Virtually@Home

ANDERS NILSSON
och
MAGNUS LINDBERG

KTH Information and Communication Technology

Bachelor of Science Thesis
Stockholm, Sweden 2009

TRITA-ICT-EX-2009:219

Virtually@Home

Anders Nilsson
Magnus Lindberg

2009-12-02

Bachelor of Science Thesis

Mentor and examiner: Prof. Gerald Q. Maguire Jr.

School of Information and Communication Technology
Royal Institute of Technology (KTH)
Stockholm, Sweden

Sammanfattning

Med dagens snabba utveckling av teknik och informationsteknologiska system har krav ställts på säkerhet och tillgänglighet hos både företag och privatpersoner. Fler och fler vill ha tillgång till sina privata filer och tjänster även när de inte är hemma. Geografiska begränsningar kan vara ett problem, och har medvetet lagts till på ett flertal nya tjänster på internet. såsom IPTV och andra mediaprodukter.

Idag är det varken säkert eller lämpligt att skicka känslig information över publika anslutningar, såsom Internet. På grund av detta har ett flertal lösningar såsom IPsec, SSL/TLS, med flera andra tekniker utvecklas och implementerats. Med hjälp av dessa kan man skapa krypterade anslutningar mellan två ändpunkter genom att använda TCP/IP protokollet. Detta går sedan i sin tur att utnyttja för att komma åt filer och tjänster som man skulle gjort om man faktiskt varit hemma – du är Virtually@home.

Rapporten visar hur man skapar en lösning till detta problem på ett enkelt, kostnadseffektivt och säkert sätt med hjälp av en proxy och ett VPN. Vi går igenom hur olika tekniker fungerar och varför vi anser att denna lösning är optimal för vårt syfte.

Abstract

With today's rapid development of technology and IT systems the demand for security and accessibility by both companies and individuals has increased. More and more people want access to their private files and services even when they are not at home. Geographical limitations can be a problem, and some such limitations have been deliberately added to a number of new services on the Internet, such as IPTV and other media services.

Today it is neither safe nor appropriate to send sensitive information over public connections, such as the Internet. Because of this, several solutions including IPsec, SSL / TLS, and several other technologies have been developed and implemented. With these, you can create encrypted connections between two endpoints over an internet protocol. This implies in turn that you can access files and services just as you would have done if you actually been at home – you are “Virtually@home”.

The report shows a solution to the problem of how to be virtually at home in a simple, cost-effective and safe manner by using a proxy server and a VPN. We review how each of the relevant technologies work and why we believe that this solution is an optimal solution to this problem.

Innehållsförteckning

Figurlista

Förkortningar

ACK	Acknowledgement
AH	Authentication Header
ARP	Address Resolution Protocol
ARPANET	Advanced Research Projects Agency Network
AS	Autonomous System
CA	Certificate Authority
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DNS	Domain Name System
DSL	Digital Subscriber Line
ESP	Encapsulating Security Payload
GNU	GNU's Not Unix
GUI	Graphical User Interface
HTTP	HyperTest Transfer Protocol
HTTPS	HTTP Secure
ICV	Integrity Check Value
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IKE	Internet Key Exchange
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPTV	IP Television
IPsec	IP Security
ISAKMP	Internet Security Association and Key Management Protocol
LDAP	Lightweight Directory Access Protocol
LZO	Lempel-Ziv-Oberhumer
MC	Multicast
MFC	Multicast Forwarding Cache
MPEG-4	Moving Pictures Experts Group-4
Mbps	Megabits per second
NAT	Network Address Translation
PIM	Protocol Independent Multicast
PIM-d	PIM deamon
RFC	Request For Comments
RSA	Ron Rivest, Adi Shamir, Len Adleman
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SA	Security Association
SSH	Secure Shell
SSL/TLS	Secure Socket Layer / Transport Layer Security
SYN	Synchronize
TAP	Terminal Access Point
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / IP
TTL	Time To Live
TV	Television
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VLC	VideoLan Client
VPN	Virtual Private Network
YaST	Yet another Setup Tool

1 Inledning

På senaste åren har fler och fler börjat utnyttjat Internet och dess sociala samhälle. Detta har lett till en snabb utveckling inom teknik och kommunikation. Man har även satsat stora summor i en utbyggnad av de existerande telefonnäten samt nybyggnationen av fibernät (även känt som stadsnät). Resultatet har blivit att i stort sätt hela landet har nu tillgång till pålitliga och snabba uppkopplingar, vilket i sin tur har lett till att flera företag och privatpersoner har börjat utveckla tjänster och produkter inom media och fildelning.

I samband med denna utveckling har datorer och dess hårdvara blivit snabbare och billigare samt priset på både lagring och datorkraft har kraftigt sjunkigt. Denna förändring har bistått till en ökad kvantitet av tjänster som förmedlar information och media över Internet. Detta har även spelat en roll i utvecklingen från de triviala videokonferenserna till IP-telefoni, som sedan vidareutvecklats till IPTV och Video on demand sändningar. Dessa sändningar streamas från tv-operatörerna över TCP/IP, vilket har till viss del tagit bort behovet av det analoga marknätet. Även den vanliga telefonen har börjat fasats ut genom IP-telefoni.

Det nya sättet att ”se” på TV har revolutionerat synen på tv-sändningar och reklam. På detta sätt kan tv-operatörer garantera klienter sändning då de hyr in sig i Internetleverantörer nät och på så sätt minimera konkurrensen med andra operatörer. De kan även skapa direktreklam speciellt framtagen för den enstaka kunden och på så sätt öka värdet för deras reklamkunder och då tjäna mer pengar.

Då denna teknik bygger på multicasting har ett stort dilemma uppstått. På grund av den ofantliga mängden paket som multicasting skickar, skulle detta skapa för mycket kongestion på Internet om dessa paket routades fritt på nätet. Av den anledningen är det standard att inte routa multicastpaket utanför ens AS eller lokala router. Detta begränsar TV-operatörerna till det lokala nätet som deras internetleverantör äger, vilket betyder att man som abonnent endast kan använda sig av tjänsten när man är hemma.

Vårt problem är alltså vad gör vi för att komma åt dessa tjänster när vi inte är hemma i det lokala nätet? Hur kan man routa multicastpaket till utomstående nätverk? Om man kan skicka multicast till hosts i andra nät, går det att utnyttja detta för andra typer av paket? Detta är precis det vi har valt att lösa.

Målet med projektet är att lösa detta problem och utveckla en okomplicerad lösning som kan tillämpas med minimala utgifter, så att du som användare kan använda dina tjänster var du än är i världen. Nästa steg är göra detta säkert då känslig information lätt kan hamna i fel händer när den skickas över något så publikt som internet. Detta är ett stort problem och olika krypteringstekniker som SSL/TLS med

flera har utvecklats. Därför kommer vi även att gå in på hur man kan skapa en säker krypterad uppkoppling mellan hemmet och fjärrdatorn.

1.1 Bakgrund

1.1.1 Vad har andra prövat?

Vad vi kunnat efterforska är att tester och lösningar med andra hårdvaruplattformar och tekniskt utrustningar existerar. Det finns varianter där man använder switchar och VLAN samt routrar för att skicka vidare ljud och bild till HTTPS-servers och andra mediatjänster. Dessa tjänster kostar oftast pengar och man är beroende av en tredjepartsleverantör.

Vår lösning bygger på att du ansluter dig till en proxy som ger dig access till ditt lokala nätverk samt de tjänster som levereras dit. Detta tillvägagångssätt är vad vi vet unik i den mening att vi använder oss av en dator som agerar proxy och router, samt använder enbart programvara som är öppen källkod, vilket gör att kostnaden är minimal.

1.1.2 Vad behöver du som läsare kunna för att förstå vår rapport?

En person med teknisk bakgrund och grundläggande kunskaper inom nätverk och kommunikation ska kunna förstå, och om denna så vill, implementera vår lösning. Denna rapport är uppbyggd på ett sådant sätt att läsaren ska kunna förstå, och med vår hjälp, skapa sitt eget virtuella nätverk och utnyttja det. För personer med bakgrund inom teknik och nätverk kommer denna rapport skapa en grund för hur man kan göra, för att sedan själva utveckla och förbättra de aspekter som de tycker är bristande.

1.1.3 Avgränsningar

Vi begränsade vårt projekt till vanlig streaming och vidarebefordran av de paket som kom ifrån streamingservern. Då vi inte hade tillgång till varken IPTV leverantör eller settop-box skapade vi vårt eget nätverk med en streamingserver som var kopplad till en router som i sin tur satt kopplad till vår proxy (se Figur 26 på sidan 30). Vi skapade ett labbnät som var så likt Internet vi kunde så att våra tester och resultat skulle kunna tillämpas i den riktiga världen. I detta labbnät kunde vi simulera riktiga nättjänster och vidarebefodra trafiken genom proxyn till klienten på andra sidan VPN-tunneln. På grund av tidsbegränsningar har vi valt att inte testa IP-telefoni fastän detta skulle vara fullt möjligt att tillämpa i vår slutprodukt.

I vår lösning finns möjlighet till omkodning, d.v.s. att manipulera dataströmmen och koda om den för att t.ex. använda mindre bandbredd (se Figur 27 och Figur 28 på sidan 31 och 32). Vi har dock valt att inte utveckla denna möjlighet mer än att testa att den fungerar. Detta beslut tog vi då hårdvara som vi hade tillgänglig under projektet enbart klarade (CPU-mässigt) extremt enkla omkodningar såsom skala. Kraftfullare datorer med snabbare processorer skulle klara av tyngre uppgifter som att byta video-codec och audio-codec, bitrate, med flera.

Vårt labb har använt ett 100Mbps ethernet nätverk och på grund av det har vi inte praktiskt kunnat testa de olika bredbandstekninkerna såsom ADSL, fiber, och kabel-modem. Detta limiterade oss till att analysera bandbreddstatistik med hjälp av IPtraf vid olika streamingförhållanden. Eftersom det inte finns en standard för IPTV utan enbart riktlinjer har vi inte kunnat replikera exakta IPTV strömmar då vi inte haft tillgång till en IPTV leverantör eller en settop-box. Vi har istället arbetat med olika

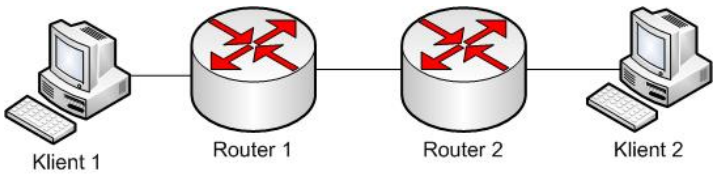
dataströmmar av MPEG4-teknik i olika kvaliteter för att bestämma bandbreddsutnjyttjande och på så sett testat oss fram till olika förhållande och krav på uppkopplingar mellan proxy och VPN-klient. Vi har alltid förutsatt att bandbredden mellan proxy och streamer/IPTV leverantör är tillräcklig för maximal kvalitet (>10Mbps).

