. These RPAs decide which content to

store based on local information such as user demand patterns and network load. When peer-to-peer communication between these storage entities is possible, local information can be exchanged to further optimize the content placement. Besides the traditional trade-off between transport costs and storage costs, a method to balance the load on the network has been added to the proposed solutions. We found that load balancing can be performed at the price of only a slightly increased average network bandwidth. As a consequence, more user requests can be served by the network.

The main services studied in this work were Video on Demand (VoD), broadcast or time-shifted television and multimedia content production and storage, the digital equivalents of traditional services that did not offer user interactivity. The proposed solutions for the network design for VoD reduce the load on the core network and the central server considerably, mainly limiting the traffic to the access networks. The network load for IPTV services is reduced as well, by introducing switched broadcast techniques in the access network, to limit the required bandwidth spectrum. Interactivity for IPTV can be supported through a time-shifted TV service, where proxy caches store sliding windows of recently broadcasted TV programs, to allow VCR-like commands such as pause and rewind. We found that using small, co-operating diskless caches at the proxies can offload the regional servers almost entirely. Collaborative multimedia production companies can benefit from Grid technologies to effectively share resource and media repositories. Bandwidth management solutions were presented, effectively balancing the server load.

As a general conclusion, we can observe that as multimedia services become more popular on the Internet, content distribution networks store their content closer and closer to the end users. Since plain replication of the origin server to the edge of the backbone network is a very expensive solution in terms of storage cost, intelligent content replication algorithms with peer-to-peer co-operation have to be brought into play. Eventually, small caches and streamers are being deployed in the access networks, storing only partial content in dynamically updated sliding intervals.

DSL based access network architectures are therefore evolving more and more towards fully IP-aware networks, thus facilitating the introduction of next-generation service enablers and application based QoS for future services, beyond basic triple-play. The current trend in broadband cable access networks is to converge towards one technological platform for internet and data services, such as the (Euro-)DOCSIS standard for cable networks, and one for digital television services, such as the European DVB platform. The main reasons for this phenomenon are on one hand the integration of different interactive

broadband services at the end user devices and on the other hand the ability for the service provider to enable dynamic bandwidth sharing among these services and reduce the dominant deployment costs from service specific edge devices (both CapEx and OpEx).

As research on content distribution architectures is still ongoing, future studies may focus on extended dimensioning and placement algorithms, including other cost metrics such as delay or networking techniques such as multicasting. The investigation of the influence of different service requirements, content characteristics or network parameters on the network optimization is a major research topic as well. Interesting new services are emerging in the context of media grids, mobile content delivery networks, IPTV networks and user-centric service networks such as storage networks for personal content. Peer-to-peer content distribution between end devices remains an interesting research topic as well.

A comparison of peer-to-peer architectures

P. Backx, T. Wauters, B. Dhoedt, P. Demeester

Ghent University (INTEC) – IBBT – IMEC

Gaston Crommenlaan 8, bus 201

B-9050 Gent

Conference proceedings of Eurescom 2002, Powerful Networks for Profitable Services, Heidelberg, Germany, October 21-24, 2002, pp. 215-222.

Abstract

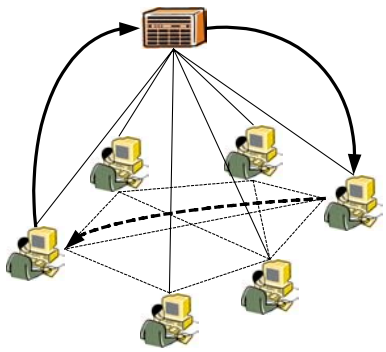
The large number of peer-to-peer file-sharing applications can be subdivided in three basic categories: having a mediated, pure or hybrid architecture. This paper details each of these and indicates their respective strengths and weaknesses. In addition to this theoretical study, a number of practical experiments were conducted, with special attention for three popular applications, representative of each of the three architectures. Although a number of measurement studies have been done in the past ([1], [3], etc.) these all investigate only a fraction of the

available applications and architectures, with very little focus on the bigger picture and to the recent evolutions in peer-to-peer architectures.

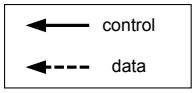
A.1 Introduction

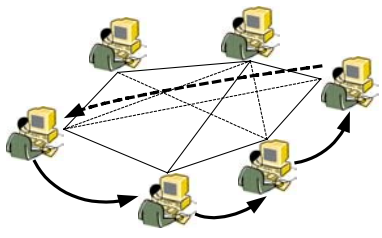
The rationale behind our peer-to-peer measurement study, reported on in this paper, is to evaluate the suitability of peer-to-peer networks for heavy traffic content distribution networks (CDNs). These networks distribute the data traffic over the entire network avoiding traffic concentration around servers.

Peer-to-peer networks are extremely popular on the Internet. Especially the file-sharing applications have a large user base. There is also a thriving community proposing (e.g. [4]) and implementing [6] new features and entirely new architectures. Generally speaking, a peer-to-peer file-sharing platform can be categorised into one of three classes of architectures: mediated, pure peer-to-peer and hybrid. These will be discussed into more detail in the next section.

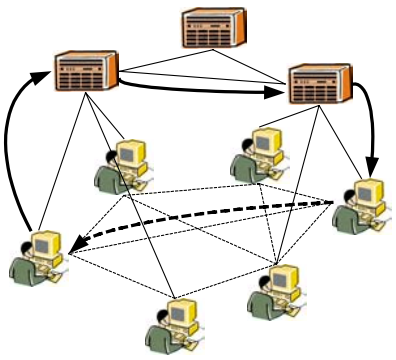


(a)





(b)



(c)

Figure A.1: Three peer-to-peer architectures: (a) mediated, (b) pure and (c) hybrid

For each of these architectures a representative was chosen. The applications were installed and run in the ATLANTIS test lab. The connections made to our peer were monitored and logged. Based on this information, a comparison was made between the three architectures.

A.2 Architectures

All peer-to-peer architectures have one thing in common: the actual data transfer is always peer-to-peer: a direct data connection is made between the peer offering the file and the requestor. The control plane however is implemented in various ways. Figure A.1 gives an overview of the three types we identified and a typical search-download sequence in which the leftmost peer searches a file and downloads it from the rightmost peer. Every individual peer-to-peer application uses one of these architectures, with its own specific quirks.

Mediated architecture

A mediated architecture uses a client-server setup for its control operations. All peers log on to a central server that manages the file and user databases. Searches for a file are sent to the server and, if found, the file can be downloaded directly from a peer. In most cases the server will have a database of files shared by peers.

However with the number of court cases against server-based peer-to-peer applications developers are hesitant to use this architecture. Soulseek [15] for instance only uses the server to log onto the network. Afterwards the server functions as a proxy that distributes the searches towards the peers. Thus every

peer searches the search string in its own local database, while the server merely distributes search strings over the network.

Pure peer-to-peer architecture

Pure peer-to-peer applications will not use a central server at all (except possibly for logging onto the network). Queries for files can be flooded through the network or more intelligent mechanisms can be used [4,7]. Pure peer-to-peer networks have become quite unpopular because they generate a lot of overhead traffic to keep the network up and running. FreeNet [5] still uses this model because it offers an unprecedented anonymity, not found in any other architecture. Furthermore, to ensure anonymity FreeNet does not send data directly from the source to the requester, but routes it over the pure peer-to-peer overlay network.

Hybrid architectures

Hybrid architectures are the latest development in the peer-to-peer community [8]. Their goal is to offer the best of both worlds. Through the introduction of so-called ultrapeers, hybrid architectures have properties of both the mediated and the pure architectures. The ultrapeer will perform the task of a server in the mediated architecture, but for only a subset of the peers. The ultrapeers themselves are connected through a pure peer-to-peer network. Thus hybrid architectures introduce two layers in the control plane: one of "normal" peers connecting to ultrapeers in a client-server fashion and one of ultrapeers connected with each other via a pure peer-to-peer network.

Both pure and hybrid architectures build an overlay network over the existing IP network. In most cases this overlay is constructed arbitrarily, however it has been shown [3] that this generates a lot of expensive inter-domain traffic that can be reduced by intelligently building the overlay.

A.3 Measurements

A number of general measurements were conducted on a dozen of peer-to-peer applications, but for a more detailed study of the peer-to-peer application’s network usage we choose a representative for each of the three architectures described above. An important factor was the popularity of the programs, which can be checked through the top lists on download sites [9].

Table A.1 gives a broad overview of some distinguishing features of several peer-to-peer file sharing applications. All recent peer-to-peer applications have very comparable features, though they might differ in the details.

	Type of files	Community	Resume	Multisource downloading / swarming	Data integrity	Anonymity
Audio-Galaxy [11]	MP3	Strong (1, 2, etc.)	Yes	No	Hash	No
Gnutella [12]	All	Weak (2)	Yes (5)	Some clients (Xolox)	Work in progress	No
FastTrack [13]	All	Weak (2, 3)	Yes	Yes	Hash	No
WinMX [14]	All	Weak (2, 3)	Yes	Yes	Hash	No
SoulSeek [15]	All	Strong (2, 3, 4)	Yes	No	None	No
FreeNet [5]	All	Application specific			Hash	Yes
eDonkey [16]	All	Weak	Yes	Yes	Hash	No

(1) User groups.
(2) Can see individual user's files.
(3) One to one messaging.
(4) Group chat.
(5) On some clients this is not done automatically.

Table A.1: Feature comparison

AudioGalaxy is the only one to focus solely on MP3 music files. Most other applications can be used to share any type of file, but usually do have special support for MP3 files through extended meta-tags. A number of applications try to build a strong community feel by offering message boards, private chat between two users (also known as instant messaging), browsing of a users shared files, etc. In turn users of those applications are usually more willing to share more files for a longer time than users of applications without as many community "features".

The next 3 features mentioned in the table are important in order to successfully transfer a file on a volatile peer-to-peer network. Resuming interrupted downloads is a standard feature on any peer-to-peer applications. Most applications will automatically detect when they can resume a broken download, a few will have to be instructed manually to do so. Multisource downloading and swarming have become increasingly important and successful for large files. Multisource downloading allows a peer to download one file simultaneously from several other peers, thus increasing the overall download speed. Swarming expands on this and allows the sharing and sending of partial files, speeding up the distribution process when only few peers have a popular file. Data integrity allows a user to quickly check whether a downloaded file has errors. When swarming is used a tree-hash function is necessary to check the individual parts.

Only FreeNet provides anonymity to its users. It is trivial in any of the other applications to find out a user's IP number and in most cases it is also possible to browse his shared files (this can be both a good and bad feature, AudioGalaxy is the only application that allows a user to turn this on or off).

Next we take a look at the three representatives we choose to test in our measurement study.

AudioGalaxy

AudioGalaxy is the application that represents the mediated architecture. Although its popularity has dropped slightly since early 2002, it remained the top choice for downloading music files until it shut down in June 2002. Monitoring this application proved to be easy because it maintains a detailed log-file.