2 Teori

2.1 Transmission Control Protocol/Internet Protocol (TCP/IP)

Transmission Control Protocol/Internet Protocol (TCP/IP) används för att skicka data mellan olika klienter eller servrar på internet[1]. TCP/IP är en utveckling av det gamla ARPANET som användes av den amerikanska militären på 1970-talet. Denna utveckling var nödvändig för att öka skalbarheten i den tidens nätverk. TCP/IP gav möjligheten att sammankoppla flera nätverk oberoende av de två lägre lagren i protokollstacken, däremot behövde näten kunna prata IP för att kopplas ihop till ett större nätverk. För att kopplingarna skulle fungera mellan näten infördes en ny typ av ”gateway” som fick namnet ”router” (se Figur 1). En router består av en maskin med flera nätverksinterfaces som kopplar samman och vidarebefordrar inkommande paket till ett annat interface, som i sin tur skickar paketet vidare mot sin destination.

Routern är nyckeln i detta koncepts skalbarhet och ger möjligheten att sammankoppla oändligt med nätverk av olika storlek till gigantiska nätverk som till exempel det vi idag kallar Internet.



Figur 1: Bild av ett TCP/IP-nät

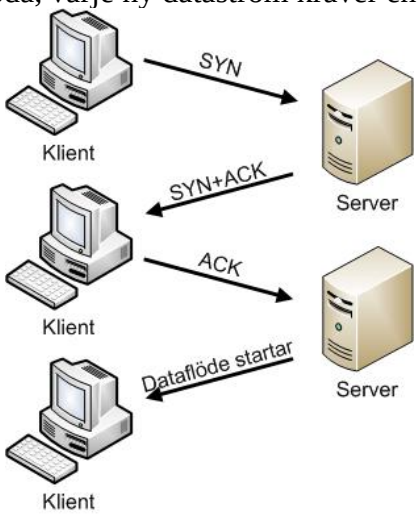
2.2 Transmission Control Protocol (TCP)

Transmission Control Protocol används för att kommunikation mellan applikationer på klienter och servrar och är ett förbindelsebundet protokoll. Till skillnad från andra transportprotokoll arbetar TCP i lager längre upp och hanterar pålitliga och ordnade strömmar av data mellan applikationer. Dess tillämpning finns i flera olika typer som webbtrafik, filöverföring, e-mail, med flera. En fördel med TCP över andra protokoll är att den kan hålla koll på om paketen i dataströmmarna kommer i ordning eller om det saknas paket. Om paketen är i oordning kan protokollet ordna paket efter det sekvensnummer som finns i TCP-headern. Om paket skulle saknas kommer protokollet begära en ”retransmission” av det saknade paketet, det vill säga begära att paketet sänds om. Protokollet hälper även till med att hantera överföringshastighet, paketstorlek och trafikstockning i nätverket.

Nackdelen med dessa funktioner är att protokollet blir långsammare och lite mer krävande än dess motsvarighet UDP. Därför lämpas inte TCP för datatrafik som är beroende av latens och snabba pakethanteringar. TCP är bra för hantering av

webbtrafik, e-mail, SSH, med flera. Däremot när det kommer till att streama film, musik eller Voice-over-IP passar UDP bättre.

TCP använder sig av sessioner (se Figur 2) när den pratar med andra applikationer. Vid upprättande av en sådan session kommer TCP skicka SYN meddelanden och begära att mottagaren ACK mottagna paket. När en session är upprättat börjar trafiken flöda, varje ny dataström kräver en ny session.



Figur 2: 3-way handshake

Då storleken av data varierar kommer den i de flesta fall styckas upp i mindre segment av TCP-protokollet. Dessa segment får ett sekvensnummer tilldelat av TCP som bestämmer dess ordning i uppstyckningen. När det sedan skickas kommer paketen troligtvis ta olika vägar till sin destination och det är anledning till att paketen har sekvensnummer. Skulle paket komma i oordning så kommer TCP själv ordna om paket efter sekvensnummer vid återskapandet av den mottagna datan. Om ett nummer saknas kommer TCP att begära en ny kopia av det saknade paketet innan datan sätts ihop.

För att trafiken inte ska överbelasta mottagaren kan TCP hantera hastigheten av dataströmmen så att pakten inte kommer för snabbt. Detta gör den genom att begära ACK meddelanden från mottagaren så den vet att den är redo att ta emot mer trafik.

För att motverka korruption och oordning skapar avsändare en lista på det som skickas ut. Varje paket på den listan har ett sekvensnummer och en timer, och när avsändaren får ett ACK meddelande gällande ett visst paket med ett eget sekvensnummer tas detta paket bort från avsändarens lista. Skulle timer gå ut innan avsändaren fått ett ACK meddelande kommer samma paket skickas igen då avsändaren förutsätter att detta paket tappats eller på annat sätt kommit bort på väg till mottagaren. I det fall mottagaren erhållit paketen men inte hunnit skicka ett ACK meddelande kommer TCP protokollet droppa det duplicerare paketet hos mottagaren för att inte skapa oordning.

Nyckeln till TCP pålitlighet ligger i att både mottagare och avsändaren har listor för att hålla koll på vad som skickas och tas emot. Samt att TCP paketerar data med sekvensnummer för att hålla reda på i vilken ordning paketen ska ligga när de kommer mottagaren.

2.3 User Datagram Protocol (UDP)

User Datagram Protocol (UDP) är ett stateless/connectionless protokoll som inte är beroende av att klienten upprättar sessioner som TCP[2]. Utav denna anledning saknar detta protokoll pålitlighet och möjligheten till ordnade dataströmmar och hantering av duplicerade paket. UDP förutsätter att all felkorrigering är onödig eller att applikationerna hanterar detta, vilket resulterar i att UDP paket kan hanteras snabbare.

En fördel med detta är att det finns frihetheten att välja om de vill ha pålitlighet eller hastighet. Därför är UDP mer lämpat för trafik som kräver låg latens och inte påverkas av att en del paket försvinner. Exempel på dessa är DNS, streaming, dataspel och filöverföring. För att erhålla pålitlighet med detta protokoll måste det ske i applikationerna.

Till skillnad från TCP behöver UDP inte starta en session innan den börjar skicka data utan börjar direkt när kontakten upprättats med mottagaren. Detta ger protokollet stor fördel för servrar som DNS som får många förfrågningar med små mängder data och även hos andra applikationer inte kräver pålitlighet eller hindras av det såsom streaming verktyg.

En nackdel med UDP och dess avsaknad av felkorrigering och duplicerade paket är att flödeskontrollen av paket måste ske av applikationer eller nätverkshårdvara. UDP-paket tar inte heller hänsyn till ordning eller hastighet och har företräde framför TCP-trafik. Detta medför att nätverk med många UDP-krävande applikationer får sämre kvalitet på TCP-trafiken då UDP tar upp för mycket plats.

2.4 Streaming

Streaming är att skicka ljud eller video alternativt båda mellan en server och en eller flera klienter[3]. Denna teknik har på senare år blivit vanligare och allt mer populär i samband med bredbandsutveckling som gett oss tillgång till billiga höghastighetsuppkopplingar. Att ”Streama” är att skicka en video sekvens eller ljudfil i realtid mellan två ändpunkter. Denna ström kommer att visas hos mottagaren samtidigt som den skickas från avsändare och inte lagras på mottagaren hårddisk.

Streaming sker i två olika lägen: live eller on-demand. När något streamas live skickas data direkt och dataströmmen mellanlagras normalt sett inte. Detta medför att live streams inte kan påverkas utan visas i en följd utan pauser. Vid on-demand streaming lagras mediet på server och klients begär att få titta på dem när den själv vill. Dessa media lagras normalt på en server och kan spelas, pausas och spolas i på begäran av klienten.

Detta kan uppnås på flera olika sätt och med flera olika programvaror. Det vanligaste sättet är att använda sig av RTP protokollet och skicka ut sina video-/ljudströmmar på multicastadresser där en eller flera kan gå med i gruppen och se på strömmen. Det förekommer även streaming till enskild klient, denna teknik använder unicast istället för multicast och det är endast klienten som begärt streamen som kan se den.

Ett alternativ till RTP är HTTP, RTSP och UDP, dessa protokoll utför liknande streaming fast till andra ändpunkter. Vid HTTP-streaming skickas strömmen till en web-server som i sin tur visar den på en hemsida. RTSP fungerar på ungefär samma sätt som HTTP men med flera val som kan skickas med dataströmmen, denna teknik

stödjer inte video-on-demand. Mer detaljer om de nämnda protokollen tas upp i följande sektioner.

2.4.1 Real-Time Transport Protocol (RTP)

Real-Time Transport Protocol (RTP) används för att skicka ljud och video över Internet[4]. Protokollet används till applikationer inom Voice-over-IP, Streaming media och IP-telefoni. RTP jobbar tillsammans med ett annat protokoll som heter RTCP (Real-Time Control Protocol). Detta protokoll hanterar överföringsstatistik och servicekvalitet.

RTP skapades för att transportera ljud och film. RTP ger ingen leveransgaranti som retransmission men den kan känna av saknade paket och kompensera för korruption och jitter. RTP stödjer multicast och är det vanligaste transportprotokollet för multicaststreams. RTP har som standard att skicka över TCP, men då dess användning inom streaming är så markant och behovet av snabba tidsenliga dataströmmar är stort, så används oftast UDP-protokollet tillsammans med RTP. Anledning till att UDP används är att ljud- och filmströmmar inte har ett behov av felkorrigering och retransmissions. I de flesta fall när man lyssnar eller ser på en ström kommer det inte märkas om ett fåtal paket tappas på vägen.

RTP använder sig av profiler när den skickar data mellan server och klient, dessa profiler innehåller information om dataströmmen, sekvensnummer och tidsstämplar med flera. Dessa används av mottagaren för att hålla reda på ordning och typ av data den erhåller från källan. Sekvensnumret används för att upptäcka paketförluster och tidsstämpeln till att spela upp strömmen i dess rätta intervaller. Vid skapandet av en RTP-session anger man destinations IP och port, sedan kommer RTP automatiskt välja en port för RTCP. Därefter kommer ljud- och bildströmmarna skickas i separata sessioner så att mottagaren kan välja om han vill ta emot både eller bara en av dem. Denna separation av dataströmmar är användbar för att uppkopplingar med begränsad bandbredd, då användaren själv kan välja att inte spela den ena för att ge mer bandbredd till den andra.

2.4.2 Real-Time Control Protocol (RTCP)

Real-Time Control Protocol (RTCP) är en del av RTP protokollet och agerar kvalitetkontrollant och ger extra statistik som kan användas till att synkronisera överföringen[5]. Dessa paket skickas med ett par sekunders mellanrum och innehåller inte data tillhörande mediaströmmen. RTCP protokollet innehåller istället information, såsom medlemsinformation, synkroniseringsdata för mediaströmmar och kvalitetsåterkoppling. Varje deltagare i en RTP-session skickar en rapport med RTCP, vilket i stora grupper kan skapa mycket data. Detta motverkar protokollet genom att dynamiskt ändra de intervaller som dessa rapporter skickas i. Som standard använder RTCP 5% av bandbredden allokerad till sessionen.

2.4.3 Real-Time Streaming Protocol (RTSP)

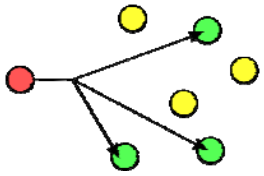
Real-Time Streaming Protocol (RTSP) är ett statefull streaming protokoll som öppnar sessioner mellan klient och server[6]. Genom dessa sessioner kan man kontrollera strömmar genom kommandon, som t.ex. stop eller paus. Protokollet i sig används inte till att skicka dataströmmar utan är enbart där för att kontrollera existerande strömmar från andra protokoll som t.ex. RTP. Tanken bakom detta protokoll var att skapa en ”fjärrkontroll” till streamingservern.

2.5 Multicast

Multicast är en nätverksteknik för att leverera samma dataström till flera klienter i ett nätverk, genom att skicka en dataström till en nätverkadress som klienterna kan lyssna på[7], [8]. Detta skapar extremt mycket mindre bandbreddsallokering än unicast men har större restriktioner för tillgänglighet. En multicastadress kommer normalt sätt enbart vara tillgänglig i det lokala nätverket eller AS-nätet, och brukar inte routas mellan nätverk då det kan skapa flooding.

Fördelar med denna teknik är att den enkelt kan nå ut till flera klienter utan att skicka mer än ett paket. Dessa paket skickas normalt sätt med UDP protokollet vilket i sin tur inte ger någon större säkerhet men hittills har inte detta ansetts ett problem då applikationer som använder denna teknik har utrymme för paketförluster och ses acceptabelt.

Då målet med multicasting (se Figur 3) är att få ut samma data till flera klienter utan att spendera onödig bandbredd, sker duplicering av trafiken hos routrar i nätet istället för servern. Det skapas även en ”optimal” passage genom nätverket och dess routers för att minimera fördröjningar i nätet och ge utrymme för övrig trafik. Dessa ser olika ut beroende på nätverkets struktur.



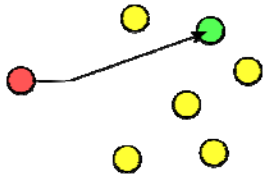
Figur 3: Enkel bild på multicast

För att skapa en multicastström skickas data till en speciell multicastadress. Dessa adresser har IP rangen 224.0.0.0 till 239.255.255.255 och är reserverade för enbart multicasting. När dataströmmen skapas och skickas till ett av dessa IP kan klienterna ansluta och lyssna med hjälp av IGMP protokollet.. När klienter ansluter till multicastkällan skapas ett distrubutionträd för multicastgruppen. Trädet skapas med hjälp av PIM (Protocol Independent Multicast) och ser till servern som skickar data till multicastgruppen kan nås av alla klienter i gruppen.

PIM använder sig av 5 olika tillvägagångssätt för att skapa detta träd, den vanligaste är Sparse-Mode. Träden kan innehålla ett oändligt antal klienter och är en av grundstenarna till dagens IPTV och video-on-demand tjänser.

2.6 Unicast

Unicast teknik skickar dataströmmar mellan källa och destination (se Figur 4) [9]. Fördelar med detta är att det inte finns några routingrestriktioner. Dock är denna teknik inte realistisk när det kommer till IPTV och liknande media, då den skulle kräva alldeles för mycket bandbredd. Ett användningområde där unicast använda är t.ex. internetradio. Denna tillämpning är dock kostsam då den generar höga bandbreddskostnader, eftersom varje lyssnare får sin egen ström från servern. Tyvärr kan multicasting tekniken inte implementeras p.g.a. routing restriktioner. Multicast skulle av denna anlendning även motsätta hela idén bakom internetradio, som är att man ska kunna lyssna var man än befinner sig.



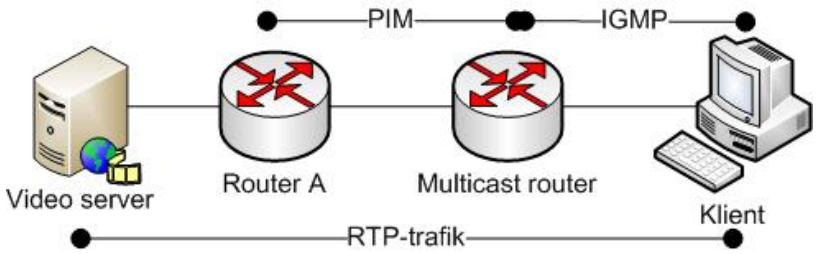
Figur 4: Enkel bild på unicast

2.7 Internet group management protocol (IGMP)

IGMP (se Figur 5) används för att upprätta medlemskap i multicastgrupper[10]. I vårt projekt använder vi oss denna teknik för att skicka videoströmmar mellan en server (streamer) och en klient (away).

IGMP är ett lager 3-protokoll som kapslas in i IP-paket vid transport. Det som är speciellt för denna typ av protokoll är att det inte garanterar någon typ av säkerhet utan arbetar efter ”best effort”-principen. Med best effort menas att den inte kan garantera att alla paket kommer fram till alla medlemmar i grupp eller att de kommer fram i rätt ordning. Genom denna funktion är protokollet skalbart och kan användas i små eller stora nätverk.

Protokollet i sig fungerar så att klienten skickar ett IGMP ”join” meddelande för att gå med i multicastgruppen. Meddelandet vidarebefordras av närmaste multicast router som då börjar vidarebefordra strömmen till den nya medlemmen. När klienten sedan skickar ett ”leave” meddelande till routern upphör routern med att forwarda multicasttrafiken till klienten.



Figur 5: Bild på IGMP-system

2.8 IGMPproxy

IGMPproxy är ett verktyg utvecklat till linux, freeBSD, Solaris och liknande operativsystem[11]. Programmet används för att vidarebefordra IGMP paket då detta är som standard avstängt i alla routers.

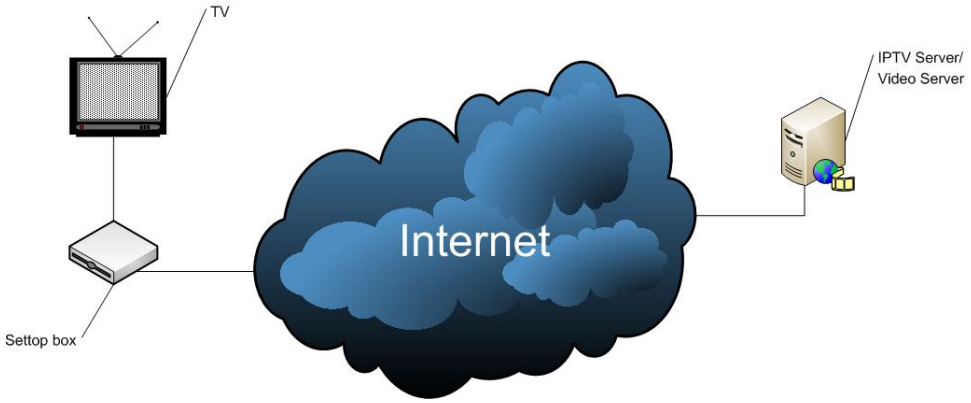
IGMPproxy körs i kernel mode och aktiverar multicast forwarding vilket är ett måste för att dessa paket ska skickas vidare. Sedan lägger sig mjukvaran och lyssnar på det interface man specificerat i konfigurationsfilen efter join och leave requests från klienter och skickar dessa vidare till servrar som handhåller multicastströmmen. När multicastservern sedan skickar strömmen kommer IGMPproxy att vidarebefordra denna ström till klienten.

IGMPproxy är en vidareutveckling av PIM-d, mrouted och smcroute då dessa projekt avslutat för ett flertal år sedan och saknar därför support för nyare system och hårdvara.

2.9 IPTV

IPTV är TV över IP-protokollet, IPTV är en relativt ny tjänst som ökar i popularitet[12]. Vad detta innebär är att man skickar videosändningarna över internet från streamingservrar till kunden. Till skillnad från dagens TV kan denna teknik erbjuda ett annat sätt att se på TV, IPTV-tekniken ger kunden en möjlighet att välja om den vill se på TV på beställning där kunden själv kan välja vad och när han/hon vill se det. Eller det traditionella sättet där programleverantörerna konstant skickar ut programmen utan att kunden kan påverka dess innehåll.

Det finns två sätt att ta emot denna teknik. Det första alternativet är att man har en SetTop Box som kopplas in mellan bredbandsmodemet och TVn (se Figur 6). Denna Box avkodar dataströmmarna som kommer från IPTV-leverantören och skickar dem vidare till din TV. Alternativ två är att fånga upp strömmen direkt med hjälp av en dator och använda mediamjukvara för att visa dataströmmarna på skärmen.



Figur 6: Visar förenklad version av IPTV med settop box

Idagsläget kräver en vanlig dataström kodat med mpeg-4 mellan 4-6 Mbit, detta kan man minska till 1-2.5 Mbit om man komprimerar dataströmmen med en nyare mediakodec som h.264. Då tillgången på höghastighetsuppkopplingar har ökat så har IPTV gått från att vara något som enbart erbjudits i fibernät till att kunna levereras till alla med en DSL uppkoppling. Ur en leverantörs perspektiv är detta en mycket fördelaktig teknik om de har bandbredden att avsätta. Tekniken använder multicast för att skicka datatrömmarna vilket menas att enbart en dataström per kanal oavsett hur många kunder som tittar på kanalen. Detta ger tekniken mycket bra skalbarhet i stora nät där leverantören kan skicka 50 kanaler i 4Mbps och enbart behöva avsätta 200Mbps. Om leverantören vill erbjuda video on demand kommer denna behöva avsätta ytterligare Mbps beroende på antal kunder som beräknas använda denna tjänst. I framtiden kommer detta troligtvis vara ett mindre problem då nya tekniker inom peer-to-peer utvecklas där kunderna själva kan hjälpa till genom att ta emot, lagra och skicka vidare data till andra i nätverket.