Recently Audiogalaxy was sued by RIAA (Recording Industry Association of America) and has entirely shut down its file sharing service in an out-of-court settlement. Because the AudioGalaxy network is server-based it was also one of the easiest networks to target and sue. Consequently, only very few server-based architectures remain and none are anywhere near as popular as Napster (the original mediated peer-to-peer application) or AudioGalaxy were. While a mediated architecture might have performance advantages, very few peer-to-peer

developers dare to risk the expensive court suits and settlements that Napster and Audiogalaxy have faced. Pure and hybrid architectures offer a more extensive insurance against such cases, since file databases remain on the users pc, either on every client, like old-style Gnutella or on the ultrapeers (almost always run by end-users) and not on the developers servers.

Gnutella

The Gnutella network is the only popular pure peer-to-peer network. Its popularity is due to the freely available protocol definition and the wide range of available peer applications for many operating systems, catering to everyone's taste. LimeWire and BearShare are the most advanced and stable applications, and therefor also the most popular. With the introduction of ultrapeers, Gnutella can no longer be considered a pure peer-to-peer network, however at the time of our first batch of measurements there were very few ultrapeers deployed.

Gnutella was monitored through the use of a modified Gnucleus peer. Gnucleus is the most popular open source Gnutella peer and was easy to enhance with measurement code.

FastTrack

KaZaA and Morpheus have dominated the top downloads lists for most of 2001 and are still increasing in popularity in 2002, in spite of all the lawsuits against them. The FastTrack peer-to-peer stack, on which these two applications are based, was thus the logical choice as a hybrid architecture representative.

Because both programs are closed-source it was not easy to monitor them. We settled on using ntop [17], a general network monitoring and diagnostic program.

Recently Morpheus abandoned the FastTrack peer-to-peer architecture after a dispute with FastTrack. The new Morpheus 1.9 Preview Edition that was used in some of the later experiments uses Gnutella as underlying network. This new Morpheus peer relies heavily on the Gnucleus (mentioned above) code base and is notoriously unstable.

Set-up

We used a relatively basic set-up, which is shown in Figure A.2 (for the AudioGalaxy case). Most (if not all) peer-to-peer programs do not work behind a firewall, NAT or proxy. Some do work, although with limited functionality. It is for instance never possible to set up a transmission between two peers behind a firewall or NAT, because none of the peers can listen to and accept incoming connections. There have been some trials with connection handover, however apparently they were not very successful. The idea is that both firewalled peers

set up a connection to a server, after which the server hands over the connections to the peers.

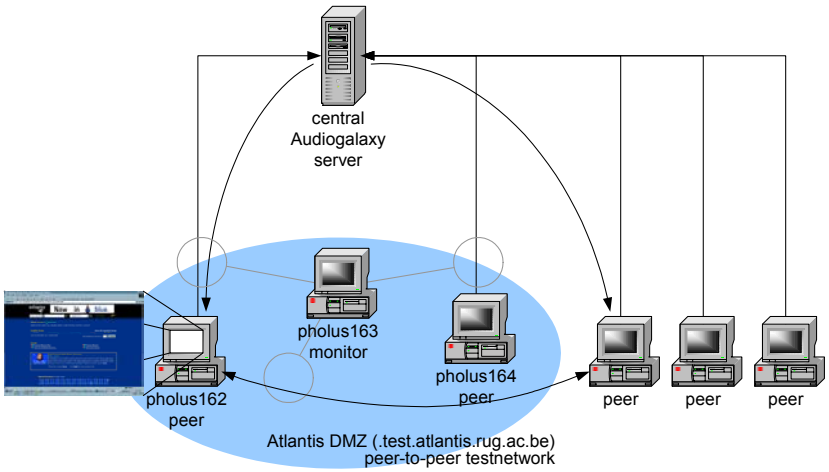


Figure A.2: Test set-up

To avoid these issues a demilitarized zone (DMZ) was created that is outside the firewall and gives the peers full access to the Internet. One or two machines are running the peering application (in this case the AudioGalaxy satellite). These both have an AMD K6 cpu running at 550 MHz and 256 MB of memory. A Pentium II based machine at 350 MHz monitors the generated traffic between our peers, the server and other peers. All machines are connected to the Internet through a 100 Mbit connection.

A.4 Experimental results

This section describes the results we obtained from monitoring the peer-to-peer applications. We could not always perform all experiments with all application. For both AudioGalaxy and FastTrack no information on the protocol was available. While AudioGalaxy maintains a rather thorough logfile, the FastTrack-based applications (in casu KaZaA) did not.

Overhead

Based on the measurements the three applications are compared. The overhead (both in network and CPU resources) involved with connecting to a certain network, staying on that network, finding and downloading files is studied.

architecture	network	#connections	network traffic	average % CPU usage	% overhead on file transfers
Mediated	AudioGalaxy	1	0.4 kB/min	1	2.2
Pure	Gnutella	5+	600+ kB/min	8	3
Hybrid	Gnutella (with ultrapeers)	1	2 kB/min	1	3
	FastTrack	1	0.1 kB/min	1	3
	eDonkey	1	0.2 kB/min	0	2.5
	OpenNap	1+	0.1 kB/min	5	2.5

Table A.2: Peer-to-peer architecture comparison

Table A.2 gives a summary of results for the various peer-to-peer applications we studied in this first experiment:

- The "#connections" column shows the number of connections that peers make to stay connected to the peer-to-peer network. A mediated architecture (like AudioGalaxy) of course needs only the connection to the central server. A peer in a hybrid architecture is usually only connected to one ultrapeer, however in OpenNap's case it is possible to connect to multiple ultrapeers at once. A Gnutella peer connects to a number of other peers. Usually one needs at least 5 connections in order to have a sufficiently large pool of reachable peers.
- The network traffic column is the amount of traffic a normal peer generates when connected to the network, but not downloading or searching. All architectures have minimal traffic, except Gnutella. This is because peers in the Gnutella network are actually routing messages for other peers, a task that is done by the server or ultrapeers in the other networks. The 600 kB/min or 10kB/s is no mistake. Clearly the Gnutella network without ultrapeers is not suited for modem users. With the introduction of ultrapeers

the network traffic to a leaf node is severely reduced, but is still somewhat higher then in most other architectures.

- The two other columns show the average percentage of CPU usage. This is related to the number of network connections a peer has to maintain and the amount of traffic on these connections.
- The last column shows the amount of overhead related to sending files. Gnutella and FastTrack use the HTTP protocol, while the other networks use a proprietary protocol on top of TCP. Since AudioGalaxy supports no error-correction nor swarming and only a very limited resume function the transfer protocol is very basic and has the least overhead.

Hopcount

Figure A.3 summarizes the hopcount between our peer and peers connecting to it. Although the three curves have similarities there is also a clear distinction between them. AudioGalaxy peers seem to be, on average, closer then peers in other networks. This indicates that AudioGalaxy does a better job at finding a close download location (and presumably a faster connection) than the other networks. Gnutella does not automatically optimise connections at all, which results in a higher hopcount.

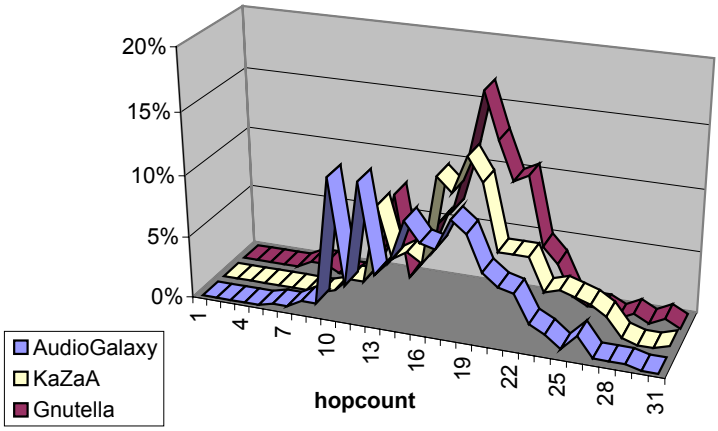


Figure A.3: Hopcount distribution for connecting peers

Speed versus hopcount

Figure A.4 shows the connection speed between our peer and another one in comparison to the number of hops between the two. Generally the speed decreases with increasing hopcount, which is what is expected. On the Gnutella network, the connection speeds are usually higher. This is most likely because there are very few modem users on the Gnutella network, due to the high network overhead (as seen in 0).

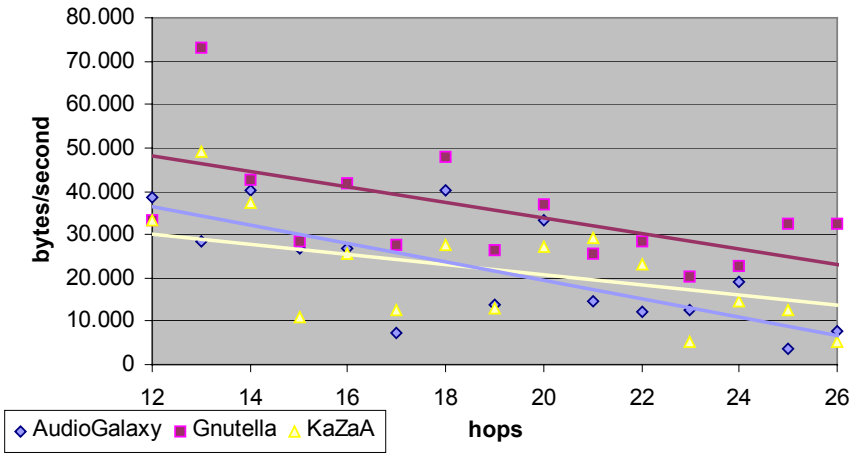


Figure A.4: Speed versus hopcount

Popularity distribution

The next measurement deals with the user behaviour. Popularity distributions for web servers exhibit a zipf-like distribution [10]. In a zipf-like distribution the relative probability of a request for the *i*'th most popular document is proportional to $1/i^{\alpha}$, with α typically less than 1. Monitoring the actual files that were downloaded proved to be very difficult if not impossible for KaZaA, so we can only present results for Gnutella and AudioGalaxy.

Figure A.5 shows the popularity of the files on our peer. The left graph is for AudioGalaxy, the right one for Gnutella. To the left of each figure are the most popular files, to the right the least popular. The Y-axis shows the number of times that particular file has been downloaded. Also shown is a least square fit with a zipf-like distribution. For AudioGalaxy the α value is 0.416, while for Gnutella it is 0.745. The Gnutella value is in line with what has been seen in web

traffic, however the AudioGalaxy value is significantly off (usual α values are between 0.6 and 0.8). Either AudioGalaxy's behaviour is different, but it is quite possible that the number of measurements was too low. The most popular file was only downloaded 10 times, which is very little compared to the most popular file on our Gnutella peer that was downloaded 263 times.

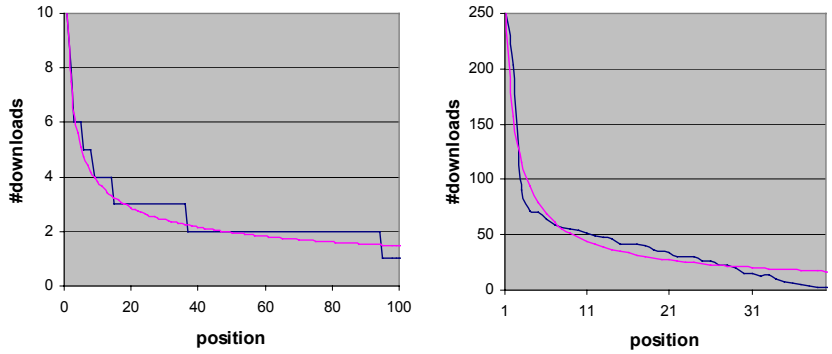


Figure A.5: Popularity distribution for AudioGalaxy (left) and Gnutella (right)

The fact that the Gnutella requests are comparable to web traffic indicates that it might be beneficial to apply (web) caching techniques to a peer-to-peer network.

Transmission errors

A next measurement logs the error codes returned by transmissions from our peer to others. This too was only possible for AudioGalaxy and Gnutella. Care should be taken when interpreting Figure A.6. The AudioGalaxy status codes are on a per-connection basis, while the Gnutella codes are on a per-transmission basis. A "completed" designation in Gnutella could mean that there were several connections during the transmission that were interrupted.

Clearly the number of errors is large compared to the successful transfers, which hints at the volatile nature of a pure-to-pure network and the need for error detection and recovery.

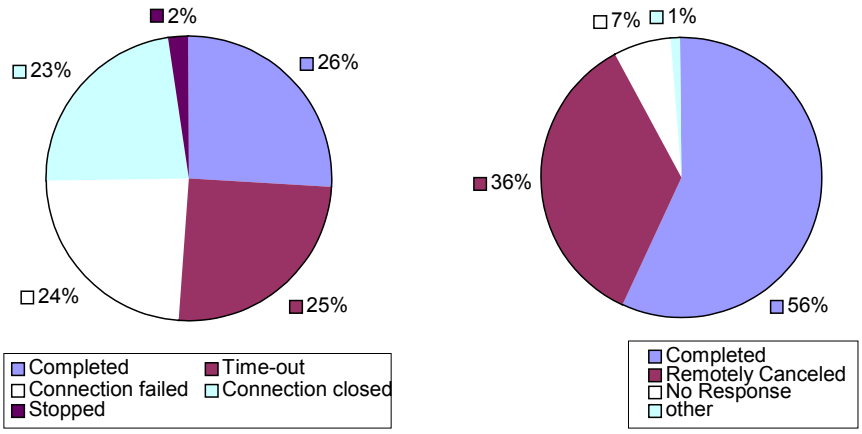


Figure A.6: Errors on outgoing transmissions (left: AudioGalaxy, right: Gnutella)

A.5 Conclusion

The peer-to-peer community is evolving towards hybrid networks. This can be seen in the architectures of new peer-to-peer applications such as eDonkey and the FastTrack peer-to-peer stack and in the introduction of ultrapeers into the Gnutella network. There was indeed a clear need for Gnutella to evolve [2]. In spite of the popularity of hybrid architectures, there still are advantages of using a mediated architecture. A global central server can optimise the connections much better then an ultrapeer with only local scope. However, scalability and legal issues might plague a centralized architecture. One more advantage of hybrid architectures, which was not mentioned before, is the business case. Since in most cases home users are responsible for running the ultrapeers there is no need to invest in expensive server farms.

References

[1] S. Saroiu, P.K. Gummadi, S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", Technical Report UW-CSE-01-06-02 University of Washington, Seattle, WA, USA, July 2001.

[2] "Gnutella: To the Bandwidth Barrier and Beyond", Clip2 report, November 6, 2000, available at http://www.clip2.com/gnutella.html.

[3] M. Ripeanu, A. Iamnitchi, I. Foster, "Mapping the Gnutella Network", IEEE Internet Computing, January-February 2002.

[4] K. Aberer, M. Punceva, M. Hauswirth, R. Schmidt, "Improving Data Access in P2P Systems", IEEE Internet Computing, January-February

2002.

[5] I. Clarke, O. Sandberg, B. Wiley, T.W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System", Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, ed. by H. Federrath. Springer: New York (2001).

[6] A. Singla, C. Rohrs, "Ultrapeers: Another Step Towards Gnutella Scalability", working draft, available from the Gnutella Developers Forum at http://groups.yahoo.com/group/the_gdf/.

[7] K. Aberer, "P-Grid: A self-organizing access structure for P2P information systems", Technical Report TR2001-016, Ecole Polytechnique Fédérale de Lausanne.

[8] B. Yang, H. Garcia-Moline, "Designing a Super-Peer Network", Technical Report, Standford University, February 2002.

[9] CNET download.com, http://download.cnet.com/.

[10] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", IEEE Infocom 1999.

[11] AudioGalaxy, http://www.audiogalaxy.com/.

[12] T. Klingberg, R. Manfredi, "Gnutella 0.6", draft protocol definition, available from http://rfc-gnutella.sourceforge.net/.

[13] FastTrack, http://www.fasttrack.nu/.

[14] WinMX, http://www.winmx.com/.

[15] Soulseek, http://www.slsk.org/.

[16] eDonkey 2000, http://www.edonkey2000.com/.

[17] L. Deri, S. Suin, "Effective Traffic Measurement using ntop", IEEE Communications Magazine, May 2000.

B

IPTV deployment: trigger for advanced network services!

T. Wauters, K. Vlaeminck, W. Van de Meerssche, S. Van den Berghe, F. De Turck, B. Dhoedt, P. Demeester

Ghent University (INTEC) – IBBT – IMEC

Gaston Crommenlaan 8, bus 201

B-9050 Gent

E. Six, T. Van Caenegem

Alcatel Research & Innovation

F. Wellesplein 1

B-2018 Antwerpen

Accepted for publication in The Journal of the Communications Network, 3Q/06.

Key words

Service awareness, access network, IPTV.

Abstract

The increasing popularity of multimedia broadband applications, beyond basic triple-play, introduces new challenges in content distribution networks. These next-generation services are not only very bandwidth-intensive and sensitive to the high delays and poor loss properties of today's Internet, they also have to support interactivity from the end user. The current trend is therefore to introduce IP-aware network elements in the aggregation networks to meet the increasing QoS requirements, offering a smooth transition from legacy ATM-based platforms towards more scalable, efficient and intelligent access networks. One of the promising services triggering this evolution is IPTV. This paper presents a large-scale IPTV service deployment in an IP-aware multi-service access network, supporting Broadcast TV, Time-Shifted TV and Pay-per-View services. Transparent proxy caches collaborate providing distributed network storage and user interactivity, while offering an adequate end-to-end Quality of Experience. As a use case, a Time-Shifted TV solution is introduced in more detail. We discuss a distributed caching model that makes use of a sliding window concept and calculates the optimal trade-off between bandwidth usage efficiency and storage cost. A prototype implementation of a diskless proxy cache is evaluated through performance measurements.

B.1 Introduction

Although telecom operators continue to build out their broadband access networks to improve high-speed Internet access and voice-over-ip (VoIP) services, IPTV services are becoming the highest-priority residential telecom services, creating very promising market opportunities. These bandwidth-intensive IPTV services have a significant impact on the underlying transport network and require more intelligent access network elements to meet the higher QoS requirements. IPTV is therefore considered as an important driver for other advanced network services.

As a consequence, the architectural model of access networks has evolved towards multi-service and multi-provider networks during the last few years. Ethernet as well as full IP alternatives have been investigated as viable connectionless successors for the legacy ATM-based platforms. While the introduction of Ethernet up to the edge solves some of the existing access networks, new ones are created. Per subscriber traffic segregation and the lack of QoS support are the main issues of standard Ethernet. While the introduction of VLANs could alleviate these shortcomings, it can be questioned whether this approach is sufficiently scalable for larger access network deployments.

Therefore an IP-aware network model [3] is often considered a valuable alternative.

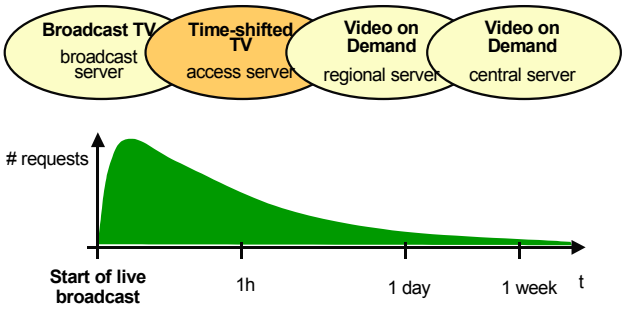


Figure B.1: Delivery mechanisms for IPTV

Depending on the popularity of the content, different IPTV services can be distinguished (Figure B.1). While traditional live TV is broadcasted from a central server deeper in the network, video-on-demand (VoD) servers are typically located at the edge of the core network. In order to support interactivity from the end-user for live TV or to serve requests for other very popular content, servers in the access network can become beneficial. This approach however has important implications for future access network architectures, as discussed further on in this paper.

B.2 Next-generation broadband services

Next to IPTV services, a wide variety of other value added (interactive) services, such as managed home networking, home automation and security management, multimedia multi-party conferencing and online gaming, can be offered by service providers, each setting its own requirements for the underlying network. Different services have highly fluctuating bandwidth requirements. Delay and jitter requirements also differ from service to service.

For interactive services a low delay over the network is a critical success factor. When several parties exchange information in an interactive way, the quality of the user experience (QoE) decreases with increasing delay. For instance, a telephone conversation will become very difficult if the network delay exceeds a few 100 milliseconds. Multimedia services are very sensitive to jitter – variation in the delay will drastically degrade audio and video quality – but in case they are non-interactive (e.g. Video on Demand), some delay can be tolerated.

Some services, such as firewalls and intrusion detection systems for managed home networking, interact directly with the network layer and could be deployed on a large scale inside the access network. Other services mainly focus on the application layer, but even these services could benefit greatly from enabling technologies in the access network: e.g. a caching system in the access multiplexer supporting multimedia content delivery.

However, several shortcomings of operational DSL access networks prevent further generalization of the Internet and the introduction of such new services.

B.3 Implications for the access network architecture

Network transformation

The connection-oriented approach of current DSL (Digital Subscriber Line) access networks (cf. Figure B.2 on current access network deployment) has been identified as a limiting factor, both in terms of access network scalability – all PPP (Point to Point Protocol) links are terminated in a single device, the broadband access server (BAS), PPP obstructs multicast support in the access network – and subscriber terminal autoconfiguration – PPP links cannot be autoconfigured as the link specification is location dependent. Also, since PPP access networks are tailored to the connection of a single device per subscriber, Network Address Translation (NAT) is required on subscriber lines where multiple IP devices are connected, breaking end-to-end IP connectivity. Furthermore, introducing new services, all imposing their own QoS (Quality of Service) requirements, is impossible over a single best-effort access link as it exists today.