2.10 VLC media player

VLC är ett gratis mediaverktyg som kan användas till att streama media till en multicast- eller unicastadress[13]. Det är ett mycket avancerat program som innehåller de flesta av dagens populära video- och ljudCODEC:n. Med hjälp av detta program kan man skicka, modifiera och manipulera dataströmmar efter behag och bandbreddsresurser. Programmet är utvecklat under GNU license av ett par studenter

vid Ecole Centrale Paris. Flera har hjälpt att utveckla detta mediaverktyg som först var tänkt att vara ett streamingverktyg för större nätverk. Programmet har sedan sin start växt till ett av marknaden mest använda mediaspelare och finns idag till de flesta operativsystem.

2.11 Virtuellt Privat Nätverk (VPN)

VPN är en teknologi som används för att skapa säkra förbindelser mellan två punkter över ett publikt nätverk, såsom Internet. Som namnet antyder är poängen att informationen i detta nätverk ska vara privat, och endast tillgänglig för auktoriserade parter. Man säger att en tunnel skapas mellan de olika punkterna, och all information som skickas i denna tunnel är krypterad.

Figur 7: Visar en VPN-tunnel över internet

VPN är väldigt åtråvärd lösning i livet då t.ex. ett företag har anställda som jobbar hemifrån och behöver komma åt resurser på företaget. VPN erbjuder just detta, och säkerhet dessutom. Beroende på vad ens egna behov är, så kan valet av VPN-lösning vara en ganska komplicerad procedur, då det finns väldigt många för- och nackdelar med de olika teknikerna beroende på vad du är ute efter. I grund och botten levererar de samma tjänst, men mellan alla olika tekniker och specifika ändamål så borde tid definitivt läggas ner på att välja rätt lösning. Nedan går vi igenom de vanliga VPN-teknikerna och jämför dem sett från en säkerhets-, prestanda- och användarvänlighetsynvinkel.

2.11.1 OpenVPN

OpenVPN är ett opensource VPN- program, som används för att skapa en SSL/TLS-krypterad tunnel mellan två eller flera punkter baserat på flexibla autentiseringsmetoder såsom certifikat, smart cards, och/eller användarnamn/lösenord. Detta tillämpas på OSI-modellens lager 3 (routing) över UDP eller TCP på porten 1194. Däremot har OpenVPN förmågan att även tunnla all trafik från lager 2 till slutpunkten, såsom ARP-trafik. Möjligheten ges också att lägga till ”access control”, alltså grupp- eller användarpolicies genom brandväggsregler som appliceras på det virtuella VPN interfacet.

Utvecklarna har utgått från den fundamentala konceptet att komplexitet är säkerhetens fiende, och erbjuder en kostnadseffektiv, säker och enkel lösning.

2.11.1.1 Kryptering - SSL/TLS

OpenVPN använder sig av säkerhetsstandarden SSL/TLS. Secure Sockets Layer (SSL) är ett universiellt accepterat sätt att kryptera och autentisera kommunikationen mellan flera parter. Utvecklades först av Netscape för internt bruk, och den första publika versionen (2.0.) släpptes i Februari 1995. Denna version innehöll en massa säkerhetsbrister, vilket ledde till att version 3.0 släpptes redan nästa år. Denna version verkade som bas för det nya protokollet Transport Layer Security (TLS), som Internet Engineering Task Force (IETF) först definierade i Januari 1999.

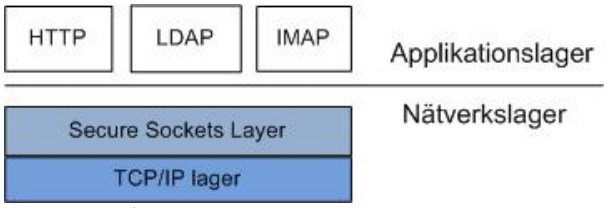
SSL/TLS är kryptografiska protokoll som designats för att skydda kommunikationen i nätverket från eventuella attacker, såsom avlyssning och att folk utger sig för att vara någon de inte är. SSL/TLS krypterar segment av nätverk på Transport Lagret end-to-end.

SSL/TLS är två olika protokoll, men i själva verkat är SSLv3 och TLSv1 i stort sett samma protokoll. Man brukar kalla TLSv1 för SSLv3.1 Skillnaden mellan de två hittar vi i RFC för TLSv1 RFC 2246:

"The differences between this protocol and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate (although TLS 1.0 does incorporate a mechanism by which a TLS implementation can back down to SSL 3.0)."[26]

TCP/IP-protokollet styr transporten och routingen över internet. Andra protokoll, t.ex. HyperText Transport Protocol (HTTP), Lightweight Directory Access Protocol (LDAP), eller Internet Messaging Access Protocol (IMAP) körs ovanpå TCP/IP i den bemärkelsen att de alla använder TCP/IP för att utföra sina applikationers uppgifter, såsom webbrowsning eller köra en e-mail server.

SSL körs över TCP/IP och under protokoll såsom HTTP eller IMAP. Vilket innebär att det använder sig av TCP/IP på uppdrag av protokoll på högre nivå. T.ex. vid säker surfning (HTTPS) så använder sig HTTP-protokollet av SSL, som i sin tur använder sig av TCP/IP för att transportera datan.



Figur 8: Bild på olika protokoll och vilket lager de jobbar på

SSL använder sig av kryptering med både publika och symmetriska nycklar för att göra anslutningen säker. En session inleds alltid med en så kallad handskakning mellan klienten och servern. Denna handskakning börjar med att servern autentiserar sig för klienten med hjälp av publika nycklar. Nästa steg är att klienten och servern kommer överrens om de parametrar som behövs för att kommunicera över SSL, såsom cipher, SSL version, session-speficik data med flera. Denna procedur skapar de symmetriska nycklar som behövs för kryptering och dekryptering av den data som skickas/tas emot.

2.11.1.2 Pre-shared static key

Här skapar man helt enkelt en statisk nyckel som sedan används för att kryptera/dekryptera all information som skickas över tunneln. Uppenbart här är ju att ett stort plus för denna sorts upplägg är att det är väldigt enkelt. Samtidigt så inses snabbt att detta förstör hela konceptet med ”perfect forward secrecy”, vilket innebär att om någon obehörig kommer över din nyckel så är all data som någonsin krypterats med den nyckeln äventyrat.

En statisk nyckel genereras med ett enkelt kommando. Exempel på resultatet visas nedan:

```
-----BEGIN OpenVPN Static key V1-----
e5e4d6af39289d53
171ecc237a8f996a
97743d146661405e
c724d5913c550a0c
30a48e52dfbeceb6
e2e7bd4a8357df78
4609fe35bbe99c32
bdf974952ade8fb9
71c204aaf4f256ba
eeda7aed4822ff98
fd66da2efa9bf8c5
e70996353e0f96a9
c94c9f9afb17637b
283da25cc99b37bf
6f7e15b38aedc3e8
e6adb40fca5c5463
-----END OpenVPN Static key V1-----
```

2.11.1.3 Certifikat

I denna lösning använder vi oss av RSA-certifikat och nycklar. Här måste man skapa egna RSA certifikat med hjälp av OpenSSL som är inkluderat med OpenVPN-paketet, alternativt låta en certifikationsauktoritet (CA) utfärda certifikat. För en vanlig användare är det enklast att skapa egna certifikat, då alternativet tar tid och kostar pengar. Till skillnad från pre-shard secret key-lösningen så erbjuder denna lösning mer säkerhet till priset av högre komplexitet.

Ett RSA certifikat är en publik nyckel i form av ett eletroniskt dokument som innehåller en digital signatur för att verifiera en identitet. En sorts elektronisk legitimation för att verifiera att användaren verkligen är den han påstår sig vara. Tillsammans med den digitala signaturen så finns även diverse information om identiteten bakom certifikatet. Stora organisationer har ofta certifikat utfärdade av en CA, detta så att användare ska känna sig säker på att certifikatet är okej.

2.11.1.4 Användarnamn och lösenord

I den senaste versionen (2.0) av OpenVPN så finns även möjligheten att autentisera sig med hjälp av användarnamn och lösenord[14], [15], [16], [17], [18]. Detta används oftast i kombination med certifikat för att lägga på ännu ett lager av säkerhet, men kan även användas separat. Detta är såklart inte att rekommendera från en säkerhetssynvinkel, men är fortfarande en möjlighet.

När man använder sig av enbart av användarnamn och lösenord så slipper klienterna använda sig av egna certifikat och nycklar, men inte root server certifikatet (ca.crt). Filen används fortfarande för att kunna verifiera serverns certifikat. Detta förenklar själva användandet och administrationen, men försämrar som sagt säkerheten dramatiskt.

2.11.2 IP Security (IPsec)

Internet Protocol Security (IPsec) är ett tvåläges, end-to-end, ramverk som med hjälp av autentisering och kryptering av varje IP-paket skapar en säker dataström mellan:

- två hosts
- två nätverk
- ett nätverk och en host

IPsec är en efterträdare till protokollet swIPe **[**19**]**, och är officiellt specificerad av IETF i en serie av Request for Comments (RFC).

En stark egenskap hos IPsec är att det jobbar i nätverksskitet (layer 3) av OSI-modellen. Vad detta innebär är att IPsec kan användas för att säkra vilken applikationstrafik som helst, över både ipv4 och ipv6. Till skillnad från några populära säkerhetsprotokoll såsom SSL/TLS och Secure Shell (SSH). Dessa protokoll jobbar i de övre lagren av modellen, vilket innebär att applikationen måste vara utvecklad med dem i åtanke.

IPsec är ett protokoll, eller snarare en svit av protokoll, som tillåter i stort sett vilka inställningar som helst så länge båda parterna går med på det.

IPsec består av de två säkerhetsprotokollen Authentication Header (AH) och Encapsulation (ESP), samt protokollsviten Internet Key Exchange (IKE). AH och ESP erbjuder integritet, autentisering, kryptering, återspelningsskydd och begränsad trafikflödeskryptering. För att dessa mål skall uppnås så kan antingen ESP användas separat, eller tillsammans med AH.

Tabell 1: Vad AH och ESP erbjuder för tjänster

	AH	ESP (endast kryptering)	ESP (kryptering och autentisering
Integritet	x		x
Autentisering	x		x
Återuppspelningsskydd	x	x	x
Kryptering		x	x
Begränsad trafikflödes kryptering		x	x

2.11.2.1 Authentication Header (AH)

AH är ett av de två huvudprotkollen som tar hand om säkerheten i IPsec. AH tillhandahåller autenticering av antingen hela, eller delar av paketet genom att lägga till en ny header.

Figur 9: Ett IP-paket efter AH lagt till sin header

Efter det att den nya headern räknats ut och lagts till så signeras hela paketet av AH för integritetens skull. Vissa fält som kan tänkas ändras på vägen till sin destination såsom t.ex. TTL (Time to Live) signeras dock inte.