To overcome all these issues, a converged IP access network architecture, as depicted in Figure B.3, was introduced in [1], showing how an IP(v6) data-plane can be used as the cornerstone of a future service-oriented access network:

- IP awareness close to the end user is required for the deployment of new advanced services in the access network.
- A converged access network reduces the capital and operational expenses (CAPEX and OPEX) of maintaining per service separated networks. Furthermore, interaction between different services is more flexible.
- Due to its connectionless nature, an IP access network allows for multiple edges, greatly improving scalability and robustness in case of edge node failure.

- In light of the growing peer-to-peer traffic volume, the ability to process local traffic without edge involvement further increases the scalability of an IP access network.

An overview of the most important elements in the network transformation process is given in Table B.1 [7].

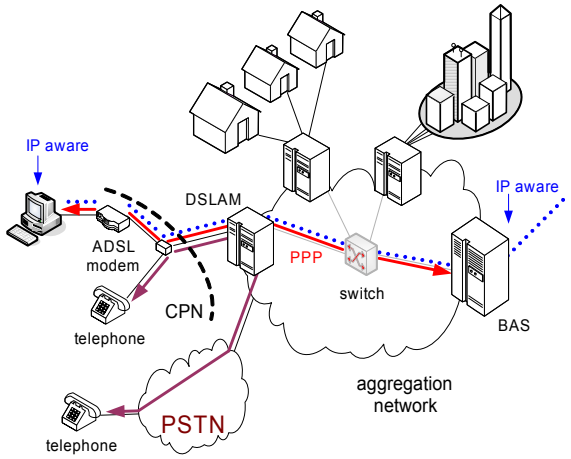


Figure B.2: Current Access Networks

For telephony, a switched path is set up over the PSTN network (Public Switched Telephone Network) between the end-devices of the caller and the called party. For DSL (Digital Subscriber Line) broadband Internet access, end-users set up PPP (Point-to-Point Protocol) connections inside ATM VCs (Asynchronous Transfer Mode / Virtual Circuits) from their customer premises network (CPN) to a central aggregation node, the Broadband Access Server (BAS). Only the tunnel endpoints are IP aware. When an end-user wants to connect multiple IP devices to the Internet, a NAT (Network Address Translation) router is used to terminate the PPP connection. Although geographically similar, PSTN and DSL networks are physically separate, except for the first mile. Broadcast TV is distributed over yet another access network (e.g., cable, satellite).

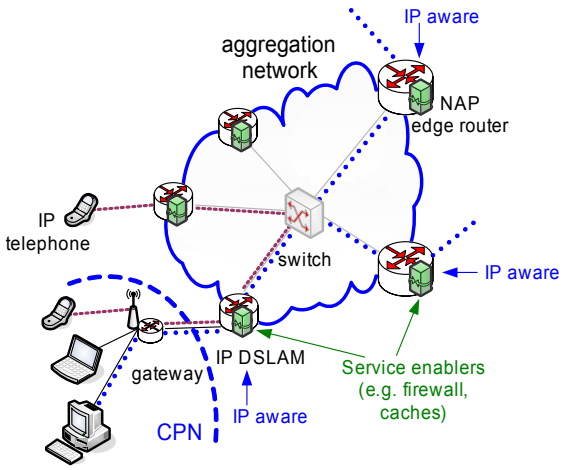


Figure B.3: Evolution towards a converged, IP aware, full service access network.

Current ATM-based broadband aggregation	Next-generation Ethernet/IP-based broadband aggregation
ATM DSLAMs	IP DSLAMs
• Unintelligent Layer 1 aggregation • Low-speed ATM uplinks • Mostly Central Office - based	• Intelligent aggregation with multicast support • Gigabit Ethernet Uplinks • Increasingly RT-based
Complex, fixed connections	Simple, flexible connections
• PPP-based • Bound to DSL CPE in the home • Provisioning cost high	• DHCP-based • Independent of device • User-based • Provisioning cost low
Centralized B-RAS	Distributed routers
• Optimized for best-effort internet • Lack of scalable routing and QoS • Typical OC-12 handoff to IP core	• Optimized for QoS-sensitive services • Highly scalable • 10 GbE handoff to IP/MPLS core
Lack of network resiliency	Highly available network
• Outages tolerated • Minimal financial repercussions	• Little to no tolerance of service interruptions • Risk of churn if reliability metrics aren't met

Source: Yankee Group "Inside the trends and Numbers of the Broadband Aggregation Market", June 2005

Table B.1: Network transformation process for triple play

Network processing power

Because each service has its own requirements for the characteristics of the underlying network, advanced QoS support will be a critical success factor for such a converged access network, requiring additional processing power to be present in IP access nodes. Furthermore, the deployment of service enablers or even full services in the access network puts additional strain on the access nodes' processing units.

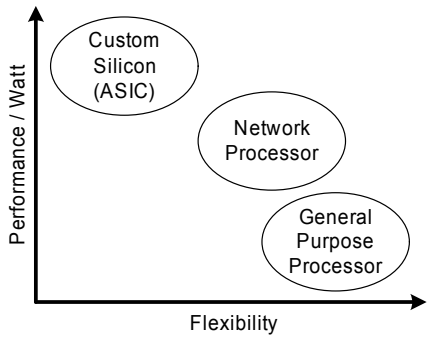


Figure B.4: Taxonomy

Traditionally, telecom equipment vendors have used fixed-function application specific integrated circuits (ASIC) to cope with the huge performance requirements of today's network systems. However, the ever-changing requirements of a service oriented access network ask for flexible solutions with assured time to market, while custom silicon provides little or no flexibility to introduce new protocols or services on existing hardware.

As opposed to ASICs, general-purpose processors certainly meet the flexibility requirements for implementing modern network services. However, they often lack performance or consume too much power (generate too much heat) for integration in large telecom systems.

For this reason, a hybrid device, called network processor (NPU), has emerged over the last few years. Network processors are highly parallel, programmable hardware, combining the low cost and flexibility of a RISC processor with the speed and scalability of custom silicon [2] (cf. Figure B.4). NPUs are considered an important technology for increasing application awareness of IP access nodes.

B.4 IPTV service deployment

Time-shifted TV

Due to the growing popularity of IPTV, a central server architecture has become insufficient to support these services. Recent deployments therefore introduce distributed servers at the edge of the core network, storing the more popular programs. The time-shifted TV concept however, as explained in more detail in the next section, even goes one step further and introduces the storage of small sliding intervals of streaming content in the access network. This way smaller, diskless streamers can be deployed close to the end users, at the proxies. This is most beneficial, in terms of network bandwidth, for very popular content, such as live TV shows on popular channels. Support of interactive commands (pause, fast forward or rewind) on live TV then becomes possible at the proxies, at least within the time window of the stored interval.

Distributed storage

End users have an increasing amount of multimedia data (digital photo albums, digital home videos, a digital music and movie collection, etc). A major opportunity of multi service access networks, is allowing users to transparently store, access and share their digital media library from anywhere. While harddisks are failure prone and recordable optical media only have limited archival lifespan [4], having a high speed network storage service, enabling users to virtually take their data with them wherever they go and relieving them of the burden of meticulously backing up all data, would make life a lot easier.

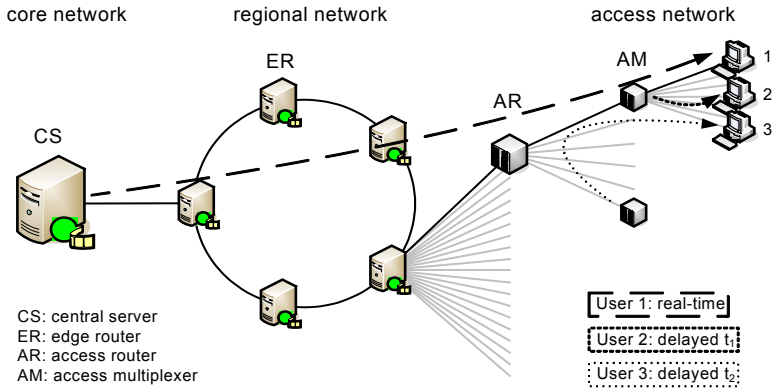
Guaranteeing fast access requires distributed servers and a pervasive replication mechanism, as introduced in [5], caching data wherever and whenever it is accessed. Since multimedia content is typically *read-mostly* data, no strong consistency is required between replicas. Occasional updates can be propagated periodically, at the same time deciding whether a replica should be retained or deleted (e.g. based on last access time, access frequency, etc). A minimum set of replicas should be maintained at all times in order to ensure reliability. A further speed-up of data access and sharing could be achieved by deploying small caches close to the end-user, operating in an analogous manner as the tsTV proxies.

B.5 Use case: time-shifted television

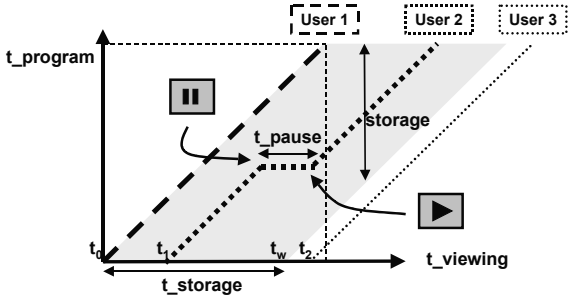
Concept

Time-shifted TV enables the end-user to watch a broadcasted TV program with a time shift, i.e. the end-user can start watching the TV program from the

beginning although the broadcasting of that program has already started or is even already finished.



(a)



(b)

Figure B.5: Time-shifted television: (a) typical access network topology and (b) tsTV streaming diagram

As shown in Figure B.1, the popularity of a television program typically reaches its peak within several minutes after the initial broadcast of the program and exponentially decreases afterwards. This means that caching a segment with a sliding window of several minutes for each current program can serve a considerable part of all user requests for that program, from start to finish, hence the benefit of using distributed streamers with limited storage capacity. In Figure B.5a and Figure B.5b for example, user 1 is the first to request a certain television program and gets served from the central server. Afterwards, other requesting users (e.g. user 2) can be served by the proxy, as long as the window

of the requested program is still grow-ing. After several minutes, the window stops growing and begins sliding, so that user 3 cannot be served anymore and will be redirected to the (central or regional) server or, in case of co-operative caching, to a neighbour proxy with the appropriate segment, if present. Pausing (parallel to the horizontal axis, Figure B.5b) can also be supported within the segment window, as well as fast forward or rewind (parallel to the vertical axis).

Caching algorithm

Our caching algorithm for tsTV services is presented in this section. Since we assume that in general only segments of programs will be stored, cache sizes can be limited to a few gigabyte in stand-alone mode or even less in case of co-operative caching. This way smaller streaming servers can be deployed closer to the users, without increasing the installation cost excessively.

We virtually split the cache into two parts: a small part *S* and a main part *L*. Part *S* will be used to cache the first few (e.g. 5) minutes of every newly requested (or broadcasted) program, mainly to learn about its initial popularity. Its size is generally smaller than 1 GB (typically 1 hour of streaming content).

Part *L* will be used to actually store the appropriate segments (with growing or sliding windows). These segments and their window size are chosen based on local popularity (especially useful in case of stand-alone caching), distance from the end user (important in case of co-operative caching) or a combination of both metrics. Figure B.6 shows the basic principle of the tsTV caching algorithm.

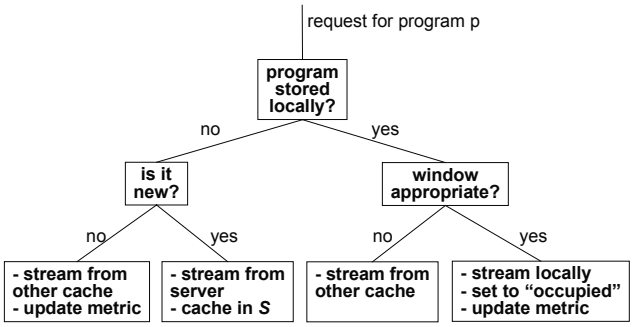


Figure B.6: Basic principle of the tsTV caching algorithm at each proxy

We assume that all caches know which segments are stored on the other caches, through a Cache State Exchange (CSE) protocol.

Deployment options

To demonstrate both deployment options, stand-alone or co-operative caching, simulations were performed on the typical access network topology shown in Figure B.5a. The server offers 5 popular channels through the tsTV service, each with 6 programs of 45 minutes per evening. The popularity of each program reaches a peak during the first interval (= 5 minutes) after the start and is halved after each interval (similar to Figure B.1), so that all requests for each program are made before the program has ended.

Figure B.7 shows the load on the different links between the edge server ER, the access routers AR and the access multiplexers AM from Figure B.5a. In stand-alone mode, requests that cannot be served by the cache at an AM are forwarded to its AR cache and, if necessary, forwarded to the ER (hierarchical). In co-operative mode, caches are present at the AMs only (no hierarchical caching), forwarding requests amongst each other effectively, using RTSP (Real-Time Streaming Protocol) messages [6].

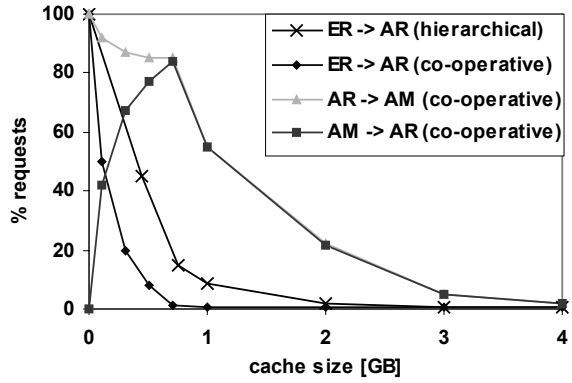


Figure B.7: Relative load on the links between ER, AR and AM (upstream and downstream) for hierarchical and co-operative caching

In co-operative mode, the server load decreases *n* times faster than in stand-alone mode without hierarchical caching, where *n* is the number of AM caches (6 in Figure B.7). At low cache sizes (<1GB), the access network traffic due to the cache co-operation is relatively high. When using larger caches, this load is reduced as well, since most requests can be served locally.

B.6 RTSP proxy

A transparent RTSP proxy for time-shifted TV has been implemented for evaluation purposes. An overview of the performance measurements on an AMD AthlonTM 64 processor 3000+ (512MB RAM) is presented in [6]. Figure B.8 shows the delay between a PLAY request sent by a PC client and the arrival of the first RTP packet at the PC client, for different configurations (server-proxy-client). Even when the proxy has to fetch the content from the server, the delay is never higher than 35 ms (1000 measurements per configuration). When the proxy acts as a mere router, the delay caused by the server is less than 1 ms. The delay on the network links between server, proxy and client is negligible.

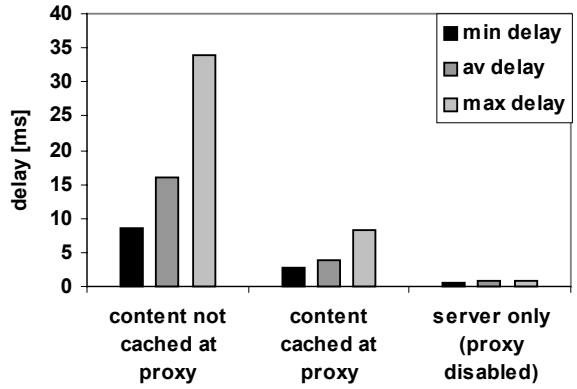


Figure B.8: Delay between a client request and the actual start of the RTP stream on a client PC

B.7 Conclusions

In this paper, the necessary transformations in access network architectures for next-generation broadband services have been described. Improved scalability, flexibility and availability can be achieved through the introduction of IP-aware network elements.

Due to their significant bandwidth requirements and steadily rising popularity, IPTV services have been identified as the main trigger for this evolution, offering opportunities for service providers to introduce other value added (interactive) services. One of the most promising IPTV services is time-shifted TV, which can be deployed using diskless distributed caches, effectively offloading the server and backbone network.

Acknowledgment

This work is partly funded by the IST FP6 MUSE project. MUSE contributes to the strategic objective "Broadband for All" of IST (Information Society Technologies) and it is partially funded by the European Commission.

References

[1] T. Stevens, K. Vlaeminck, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, "Deployment of service aware access networks through IPv6", 8th International Conference on Telecommunications, ConTEL 2005, Zagreb, Croatia, June 15-17, 2005, Vol. 1, p. 7-14.

[2] D.E. Comer, "*Network System Design using Network Processors*", 2004, Pearson Education Inc., New Jersey, ISBN 0-1314179-4-2.

[3] E. Gilon, W. Van de Meerssche et al., "Demonstration of an IP Aware Multi-service Access Network", BroadBand Europe 2005, December 2005, Bordeaux, France.

[4] T. Vitale, "*Digital Imaging in Conservation: File Storage*", AIC News Vol. 31 no. 1, January 2006.

[5] Y. Saito, C. Karamanolis, M. Karlsson, M. Mahalingam, "*Taming aggressive replication in the Pangaea wide-area file system*", 5th Symposium on Operating System Design and Implementation (OSDI 2002), December 2002, p. 15-30.

[6] T. Wauters, et al., "Co-operative Proxy Caching Algorithms for Time-Shifted IPTV Services", 32nd Euromicro Conference, Cavtat/Dubrovnik, Croatia - Aug. 28th - Sept. 1st, 2006.

[7] R. Mestric, M. Sif, E. Festraets, "Optimizing the network architecture for triple play", Alcatel Strategy White Paper, http://www.alcatel.com/doctypes/opgrelatedinformation/TriplePlay_wp.pdf, 2005.

C

Virtual topology design issues for variable traffic

T. Wauters, D. Colle, E. Van Breusegem, S. Verbrugge, S. De Maesschalck,

J. Cheyns, M. Pickavet, P. Demeester

Ghent University (INTEC) – IBBT – IMEC

Gaston Crommenlaan 8, bus 201

B-9050 Gent

IEICE Electronics Express, Electronic journal, http://www.elex.ieice.org/, September 25, 2004, Vol. 1, pp. 328-332.

Abstract

This paper discusses and evaluates, in terms of number of wavelength channels and router port count, different grooming strategies exploiting the benefits of statistical multiplexing. For the network design, a hybrid solution, combining the advantages of both the end-to-end and the link-by-link grooming scenario, is proposed.

C.1 Introduction

When designing a virtual topology, a key issue is to groom the traffic in the logical links in such a way that a good compromise between capacity efficiency and node cost is achieved. Two extreme grooming strategies exist. In end-to-end grooming, a dedicated logical link is used for each traffic demand, possibly resulting in a full-mesh virtual topology. In link-by-link grooming, each network node terminates all logical links entering that node: the virtual topology corresponds to the physical topology.

Variable traffic can be modelled using distributions. For example, the demands from one of the left nodes to the right node in Figure C.1 requires on average the capacity of 2.5 circuits, but per demand only 2.5*C*-*A* of the capacity is (almost) always needed, while up to 2.5*C*+*A* of the capacity is sporadically required. Grooming or aggregating variable traffic leads to a smaller variability relative to the average value: indeed, the variance typically follows a square-root dependency while the average traffic volume grows linearly [1].

A good approximation for the aggregated capacity needed to transport N traffic flows with an average of m_i and a standard deviation of σ_i, so that no more than a fraction ε of the traffic gets lost, is given by:

$$C_{\mathbf{a}} = M + \alpha \cdot \sigma \text{ with } \alpha = \sqrt{-2\ln(\varepsilon) + \ln(2\pi)} \quad (1)$$

In this equation M is the mean aggregate traffic rate and σ is the standard deviation of the aggregate traffic:

$$M = \sum_{i=1}^{N} m_i \text{ and } \sigma^2 = \sum_{i=1}^{N} \sigma_i^2 \quad (2)$$

The example of Figure C.1 (top part) illustrates how this feature can be exploited in order to decrease capacity/investment costs. With end-to-end grooming, each of the three demands needs 4 wavelength channels that are all cross-connected in the middle node: 3*4 = 12 wavelength channels are needed on the link between the middle and right node. However, in the link-by-link grooming case, the three demands are aggregated in the same logical link between the middle and right node, only requiring 2.5*C**3 + *A**√3 = 7.5*C* + *A**√3 channels. Thus, when *A* <= 1.5/√3, only 9 instead of 12 wavelength channels are needed on the fibre between the middle and right node.

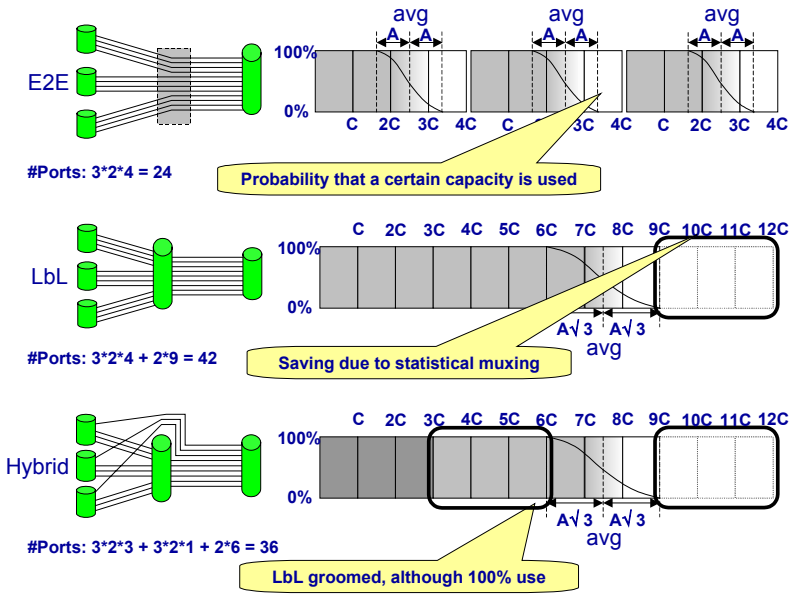


Figure C.1: End-to-end versus link-by-link grooming. The hybrid scenario combines the advantages of both strategies (C: capacity of a circuit/lightpath).