AH använder sig av en funktion som är likvärdig till en checksumma. Inom nätverk, innan en datalast skickas iväg så räknas det med hjälp av en standardalgoritm ut en checksum eller en Cyclic Redundacy Check (CRC)-kod med hjälp av datan som skall skickas. Denna kod skickas sedan tillsammans med originaldatan till mottagaren, som i sin tur upprepar beräkningen. Stämmer inte checksumman som mottagaren räknat ut med den som skickades med från avsändaren så slängs paketet.

Detta är grundidéen bakom AH, men istället för att använda en enkel algoritm som alla känner till så används en speciell hashing-algoritm och en hemlig nyckel som bara är kända av avsändaren och mottagaren. Avsändaren använder data som skall skickas och räknar ut en så kallad Integrity Check Value (ICV) och placerar den i en ny header. Mottagaren gör samma uträkning med hjälp av de två nycklarna som båda känner till, och därefter ser mottagaren direkt om något fält blivit modifierat (antingen av fel som uppstått på vägen eller av illvilja)

Uppgifterna AH har, är alltså att garanterar att data kommer fram oförändrade (integritet), signerar data så att avsändaren kan verifieras (autenticitet) och oberoende av hur många gånger ett paket skickas så bearbetas det endast en gång (återuppspelningsskydd).

Värt att notera är att beräkningen av ICV inte ändrar orginaldatan i paketet. AH-headern tillåter oss att verifiera datans integritet, men den krypterar den inte.

2.11.2.2 Encapsulating Security Payload (ESP)

Det andra säkerhetsprotokollet i IPsec är ESP. För många är endast verifiering av paket inte tillräckligt. Ofta vill man inte bara se till att datan kommer fram ordentligt, utan även hindra obehöriga personer från att komma åt och läsa den. Här kommer ESP in i bilden och erbjuder precis det: Kryptering.

En krypteringsalgoritm kombinerar datan i paketet med en nyckel och skickas iväg till destinationen, som i sin tur använder samma algoritm för att återställa paketet till dess ursprungliga form. ESP erbjuder även egen autentisering som det som används i AH, alternativt så kan det användas tillsammans med AH.

Till skillnad från AH som bara lägger till en ny header, så lägger ESP till tre nya fält:

- **ESP Header:** Innehåller Security Parameter Index (SPI) och Sekvensnummer, och kommer innan den krypterade datan.
- **ESP Trailer:** Innehåller padding och next header fältet, och placeras efter den krypterade datan.
- **ESP Authentication Data:** Innehåller en liknande ICV som AH använder, men enbart när ESP egna autenticering används.

Figur 10: Visar ett paket efter ESP redo för transport

Det finns två anledningar till varför det ser ut såhär. För det första så kräver några krypteringsalgoritmer att datan som ska krypteras har en viss blockstorlek, så paddingen måste komma efter och inte före datan. Andra anledningen är att ICV:n används för att autentisera resten av det krypterade paketet efter krypteringen är gjord. Det betyder att den inte kan finnas i ESP headern eller trailern.

2.11.2.3 Internet Key Exchange (IKE)

Syftet med IKE är att tillåta utbyte av den information som krävs för att säker kommunikation mellan två parter ska kunna upprättas. Som många säkra nätverksprotokoll är även IPsec baserat på konceptet “shared secret”. Två parter som vill skicka information sinsemellan krypterar och dekrypterar den med hjälp av en

”delad hemlighet” som bara de känner till. En person som inte känner till denna hemlighet kan snappa upp trafiken, men antingen inte läsa den (om ESP används för att kryptera datalasten), alternativt inte manipulera den (om AH används). Innan AH eller ESP kan användas så är det nödvändigt att parterna känner till den delade hemligheten som säkerhetsprotokollen kommer använda. IKE tillåter enheter att byta så kallade Security Associations (SA). En SA kan ses som ett en-vägsförhållande mellan sändare och mottagare som innehåller egenskaper som i sin tur sedan används för att skicka krypterad data med hjälp av ESP eller AH. En SA kan innehålla inställningar såsom villken typ av kryptering, hash algoritm och autentiseringsmetod som skall användas.

IKE räkas som ett hybridprotokoll, då det kombinerar och använder sig av funktioner från tre olika protokoll. Internet Security Association and Key Management Protocol (ISAKMP) är det första protokollet (som egentligen kan ses som ett ramverk), och används för utbyte av nycklar som används för kryptering samt förhandlig av SAs genom en serie av faser. De andra två protokollen (som finns inom ISAKMP ramverket) är OAKLEY och SKEME. OAKLEY står för den större delen av nyckel-utbytesprocessen, medan SKEME står för bland annat krypteringen av publika nycklar samt dess snabba nyckeluppdatering.

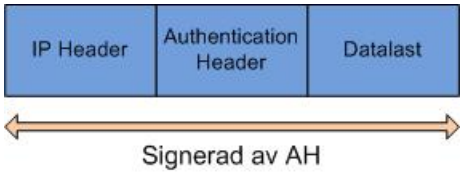
IKE processen kan delas in i följande två faser:

- **Fas 1:** I denna fas så kommer de två enheterna överrens om hur de ska skicka ytterligare information på ett säkert sätt med hjälp av den kända Diffie-Hellman algoritmen för att generera en delad nyckel för framtida kommunikation.. Detta skapar en SA för ISAKMP själv, en så kallad ISAKMP SA.
- **Fas 2:** I den andra fasen så används ISAKMP SA:n som skapades i första fasen till att skapa SAs för andra protokoll. Det är här parametrarna för AH och ESP förhandlas.

2.11.2.4 De två lägena

IPsec har som tidigare nämnt två olika lägen[20], [21], [22], [23], [24], [25]. Dessa två lägen är Transport- och Tunnelläget. Vad som skiljer de två åt är hur de behandlar paket, med hjälp av de två säkerhetsprotokollen AH och ESP, innan de skickas iväg.

Transportläget är standardvalet i IPsec, och används för kommunikation mellan två hosts. I transportläget är det enbart själva datalasten av IP-paket som autentiseras och/eller krypteras. IP headern modifieras eller krypteras inte, vilket innebär att routingen är intakt.

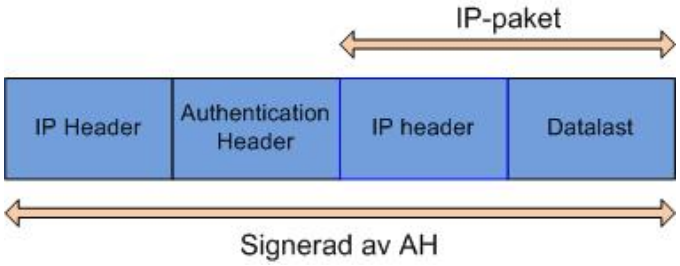


Figur 11: Ett paket signerat av AH i transportläget

Detta medför att NAT:ning görs omöjlig, och är anledningen till att transportläget endast fungerar mellan två hosts. Denna uppgift anförtros till tunnelläget.

I tunnelläget skapas en virtuell ”tunnel” och används mellan två gateways, en host och en gateway eller mellan två hosts. Tunnelläget krypterar inte bara datalasten

utan även IP headern. IP-paketet behandlas som en AH/ESP datalast, och krypteras med en ny IP header. IP-adresserna i den nya/yttre headern är tunnelns ändpunkter, medan adresserna i den inkapslade IP headern är det slutgiltiga käll- och destinationsadresserna.



Figur 12: Ett paket signerat av AH i tunnelläget

IPsec erbjuder användaren integritet, autenticitet, återuppspelningsskydd och kryptering. För att uppnå detta så använder sig IPsec av de två säkerhetsprotokollen AH och ESP.

3 Tillvägagångssätt

3.1 Konstruktion

Grundidéen med konstruktionerna är att lokalt simulera en miljö så pass lik internet som möjligt. Projektet är baserat på fyra datorer, varav tre stycken kör openSUSE 11.1 och en (away) kör Windows XP. Away kan däremot lika gärna köra linux med några små ändringar av openVPN konfigurationen för away. Vi har dock valt att köra Windows XP och vår lösning utgår således från att away är en XP-maskin.

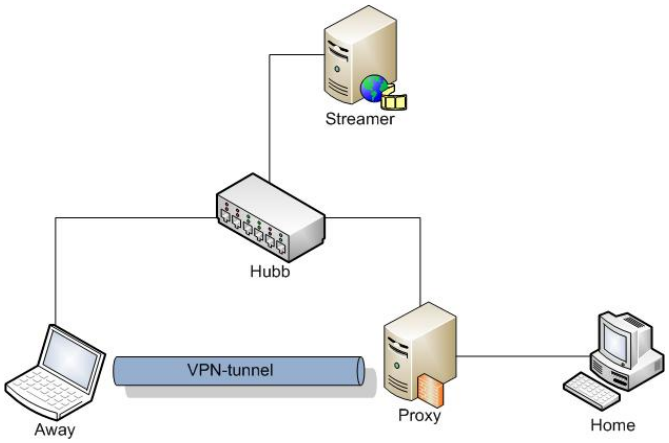
- **Router –** Dator som agerar en router ute på internet vars uppgift är att hitta rätt väg för paket och skicka dem vidare.
- **Away –** Simulerar vår bärbara dator som vi har med oss var vi än åker, och som skall kopplas ihop genom en VPN-tunnel med proxyn (hemmanätverket)
- **Streamer –** En dator som vi har installerat VLC på som video streamas ifrån. Detta simulerar vår IPTV-leverantör.
- **Proxy –** Denna dator agerar VPN-server och proxy. Det vill säga att den inte begär något själv, utan bara i uppdrag av någon annan som den sedan skickar vidare.
- **Home –** Fiktiv dator som symboliserar en klient i det lokala nätverket.

Under projektets gång gjordes kontinuerligt små förändring i nätverket. Nätverkskort flyttades mellan datorer, kopplingar ändrades, hårdvara bytte funktion m.m. Men de tre modellerna nedan är de koncept vi arbetat med.

I de tre nedanstående modellerna hade vi en sak genensam för att komma ifrån det ”lokala nätverket”. Med detta menas att datorerna har separerats in i tre olika nät, med proxyn inkopplad i alla tre näten. Idén var att datorerna inte skulle ha direkt kontakt med varandra, utan att de var tvungna att gå genom den centrala punkten som agerade ”internet”.

3.1.1 Version 1

Figur 13 visa den första konstruktionen av nätverket.