Figure C.1 also shows that, although link-by-link grooming benefits from the statistical multiplexing gain, it requires more router ports (expensive O/E interfaces): 42 instead of 24 router ports (thus an increase of almost 100%). This very simple design strategy did however not take into account that each end-to-end demand always needs more than one circuit/wavelength channel. And thus (see Figure C.1, bottom part), there is no need to terminate these completely filled lightpaths (dark grey capacity) in the middle node and to process each individual packet carried in these lightpaths. Thus, by cross-connecting lightpaths filled up for 100%, up to 2*3 = 6 router ports can be saved in the middle node, without impacting the statistical multiplexing gain.

At the bottom of Figure C.1 we consider only lightpaths with a filling of exactly 100% to be cross-connected in the middle node (thus a single wavelength channel per demand). However cross-connecting a second wavelength channel/circuit per demand would probably also make sense, since these circuits are also nearly completely filled. Of course, this would probably require an additional wavelength channel on the link between the middle and right node (and thus two additional router ports), but cross-connecting these three wavelength channels in the middle node will save there 2*3 = 6 router ports. Thus, this would result in a net gain of 6-2 = 4 router ports.

It might be interesting to reuse the unused capacity of these cross-connected lightpaths to transport some other packets (routed hop-by-hop) by marking these packets with an orthogonal label (e.g., FSK label): this is called Overspill Routing In Optical Networks (ORION) [2].

C.2 Evaluation

To evaluate the different options discussed in the previous section, we consider a tree network, consisting of 8 levels. A node in level *N* connects to 5 nodes in level *N*-1. It is assumed that between each leaf node and the root node on average a set of 10 streams have to be routed and that the capacity of a wavelength channel is equivalent to the bandwidth of 100 streams.

Hybrid grooming scenario

Figure C.2 shows the total number of router ports needed for different levels of traffic variability. The figure compares pure end-to-end grooming, pure link-by-link grooming and a hybrid end-to-end/link-by-link grooming design. The latter situation corresponds to what has been presented at the bottom of in Figure C.1: wavelength channels that are completely filled at level *N* are cross-connected in all higher levels. Thus in Figure C.1, level 2 would cross-connect only 3 wavelength channels, while level 3 would cross-connect 6 wavelength channels, due to the statistical multiplexing gain at level 2. In the peak design it is assumed that no statistical multiplexing gain can be achieved (the variance proportionally depends on the average traffic volume), while in the statistical design a square-root dependency for the variance results in the obtained statistical multiplexing gain.

As the figure shows, the larger the traffic variability relatively to the average traffic volume, the larger this statistical multiplexing gain (in this example up to 23% and 42% in respectively the link-by-link and hybrid grooming designs). Of course, in case of no variability, there exists no difference between the peak and statistical design. Note also that the hybrid grooming design is closest (but not equal due to the granularity mismatch between the demands and the wavelength channel capacity) to the end-to-end grooming design in case of no traffic variability and moves more and more to the link-by-link grooming design as the traffic variability grows.

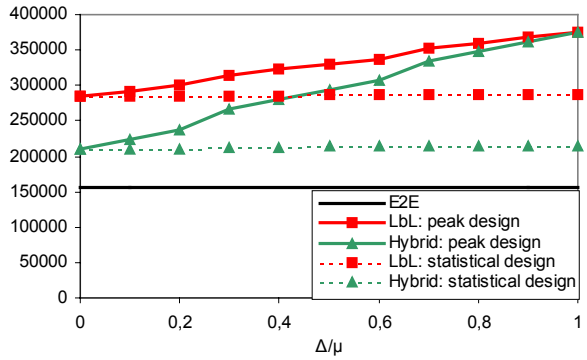


Figure C.2: Number of router ports for increasing traffic variability. The link-by-link and the hybrid strategies are compared to the end-to-end scenario, for peak and statistical design.

The hybrid and link-by-link grooming designs would become identical when at the pen-ultimate level, not a single wavelength channel gets cross-connected. This would be the case for a much higher traffic variability in the statistical design compared to the 100% variability in the peak design (i.e., proportional dependency between average and variance).

Network design

Figure C.3 considers the same network scenario as in Figure C.2 and assumes a relative traffic variability of 20%. As explained at the end of section 1, it can be interesting to also cross-connect highly (but not only completely filled) wavelength channels in the hybrid design. For this purpose we split the network in two parts: the first N levels are designed according to the hybrid grooming strategy (cross-connection of only completely filled channels) and the remaining 8-N levels are designed according to the end-to-end grooming strategy. It is clear from the figure that up to level 3 the statistical multiplexing gain is dominated by the impact of the grooming due to the granularity mismatch between the demands and the wavelength channel capacity. A depth up to level 4 realizes a statistical multiplexing gain of 13% (compared to the peak design) in terms of wavelength channels entering the root node, at the price of less than 2% more router ports compared to the design with a depth up to level 3.

Extending the hybrid design even further will never lead to a statistical multiplexing gain higher than 17%, requiring 1% more router ports compared to a depth up to level 4.

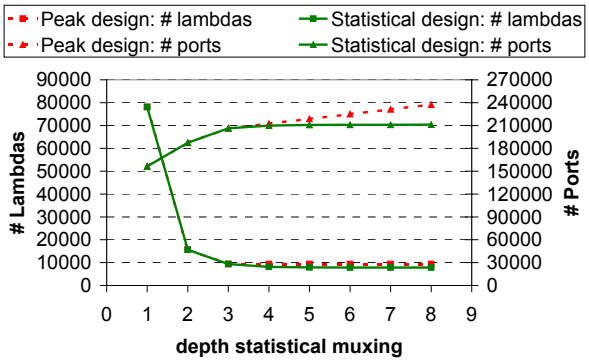


*Figure C.3: Increasing number of levels (*N*) assuming the hybrid grooming strategy, allowing for statistical multiplexing gain*

C.3 Conclusions

We have demonstrated that an important saving in wavelength channels and router ports can be achieved due to the statistical multiplexing gain in case of variable traffic, by comparing the peak and the statistical design. The hybrid end-to-end/link-by-link grooming strategy was found to combine the benefits of statistical multiplexing of the link-by-link grooming strategy and the lower number of router ports of the end-to-end grooming design.

By cross-connecting highly and not only completely filled wavelength channels in this hybrid scenario, a dominant fraction of the highest possible statistical multiplexing gain can already be realized at the price of a relative small router port penalty.

Acknowledgments

This work was partly funded by the EC through the IST-projects NOBEL, ePhoton/ONe and STOLAS, by the Flemish government through the FWO project G.0315.04, the IWT-project ONNA, the IWT/ITEA-project TBONES and by Ghent University through the BOF-project RODEO.

D. Colle, E. Van Breusegem and S. Verbrugge thank the IWT for its financial support through their post-doctoral/PhD grants. J. Cheyns is a Research Assistant with the Flemish Fund for Scientific Research.

References

[1] R. Guérin, H. Ahmadi, and M. Naghshineh, "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks", *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, September, 1991.

[2] E. Van Breusegem, et al, "Overspill Routing In Optical Networks: a new architecture for future-proof IP over WDM networks", *Fourth Annual Optical Networking and Communications Conference*, Dallas (USA), October, 2003.

D

Bandwidth management on MediaGrids for multimedia production and collaboration

T. Wauters, B. Volckaert, K. Lamont, B. Dhoedt, P. Demeester

Ghent University (INTEC) – IBBT – IMEC

Gaston Crommenlaan 8, bus 201

B-9050 Gent

Internal document

In this document, the network design and bandwidth management for a multimedia production and collaboration service is presented. Compared to the other service specific solutions discussed in this book, several differences can be noticed. A first difference is that besides plain multimedia content storage, computational resources are available in the network as well, to allow for production and modification of content. Furthermore, user profiles are much more complex, as different formats of a particular stream can be requested, new

content can be uploaded into the network or multiple streams can be downloaded simultaneously. A last major difference is that the system environment is professional and not commercial, which increases the QoS requirements considerably. Therefore, a MediaGrid is proposed, combining the benefits of Grid technology with the content delivery network techniques described in Chapter 2 of this book.

Further on in this document, we introduce the different application, user and company profiles studied in this project [3]. Afterwards, the MediaNSG Grid simulator developed by Bruno Volckaert is presented briefly, as further results are described in his Ph.D. book. Finally, a bandwidth management tool, implemented by Kristof Lamont, is described. Its goal is to balance the load on the network servers. A set of bin packing algorithms is proposed and evaluated on a basic network configuration as a proof of concept, as this project is still ongoing.

D.1 Introduction

Much in the same way as other businesses, the media industry has been confronted with an increasing complexity in both the technical domain and the business domain. Up until now, a broadcaster was an umbrella organisation for different kinds of in-house activities like media production, distribution and play-out, etc. More and more however, business drivers such as cost reduction, added value management, partnerships, global sourcing, and business componentization are forcing these companies to become more agile, find partnerships and evolve to dynamically extending organisations, with business models based on business services available within the media market. These parameters combined with possible future mergers, acquisitions and fusions drive the media businesses to become more agile.

Furthermore, exponentional decrease of harddisk costs [1] ignited a paradigm shift in the production of audiovisual media from tape to file based. Current cost per byte of harddisk based storage systems rivals that of tape based systems and is expected to go below the stagnating prices of the latter. Although todays architectures promise democratization of data access, i.e. inexpensive, non-mediated, and shared access to centrally-managed storage, this promise is only partially met by existing installations. On a software level, generic (Grid-enabled) applications are tuned towards typical ICT related requirements and are not yet fitted for the specific challenges induced by a file based media production and archiving platform.

In the long term, one wants to allow automated interaction between several audio/video media production sites, and share centralized storage, computational

and specialised (e.g. capturing devices, broadcasting equipment) resources with several independent corporate users in a controlled manner. It is in this domain that media production environments can benefit from Grid technology to both improve media handling/processing times and provide a means for securely sharing and utilising distributed resources and applications amongst multiple virtual organisations by employing specialized Grid middleware.