Figur 13: Nätskiss v.1

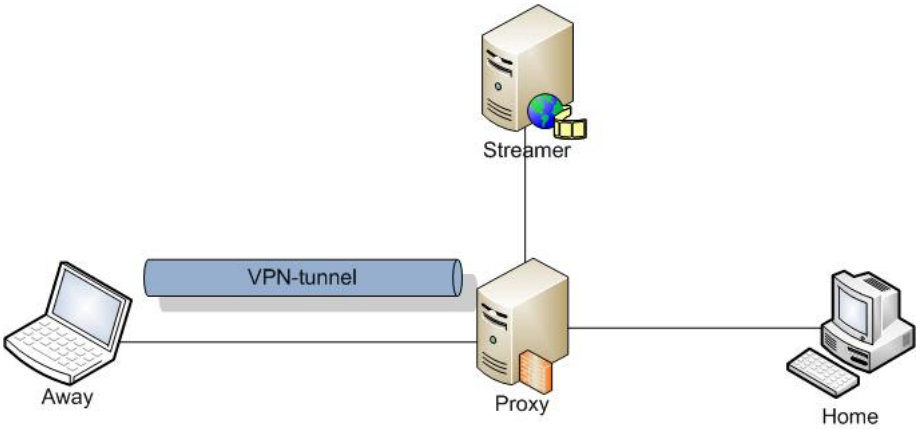
Som bilden visar så kopplades datorerna in i hubben. OpenVPN-server installerades på proxyn, som i sin tur away kopplade upp sig till. Felet med denna konstruktion är givetvis att vi har en hubb, istället för en switch. All data som kommer in till hubben skickar den vidare till resterande portar. Detta gav oss problemet att

away såg streamerns trafik direkt, istället för genom VPN:et vilket inte gick att ändra pga att hur routing fungerar, dvs minst antal hop vinner. Den här konstruktionen skrotades ganska snabbt, då vi insåg att hubben var den svaga länken.

Idéen med konstruktionen var att hubben skulle simulera internet. Men som vi nämnt ovan är detta inte möjligt pga hur en hubb fungerar, till skillnad från en switch eller router.

3.1.2 Version 2

I nästa struktur på nätverket togs hubben bort helt. Här agerade proxyn ”internet” och kopplade samman alla andra nätverk.

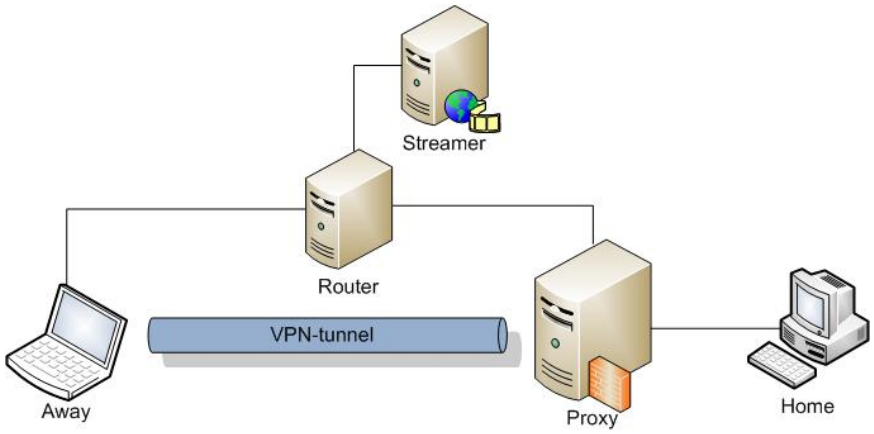


Figur 14: Nätskiss v.2

Här installerades igmpproxy på proxyn för att vidarebefordra MC-meddelanden från away till streamern. Detta var den första skissen som funkade som vi ville, dvs vi kunde skicka igmp-meddelanden från away, genom VPN-tunneln till proxyn, som i sin tur skickade vidare förfrågningen till streamern. Streamern började i sin tur skicka paket till proxyn, som sedan skickade vidare dataströmmen till away.

Som sagt så uppfyllde denna skiss vår uppgift, men återspeglade inte hur det skulle se ut om vi byggde lösningen med internet i åtanke. Eftersom proxyn var direkt kopplad till streamern så blev den floodad av paket även om away hade stängt ner dataströmmen, eller ens öppnat den. Denna trafik skulle inte ses av vår proxy i ett riktigt nät, vilket gav upphov till den nedanstående och slutliga lösningen.

3.1.3 Version 3



Figur 15: Nätskiss v.3

I den här versionen så har vi implementerat en router mellan streamern och proxyn, för att återskapa en så snarlik miljö som möjligt till internet. Om vi tittar på hur internet är uppbyggt, och jämför det med vår lösning så ser vi snabbt att detta ser väldigt likt ut. Routerns uppgift i denna modell är att simulera en router ute på internet som skickar vidare streamerns paket

3.2 Applikationer

Målet med projekten har alltid varit att göra en så användarvänlig, kostnadsfri, säker och enkel lösning som möjligt. Detta har vägt tungt och varit avgörande i alla val vi gjort angående programvaran som använts i projektet.

3.2.1 Brandvägg

OpenSuse kommer med en inbyggd brandvägg som administreras genom yast. Denna brandvägg skapade stora problem och tvingade oss starta om proxy servern vid ett flertal tillfällen då brandväggen inte uppdaterade korrekt. Detta ledde till av vi valde att permanent stänga av den. Istället för openSuse branväggen kan IPtables programvaran användas till att filtrera trafik. Det som är viktigt ifall en brandvägg ska vara påslagen är att port 1194 (UDP/TCP) öppnas så att trafiken för VPN-tunneln släpps igenom.

3.2.2 VPN-teknik

Här stod valet egentligen mellan två olika tekniker: OpenVPN och IPsec.
IPsec är en komplicerad protokollsvit, och det finns otroligt många sätt att skapa en lyckad förbindelse på. Desto djupare vi kom i IPsec träsket, så insåg vi mer och mer vilka möjligheter man har med IPssec, samtidigt insågs också att det kanske lämpar sig mer för någon som är mer insatt i nätverk och vet exakt vad de vill ha. Då vi vill att även nybörjare ska kunna använda vår lösning så kom vi fram till att IPsec inte var för oss, och gick vidare till OpenVPN, som vi hade hört var en enklare metod att få igång ett VPN. Efter lite undersökning kom vi till insikten att OpenVPN uppfyller alla de mål vi ställt på projektet och val av VPN-programvara var klar.

OpenVPN kan liknas en glass med en enda kula, medan IPsec är en glass med flera tiotals kulor. Glassen med en kula är tillräckligt mycket så att det fortfarande är gott, medan den större glassen blir för mycket. Det är gott, om man klarar av att äta allt, men för de flesta är det alldeles för mycket.

3.3 Konfiguration

Nuförtiden kan man hitta det mesta på Google, så även detaljerade steg-för-steg guider hur saker och ting ska installeras. OpenVPN är så generösa (läs: användarvänliga) att även de ger oss en väldigt detaljerad (om än lite för detaljerad för gemene man) how-to hur man får igång OpenVPN. Vi har använt denna som grund när vi installerat OpenVPN och nedan följer exakt hur vi har gjort, även om det är långt ifrån det enda sättet:

3.3.1 Proxy installation

Eftersom huvudsyftet med projektet var att streama media så började vi med att ladda ner och installera LZO (Lempel-Ziv-Oberhumer). LZO är ettt datakomprimeringsbibliotek som är lämplig för data de-/komprimering i realtid. Detta

innebär att det gynnar hastighet över kompressionsförhållande, vilket är en bra egenskap när det gäller streaming/VPN.

 # wget http://www.oberhumer.com/opensource/lzo/download/lzo-2.03.tar.gz
 # tar –xzvf lzo-2.03.tar.gz
 # cd lzo-2.03
./configure
make
make check
make test
make install

Efter det så laddade vi ner källkoden av OpenVPN från deras officiella sida (openvpn.net) och fortsatte med att installera det.

 # wget http://www.openvpn.net/release/openvpn-2.0.9.tar.gz
 # tar -xzvf openvpn-2.0.9.tar.gz
 # cd openvpn-2.0.9
 # ./configure
 # make
 # make install

När VPN-servern är installerad är det dags för att generera certifikat. Om du planerar att generera flera nycklar (flera klienter) så rekommenderas att editera easy-rsa/vars filen för att göra processen snabbare vid generation av nycklarna.

 # cd openvpn-2.0.9/easy-rsa
 # vi vars

Väl inne i vars så är det följande parametrar som skall ändras:

export KEY_DIR=$D/keys # Nycklarna skapas i easy-rsa/keys/

export KEY_COUNTRY= # Ange land
export KEY_PROVINCE= # Ange provins
export KEY_CITY= # Ange stad
export KEY_ORG=”” # Ange organisation
export KEY_EMAIL=”” # Ange e-mail

När vars är editerad är det dags för nästa steg. Spara och stäng ner.

Nu är det dags att sourca vars-filen och skapa ett certifikat och nyckel för vår Certificate Authority (CA).

 # . ./vars
 # ./clean-all
 # ./build-ca

```
Generating a 1024 bit RSA private key
.............++++++
...........++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

```
-----
Country Name (2 letter code) [SE]:
State or Province Name (full name) [Sthlm]:
Locality Name (eg, city) [Sthlm]:
Organization Name (eg, company) [Test]:
Organizational Unit Name (eg, section) [Test]:
Common Name (eg, your name or your server's hostname) []:Proxy
Email Address [me@myhost.mydomain]:
```

Här kommer det vi skrev in i vars till nytta. Det som står inom [] i frågorna ovan är det vi skrev in där, och är standard svaret. Dvs att vi bara kan trycka på enter istället för att skriva in något godtyckligt (vilket kan vara skönt om flera nycklar skall skapas som sagt). Common Name är det enda som måste skrivas in och borde vara namnet på datorn som certifikatet är till.

Nu ska vi generera nycklarna för servern och klienterna. Vi går igenom samma procedur med Country Name, State or Province Name etc som ovan för både server och klienter.

 # build-key-server server # Detta komma generera server.crt och .key
 # build-key klient # Genererar klientens certifikat och nyckel

Nu har vi skapat alla cerfitikat och nycklar. Det enda som återstår är att generera en Diffie-Hellman (DH) nyckel.

 # ./build-dh