Due to the specific scenario however, current Grid technology can not be introduced in a straightforward way. The high bandwidth, reliability and short response time requirements when handling audio/video streams imply the need for special care in the design of the overall architecture and in particular in the scheduling and resource control process. Media handling can take place at local sites before streaming them to a remote site or can be performed at a remote site: the scheduling, resource control and QoS management components of the Grid will have a high impact on the achieved application performance. Furthermore, the software architecture of the management platform will need to exhibit high performance and reliability to meet the specific application requirements. The MediaGrid framework presented in this chapter has been developed to cope with these challenges, and will make it possible for media partners to evolve to extended organisations where partnerships, media communities and commercialisation of media services are omnipresent.

Advantages of Grid-enabling the audiovisual media production/distribution companies would be:

- Ability to distribute media files among different companies within an environment with high reactivity requirements and various levels of Quality of Service (QoS)
- Ease the exchange of media resources/assets (rendering farms, specialised media capture devices, etc.)
- Integration of broadcast media exchange standards (e.g. the EBU's P/Meta standard [2]) in a grid services environment to provide interoperability between different media content providers
- Migration from special purpose resources and applications to conventional IT hard/software
- Stimulate the growth of media community Virtual Organisation (VO) setups supporting advanced collaborative working

D.2 Application, user and company profiles

Together with the FIPA [3] partners from the media industry (more specifically the Flemish Radio- and Television Network [4] and Video Promotion [5], a company active on the broadcast television market), we studied the characteristics and requirements for the audiovisual applications that are to be supported by the MediaGrid architecture. This resulted in task, user and company profiles that have been implemented in the MediaNSG simulator (see Section D.3) and that can readily be used in simulations.

Application profiles

Audiovisual application classes show large differences in their processing, network and storage requirements. In Table D.1 we give an overview of average task class/application requirements of the most typical tasks/applications in a media centered company. The Quality of Service parameter can be used by MediaNSG while scheduling and during service management to ensure priorities are given to high QoS tasks. Table D.2 shows the network and storage requirements for different resolution audio and video streams.

- Ingest: deals with bringing media files onto the storage/archive system, extracting keyframes and constructing metadata about the ingested media.
- Quality checking, HiRes Browse: tasks from this class inspect the quality of media files in high resolution to see if it's fit for playout.
- LoRes browse: mainly used to rapidly shuffle through different archived media files in low resolution when trying to find specific or suitable source material.
- LoRes rough EDL: construction of a rough Edit Decision List (EDL). This Edit Decision List is a list of events that include the sources to be recorded from and information about transitions (cuts, dissolves, wipes), transition durations, etc.. Once an EDL has been processed, the result will be a newly constructed media file.
- Send to/Restore from archive: fetching data from the archive or storing new media files mainly stresses the available network resources.
- Craft editing: high quality finegrained editing and jog shuttling of multiple audio/video streams.
- Rendering, conforming, transcoding: this task involves rendering graphics, conforming of media to different video standards and transcoding of audio/video data to different qualities/resolutions/standards.

- Playout: Viewing multiple audio/video streams and sending one of those to playout equipment (e.g. broadcast equipment).
- Audio editing: Editing of multiple audio streams (possibly in conjuction with a video stream that needs to have the associated audio stream edited)
- Graphic creation: The creation of computer-generated imagery (CGI) imagery, custom scene transitions, ...

	Bandwidth	**CPU**	**Storage**	**QoS**	**No**
Ingest	Lo- or HiRes A/V	Low	0,65-25,7GB/h	High	1
Quality checking, HiRes browse	HiRes A/V	Low	25,7GB/h	Low	2
LoRes browse	LoRes A/V	Medium	0,65GB/h	Low	3
LoRes rough EDL	LoRes V, Lo- or HiRes A	High	0,5GB/h; 0,15-0,7GB/h	Medium	4
Send/Restore archive	Lo- or HiRes A/V	Low	0,65-25,7GB/h	Medium	5
Craft editing	5-10 HiRes A/V	High	5-10 25,7GB/h	High	6
Rendering, conforming, transcoding	HiRes A/V	High	25,7GB/h	Low	7
Playout	1-40 HiRes A/V	Low	1-40 25,7GB/h	High	8
Audio editing	Lo- or HiRes A/V	High	0,65-25,7GB/h	Medium	9
Graphic creation	HiRes V	High	25GB/h	Low	10

Table D.1: Average audiovisual application requirements

Streams	Bitrates	Storage
HiRes video	20-50 Mbps	25 GB/h
LoRes video	1 Mbps	0.5 GB/h
HiRes audio	1.5 Mbps	0.7 GB/h
LoRes audio	256 kbps	0.15 GB/h
HD HiRes video	200 Mbps	100 GB/h

Table D.2: Network and storage requirements for audio/video streams

User Profiles

Now that we have discussed the different application/task classes and their requirements, we can look at the different user classes of typical audiovisual companies, with each user class showing widely differing characteristics regarding which applications they use:

- Ingester: This profile includes tasks like quality checking and low resolution browsing, besides the actual ingesting of media onto the storage archive.
- Video journalist: The main tasks of a journalist are low resolution browsing, low resolution rough Edit Decision List (EDL) construction and rendering, conforming and transcoding.
- Audio/Video editor: an audio editor deals with mixing and editing multiple audio tracks, while video editing includes quality checking, craft editing, rendering/conforming/transcoding and graphic creation.
- Producer/Director: involved at different stages of media production, mainly doing low resolution browse tasks, with the occasional sending to/restoring from archive and some quality checking and/or high resolution browsing.
- Playout: tasks include quality checking, low resolution browsing and playout.
- Archivist: an archivist mainly performs low resolution browsing and sending to/restoring from archive.

This information, together with the user/task workflows (presented in Figure D.1) and average application characteristics presented in Table D.3, has been used to construct accurate media user profiles for use in MediaNSG simulations.

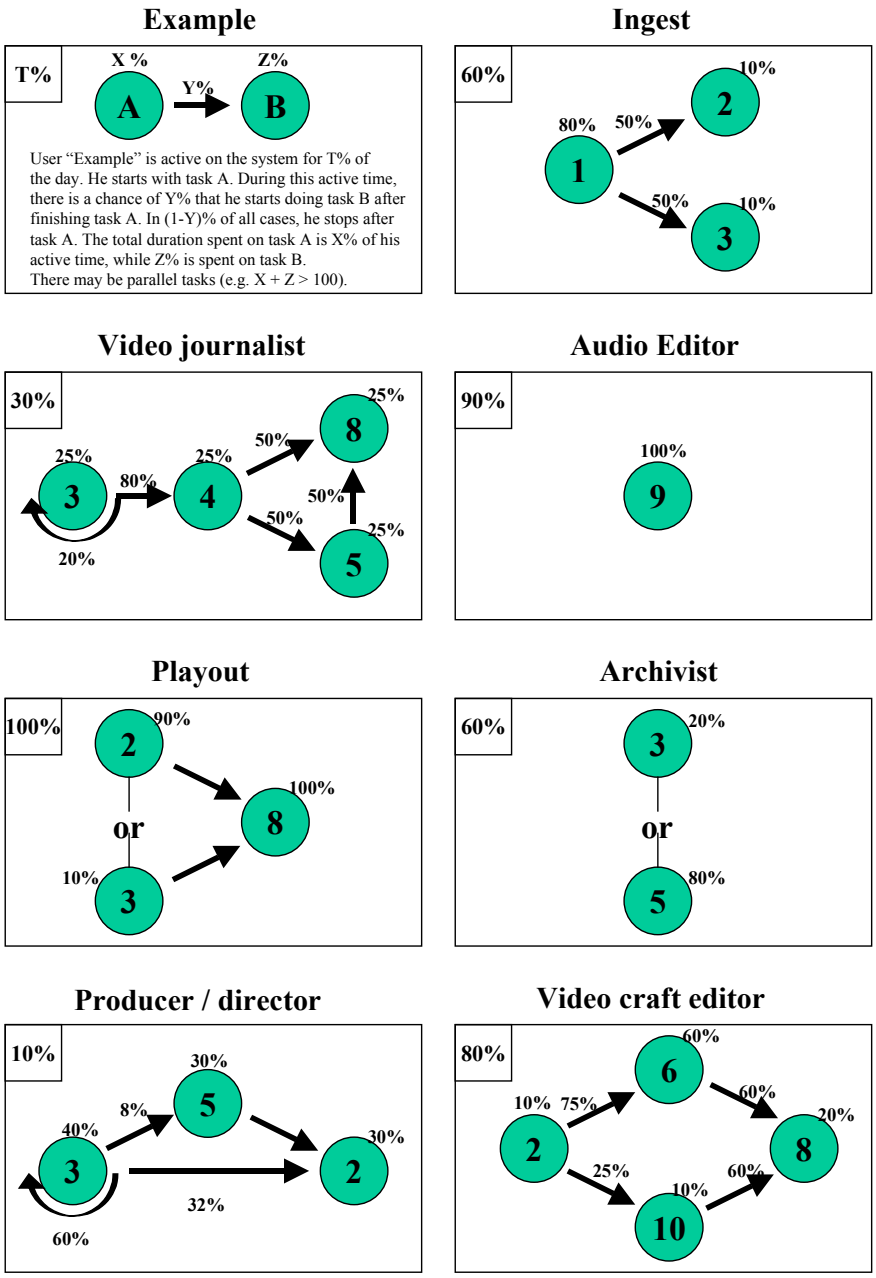


Figure D.1: Task workflow of typical audiovisual company user profiles

Company profiles

Finally, profiles have been provided for typical audiovisual companies (mainly describing the average amount of users from each userclass working simultaneously). The most important profiles are:

- Television production: an example of television production is news program production. In these organizations tens (regional) or hundreds (national) of video journalists gather information that has to be ingested, edited, archived and played out.
- Television post production: in a post-production facility the same user classes are present, along with producers / directors managing the studio work.
- Television broadcast: television broadcast companies are not involved in (post) production. The focus is more on playout than on editing.
- Television program supplier: these companies combine individual items into finished programs and send these to television broadcasters. Editors and producers/directors are the most important user classes in this type of organization.
- Video on Demand: companies delivering Video on Demand services mainly focus on indexing of the available material, user and channel dependent encoding of the streams and play out.
- Radio broadcast: similar to television broadcast, but with different requirements (e.g. no buffering or delays allowed).

D.3 Media grids

If we wish to develop MediaGrid suitable scheduling/service management algorithms, or wish to evaluate the performance of different network/computational/storage resource configurations, we either have to construct a testbed and measure task/resource performance, or we can simulate the MediaGrid's behaviour. Due to the size and the amount of resources involved in setting up a MediaGrid testbed each time a new scenario needs to be evaluated, accurate simulation of MediaGrid scenarios is likely to be more efficient.