```
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
.................+.........................................
.................+.............+.................+.........
.....................................
```

Nu när alla nycklar är skapad så är det enda som återstår att föra över nycklarna till respektive maskin. Nedan är en lista på alla filer som vi ska ha genererat ovan, deras syfte, om de är hemliga nycklar och vilken dator som skall ha dom. Notera att de hemliga nycklarna inte borde hanteras över medium som är känsliga för avlyssning, t.ex. nätverk.

Tabell 2: Visar nycklar och certifikat som genererats

Filnamn	Behövs av	Syfte	Hemlig?
ca.crt	Server + alla klienter	Root CA certifikat	Nej
ca.key	Nyckelsignerande maskin	Root CA nyckel	Ja
dh{n}.pem	Server	Diffie Hellman parametrar	Nej
server.crt	Server	Server Certifikat	Nej
server.key	Server	Server Nyckel	Ja
klient.crt	Klient	Klient Certifikat	Nej
klient.key	Klient	Klient Nyckel	Ja

Nu borde OpenVPN-servern vara färdigkonfigurerad och acceptera anslutningar. Vi testar att starta servern med konfigfilen /etc/openvpn/proxy.conf.

 # openvpn /etc/openvpn/server.conf

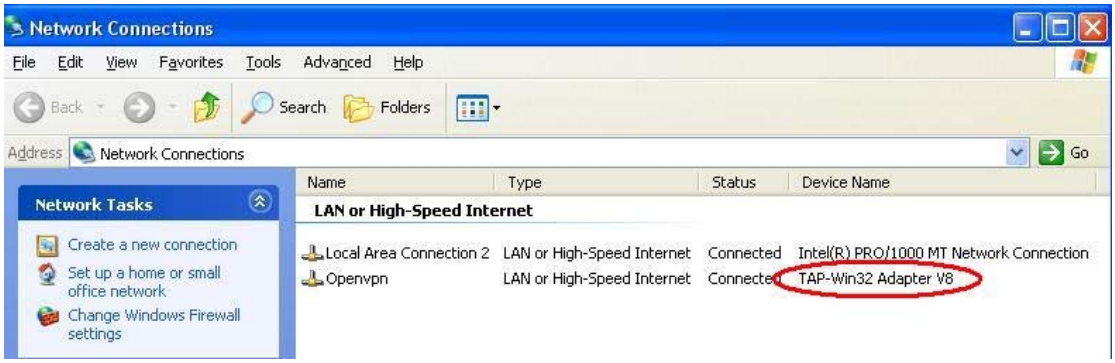
Figur 16: Visar start av korrekt konfigurerad OpenVPN-server

3.3.2 Klienten

Nu när servern är installerad så behöver vi installera och konfigurera klienten. Ladda ner OpenVPN-binären med ett användarvänligt GUI från: http://www.openvpn.se/files/install_packages/openvpn-2.0.9-gui-1.0.3-install.exe

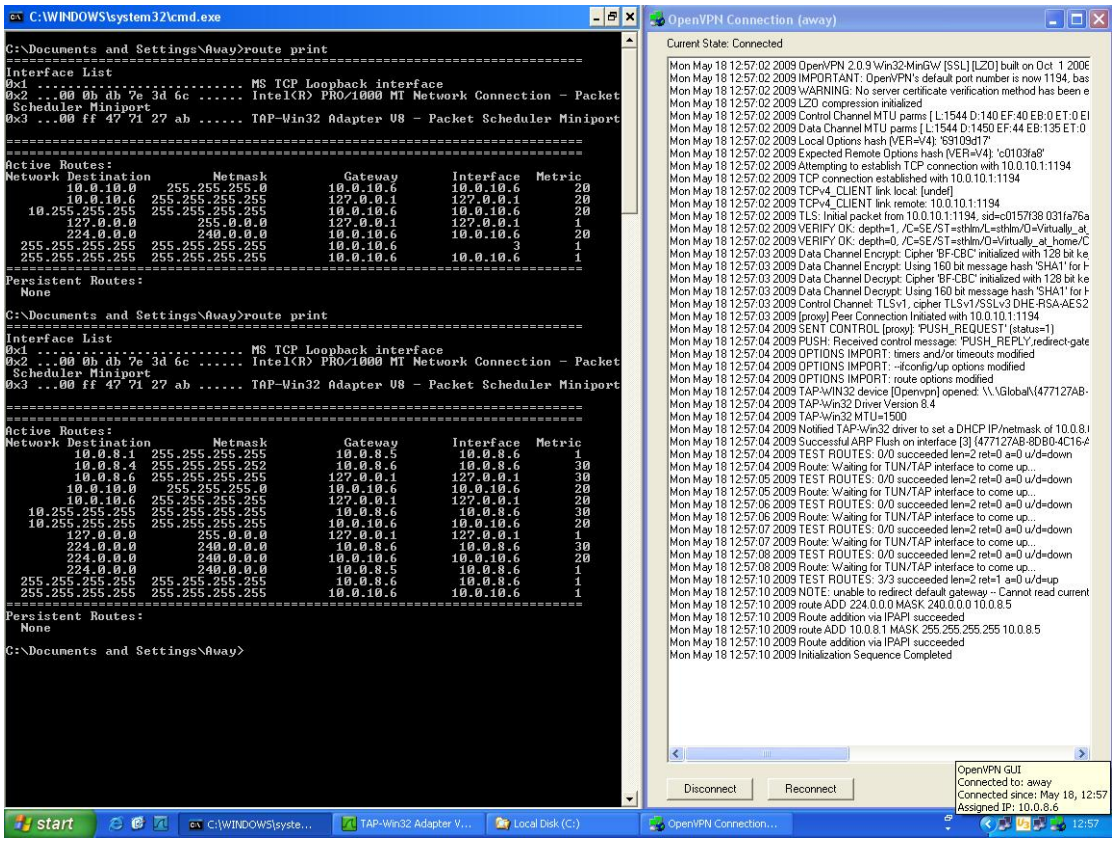
Följ installationsanvisningarna och installera i C:\Program files\OpenVPN. Skapa sedan en katalog vid exempelvis namn keys under OpenVPN katalogen och lägg däri de filer som specificeras av figur 19 som tillhör klienten.
Kopiera därefter konfigurationen till klienten (**appendix X**) och spara som namn.ovpn i C:\Program files\OpenVPN\config\. Se till att nycklar och certifikat ligger i samma katalog som specificerats i konfigurationen.

Installationen av OpenVPN skapar ett virtuellt interface som agerar termineringspunkt för klientens sida av VPN-tunneln. Detta interface är lätt att identifiera då enhetsnamnet är ”TAP-Win32 Adapter V8” och av logiska anledningar döpt till Openvpn.



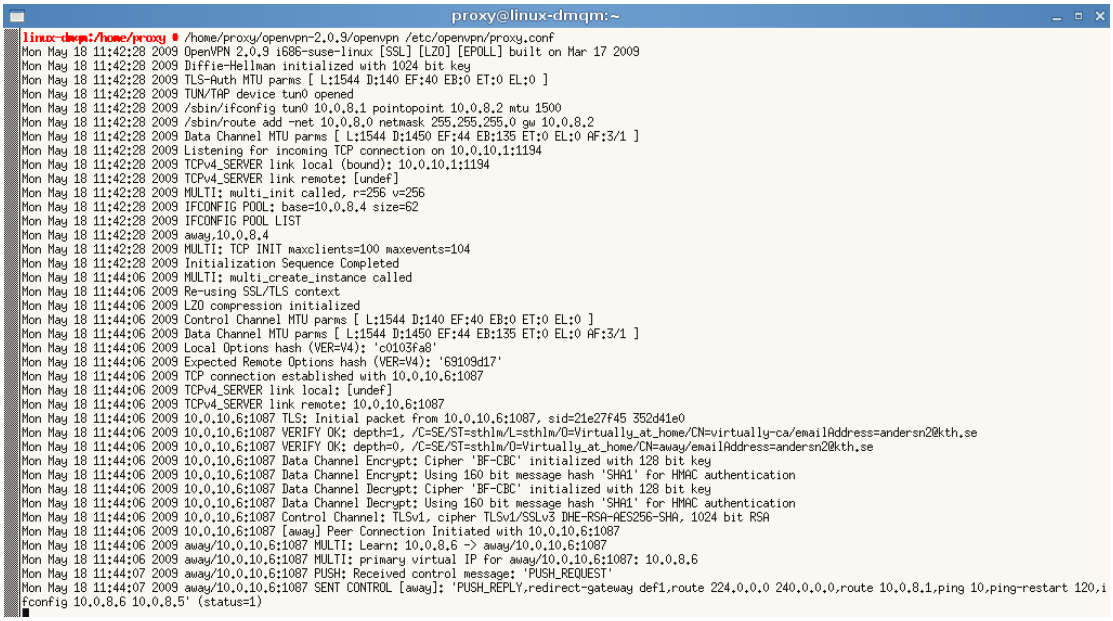
Figur 17: Visar det virtuella interface som installationen av OpenVPN lagt till

När detta är gjort så kan vi testa om det funkar. Starta OpenVPN GUI och högerklicka på OpenVPN GUI:s ikon nere i aktivitetsfältet och tryck ”Connect”. Om allt funkar som det ska borde den högra delen av bilden nedan visas.



Figur 18: Visar till vänster routes i Windows för och efter anslutning är gjord till OpenVPN-servern. Till höger visas OpenVPN GUI efter en lyckad anslutning

När en lyckad anslutning är gjord från klienten genom VPN:et så ser det ut såhär på servern.



Figur 19: Visar en lyckad anslutning genom VPN:et sett från servern

3.3.3 IP Forwarding

För att få trafik att vidarebefodras genom VPN-tunnel så krävs lite konfiguration. Först och främst måste vi göra så att vår Proxy agerar router. Det vill säga skickar vidare inkommande paket som skall till en annan adress än proxyns egna. Detta görs genom YaST och heter IP Forwarding.
 # yast

När YaST har startat gå in på Network Devices->Network Settings->Routing och klicka i ”Enable IP Forwarding”.

För att sedan tvinga all IP-trafik att gå igenom VPN:et så lägger vi till en rad i serverns OpenVPN-konfig som skickar med direktiv till klienterna att ändra default gateway. Detta tvingar klienten att ändra deras default gateway till VPN-servern vilket medför att all trafik från klienten skickas genom tunnel.
`push "redirect-gateway def1"`

Nu tvingas all trafik genom VPN:et, så även DNS-förfrågningar. Detta medför att vi även måste skicka med proxyns DNS-servers för att förfrågningarna skall skickas rätt. Även detta görs genom att en rad läggs till OpenVPN-serverns konfiguration.
push "dhcp-option DNS x.x.x.x"

Nu återstår ett sista steg för att allt ska fungera. Proxyn får nu in all trafik igenom tunneln, men vet ej hur den skall hanteras. Det gör vi genom att NAT:a trafiken som kommer från tunneln ut till internet. –o i våran konfiguration är eth2.
 # iptables -t nat -A POSTROUTING –s 10.0.8.0/24 -o <outgoing_interface> -j MASQUERADE

3.3.4 Multicast Forwarding

MC-forwarding visade sig vara den stora tidsboven i detta projekt. Det finns väldigt lite information, och tydligen inget vidare intresse av det om man tror på den begränsade informationen om ämnet, samt brist på ordentliga svar.

3.3.5 Iptables

Iptables kändes som det logiska valet vid IP Forwarding eftersom det ingår i de flesta linuxdistar, och informationen kring det borde vara riklig. Vi testade genomgående förgäves att försöka routa vidare MC-trafik, men insåg efter ett tag att det helt enkelt inte var möjligt med hjälp av iptables. Vi fick forwardering av vanlig trafik att fungera på ett flertal sätt, men dessvärre inte med MC-paket. Vi testade även att streama media till en unicast adress istället för multicast, vilket fungerade alldeles utmärkt. Detta var dock ingen acceptabel lösning och vi gick vidare.

Efter ett tag hittade vi följande, vilket visar att MC-forwarding är något helt separat från vanlig forwarding, och inte är påslaget som standard:
 # /sbin/sysctl –a| grep mc_forwarding
`net.ipv4.conf.all.mc_forwarding = 0`

När vi försökte ändra denna parameter så möttes vi av följande:
 # /sbin/sysctl –w net.ipv4.conf.all.mc_forwarding=1
Error: “Permission denied” setting key “net.ipv4.conf.all.mc_forwarding”

Detta tillsammans med den informationen vi lyckats hitta kring ämnet så drog vi slutsatsen att man helt enkelt inte kan ändra denna parameter i user mode, men ett program som körs i kernel mode kan ändra den åt oss.

3.3.6 IGMPproxy

Då vi inte fick iptables att fungera så började vi leta efter andra alternativ. Vi upptäckte då att det finns programvaror för att vidarebefordra MC-trafik. Program såsom pimd, mrouted och smcroute fungerade dock inte då programen tyvärr är föråldrade och inte var kompatibla eller inte gick att kompilera helt enkelt. Men dessa program ledde oss vidare, och till slut hittade vi igmpproxy som är en vidareutveckling av dessa program. Från OpenSuSes egna repository (software.opensuse.org) laddade vi ner och installerade rpm:en för senaste versionen.

 # wget
http://download.opensuse.org/repositories/home:/dmitry_serpokryl/openSUSE_11.1/i586/igmpproxy-beta2-7.1.i586.rpm
 # rpm –Uvh igmpproxy-beta2.7.1.i586.rpm

Detta installerar följande:
/etc/igmpproxy.conf
/usr/bin/igmpproxy
/usr/share/man/man5/igmpproxy.gz
/usr/share/man/man8/igmpproxy.conf.gz

Igmpproxy är ett relativt litet program, och fokuserar enbart på MC-forwarding. Detta innebär att lite konfiguration behövs och vi behövde bara specificera vilket interface igmp-requests skall skickas på, vilket interface den mottagna MC-trafiken skall skickas vidare på samt alternativa nät på interfacen.

Vi editerar exempelkonfigfilen som följer med igmpproxy och sätter tunneln som vårat downstream interface och specificerar vilket nät tunneln har. Vi säger alltså åt igmpproxy att den MC-trafik som kommer in skall skickas vidare till nätet 10.0.8.0/24 på tun0 .

phyint tun0 downstream ratelimit 0 threshold 1
altnet 10.0.8.0/24

Upstream Interfacet är det interface som är kopplat utåt, till internet (i vårat fall eth2), och är det interface som kommer prata med streaming servern. Vi specificerar från vilket nät vi accepterar MC-paket. Detta är adressen till din IPTV-leverantör.

phyint eth2 upstream ratelimit 0 threshold 1
altnet 192.168.0.0/24

Vi startar igmpproxy i debug mode med våran nyligen skapade konfigfil (**Bilaga A**):

 # /usr/bin/igmpproxy –d –c /etc/igmpproxy.conf