	Ingest	Video journ.	Audio ed.	Video ed.	Prod. / dir.	Play out	Archi vist
Regional TV prod.	2	30-50	2-3			2	2
National TV prod.	3	300-500	20-30			3	4
TV post prod.	1	10-50	1-3	5-20	10	1	1
TV broadcast	1	5-10	1-5			1	1
TV prog. supplier			25		25		
Video on Demand	2		5			2	1
Radio prod. / broadcast			30		20	50	

Table D.3: Audiovisual company average user class representation

MediaNSG, a MediaGrid specific extension to NSGrid has been developed allowing users to simulate typical task submission behaviour of different media company organisations and experiment with scheduling and service management architectures. MediaNSG supports the simulation of both Micro (single site) and Macro Grid behaviour (Grid comprised of different interconnected Micro Grids), and provides the user with output data regarding job execution statistics (job response time, time spent in scheduling queue, data transfer size/speed, etc.) for the different tasks, resource (computational, storage and network resources) and management component (scheduler, information service, etc.) usage statistics, bottlenecks, etc..

Grid model

In MediaNSG, we regard a Macro Grid as a collection of Micro Grid sites (Figure D.2) interconnected by WAN links (Figure D.3). Each Micro Grid site has its own resources (computational, storage and data resources) and a set of

management components, all of which are interconnected by means of LAN links. Management components include a Connection Manager (capable of offering network QoS by providing bandwidth reservation support, and responsible for monitoring available link bandwidth and delay), an Information Service (storing registered resources' properties and monitoring their status) and a Scheduler. The Service Monitor and Service Management components deliver advance resource reservation support in order to provide Quality of Service to jobs.

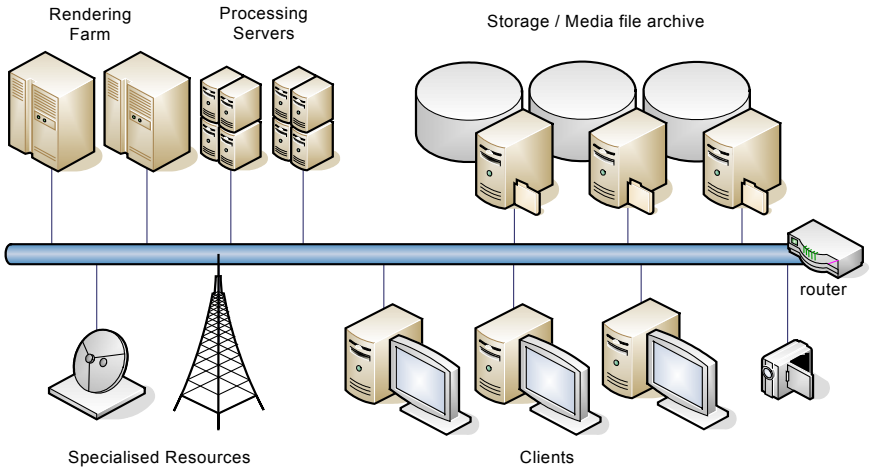


Figure D.2: Micro Grid

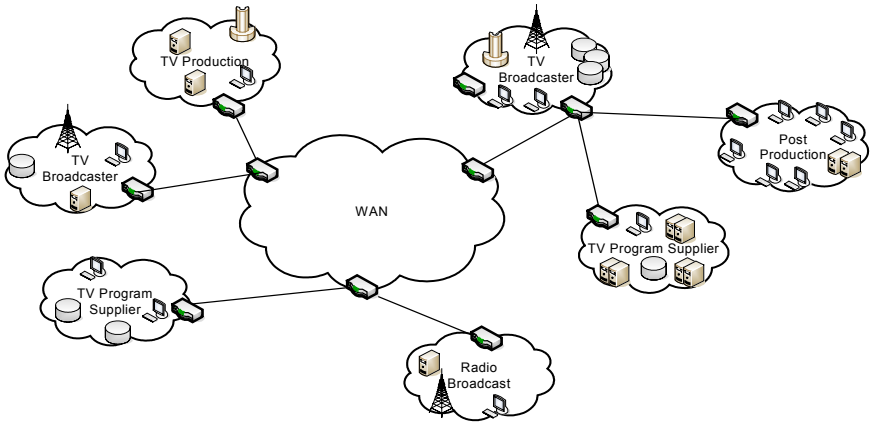


Figure D.3: Macro Grid

D.4 Bandwidth management

Besides scheduling/service management algorithms suitable for this MediaGrid, we also need an efficient bandwidth management solution. The aim is to determine the number of servers required to cope with the peak load from the end users and balance this load over all available network servers.

Network configuration

At this moment, no network restrictions are taken into account. Figure D.4 therefore shows a simplified network configuration where a group of clients is connected to the servers through a switch. We assume that there are no limitations to the switching capacity and no blocking occurs. The peak demand from each client, given as a multimedia stream with a particular bandwidth, is known in advance and can be calculated from the user profiles described earlier in this document. Two approaches to this dimensioning problem, similar to traditional bin packing or knapsack models, are identified.

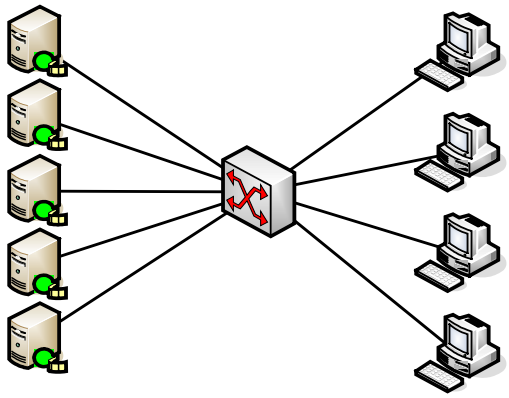


Figure D.4: Network configuration

For a given set of items (= input streams), each with a certain height (= bandwidth), the first model fixes the number of infinite bins (= servers without capacity restrictions taken into account) and distributes the items (= streams) in such a way that the maximum height of all bins (= maximum server load) is minimized.

The second model (the traditional bin-packing model) follows an alternative approach, by determining how many finite bins (= servers with a given limited capacity) you need at least to store all items (= to support all clients).

Model 1: fixed number of servers with infinite capacity

At the moment, two algorithms have been implemented.

- *Brute Force*: this algorithm calculates all possible combinations of clients and servers and therefore always finds the optimal solution minimizing the maximum server load. However, since the problem is NP-complete, the calculation time increases exponentially (the number of possible combinations equals (# servers) ^ (# clients)).
- *Sort & Fit*: this heuristic first orders all requests from large bandwidth to small bandwidth and then directs these requests step by step to the server with the smallest load.

The *Sort & Fit* heuristic performs optimal in most practical cases, with input parameters derived from the users profiles. Below is an example of a combination of clients and servers where the *Sort & Fit* heuristic does perform suboptimal.

When 7 clients are connected to 3 servers through a perfect switch and they each request a stream with bandwidth of 2, 3, 4, 7, 8 and 9 units, the optimal solution, calculated by the *Brute Force* algorithm, is as follows:

- Server 1 serves 9 + 3 = 12 bandwidth units
- Server 2 serves 8 + 4 = 12 bandwidth units
- Server 3 serves 7 + 3 + 2 = 12 bandwidth units

The *Sort & Fit* algorithm however finds the following suboptimal solution:

- Server 1 serves 9 + 3 = 12 bandwidth units
- Server 2 serves 8 + 3 = 11 bandwidth units
- Server 3 serves 7 + 4 + 2 = 13 bandwidth units

Model 2: variable number of servers with finite capacity

At the moment, two algorithms have been implemented, similar to those for the first model.

- *Brute Force*: this algorithm first calculates the minimum number of servers needed to serve all request, then analyses all possible combinations of directing clients to those servers and increases the number of servers by one in case no combination can be found where all servers can handle the traffic. It again always finds the optimal solution.

- *Sort & Fit*: this heuristic first orders all requests from large bandwidth to small bandwidth and then directs these requests to the server with the smallest load at that time, adding a new server if necessary. This heuristic is suboptimal in some cases, but never by more than 22% (proved by Hoffman [6]).

Bandwidth management tool

The java based bandwidth management tool implements the above-mentioned algorithms and calculates the input parameters for each of the user and application profiles discussed earlier. As a consequence, the number of servers and their capacity can be calculated for each company profile. New profiles can be added and existing profiles can be modified. Text based output files show the results of the algorithms and also provide detail on the specific client-server connections. Figure D.5 shows a screenshot of the current version of the tool.

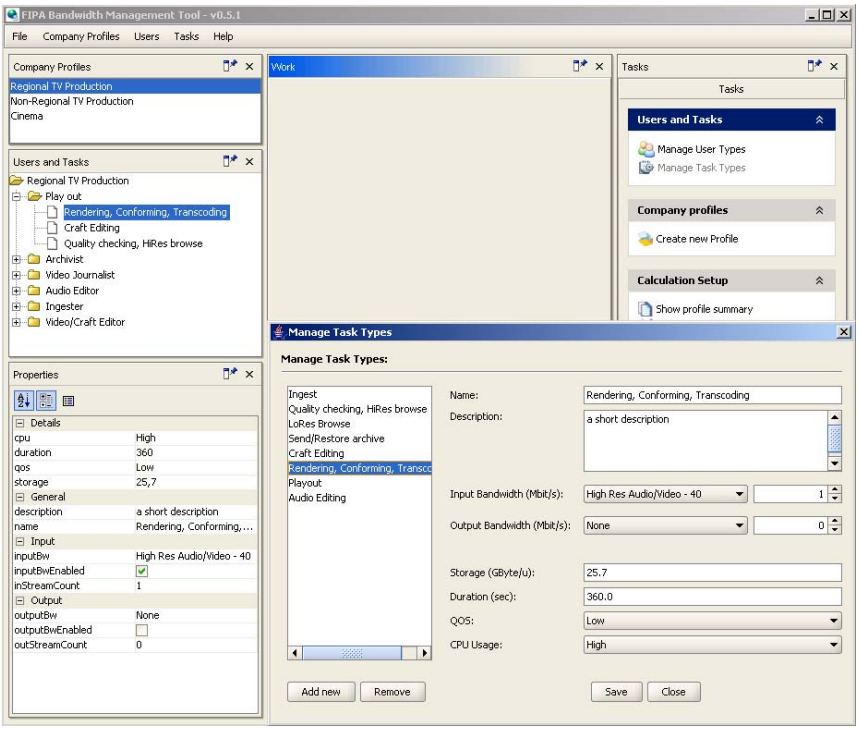


Figure D.5: Screenshot of the bandwidth management tool

References

[1] S. Gilheany. Projecting the Cost of Magnetic Disk Storage Over the Next 10 Years. http://www.archivebuilders.com/whitepapers/22011p.pdf, 2001.

[2] EBU Project Group P/Meta Metadata Exchange Scheme, V. 1.0. http://www.ebu.ch/en/technical/trev/trev 290-hopper.html.

[3] FIPA - File based Integrated Production Architecture Project. https://projects.ibbt.be/fipa/.

[4] VRT - The Flemish Radio- and Television Network. http://www.vrt.be.

[5] Video Promotion. http://www.videopromotion.be.

[6] Hoffman, P. The Man Who Loved Only Numbers: The Story of Paul Erdos and the Search for Mathematical Truth. New York: Hyperion, 1998.