```
Current routing table (Insert Route):
---------------------------------------------------

Debu: #0: Src: 0.0.0.0, Dst: 239.255.255.250, Age:1, St: I, OutVifs: 0x00000005
Debu:
---------------------------------------------------
```

Figur 20: Visar i debugmode igmpproxys nuvarande routingtabell utan några forwardade adresser

Nedan syns en bild på IGMPProxy i debug mode när våran klient away vill starta en dataström från streamingservern som är igång till MC-gruppen 224.255.255.1.

1. Visar hur proxyn får in en förfrågan från 10.0.8.6 (aways IP i VPN:et) för att gå med i MC-gruppen 224.255.255.1. IGMPProxy tycker att gruppen borde läggas till routing tabellen och gör så
2. Visar hur IGMPProxy går med i MC-gruppen 224.255.255.1 med adressen 10.0.0.1 (proxyns ip ”mot internet”) på eth2
3. IGMPProxy sätter TTL på MC-paketen som kommer in till vårat virtuella interface till 1. Lägger till en Multicast Forwarding Cache (MFC) som säger att 192.168.0.1 (streamern) är den som skickar ut till MC-gruppen 224.255.255.1
4. Visar den nuvarande routing tabellen för IGMPProxy. Streamern har lagts till som källa för inkommande paket till 224.255.255.1.

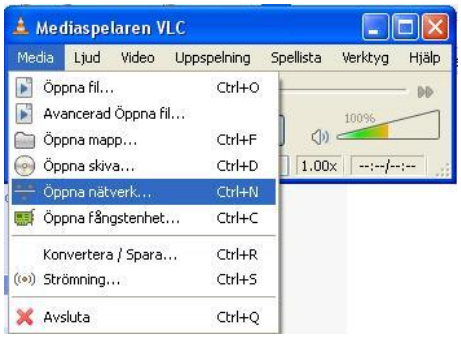
```
Current routing table (Insert Route);
---------------------------------------------------

Debu: #0: Src: 0.0.0.0, Dst: 239.255.255.250, Age:2, St: I, OutVifs: 0x00000005
Debu:
---------------------------------------------------

Note: RECV V2 member report   from 10.0.8.6        to 224.255.255.1
Debu: Should insert group 224.255.255.1 (from: 10.0.8.6) to route table. Vif Ix : 2   1
Debu: No existing route for 224.255.255.1. Create new.
Debu: Found existing routes. Find insert location.
Debu: Inserting at beginning, before route 239.255.255.250
Info: Inserted route table entry for 224.255.255.1 on VIF #2
Debu: Joining group 224.255.255.1 upstream on IF address 10.0.0.1
Note: joinMcGroup: 224.255.255.1 on eth2   2
Debu:
Current routing table (Insert Route);
---------------------------------------------------

Debu: #0: Src: 0.0.0.0, Dst: 224.255.255.1, Age:2, St: I, OutVifs: 0x00000004
Debu: #1: Src: 0.0.0.0, Dst: 239.255.255.250, Age:2, St: I, OutVifs: 0x00000005
Debu:
---------------------------------------------------

Note: RECV V2 member report   from 10.0.0.1        to 224.255.255.1
Note: The IGMP message was from myself. Ignoring.
Note: Route activation request from 10.0.0.1 for 224.255.255.1 is from myself. Ignoring.
Debu: Route activate request from 192.168.0.1 to 224.255.255.1
Debu: Vif bits : 0x00000004
Debu: Setting TTL for Vif 2 to 1
Note: Adding MFC: 192.168.0.1 -> 224.255.255.1, InpVIf: 1   3
Debu:
Current routing table (Activate Route);
---------------------------------------------------

Debu: #0: Src: 192.168.0.1, Dst: 224.255.255.1, Age:2, St: A, OutVifs: 0x00000004   4
Debu: #1: Src: 0.0.0.0, Dst: 239.255.255.250, Age:2, St: I, OutVifs: 0x00000005
Debu:
---------------------------------------------------
```

Figur 21: Visar IGMPproxy i debug mode

3.3.7 VLC – Streaming

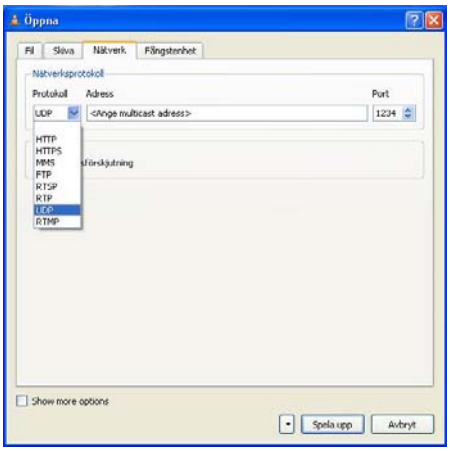
Vid val av mediaspelare valde vi att använda oss av VLC media player (www.videolan.org) för att spela upp våra dataströmmar. Anledningen till att vi valde detta program är att det finns tillgängligt för de flesta stora operativsystem, är gratis och kommer med de flesta populära videocodecs inbyggt, vilket medför mindre problem.

Nu är det dags att testa om vi kan ta emot en dataström genom tunneln. Öppna VLC och klicka på ”Media” menyn enligt bilden nedan.



Figur 22: Meny som skall öppnas för att spela upp en dataström

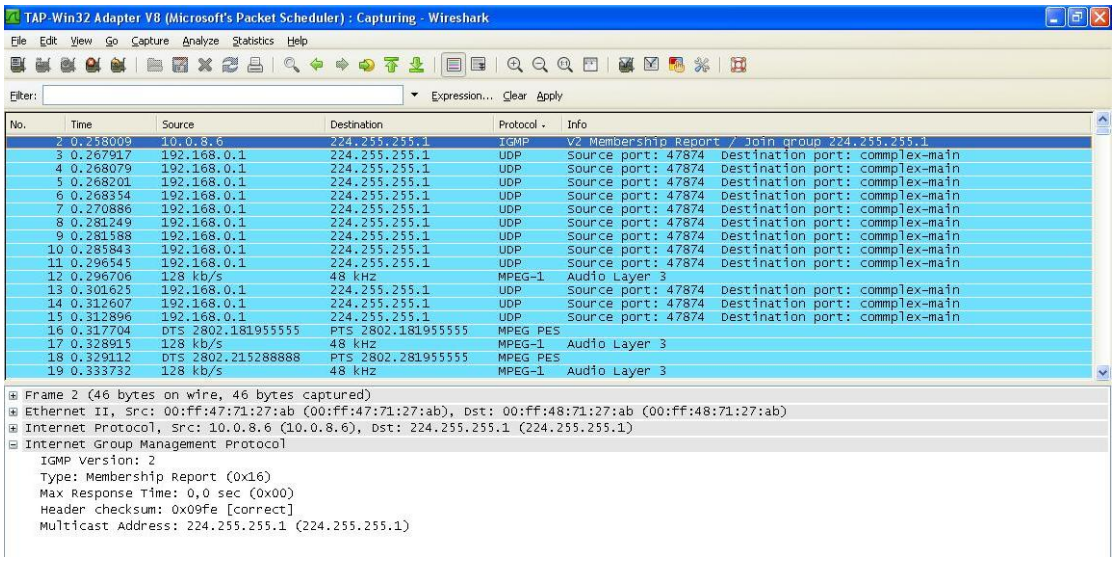
Figur 22 visar vilken meny som skall öppnas för att spela upp en dataström. Menyn ”Öppna nätverk” öppnar nedanstående bild.



Figur 23: Här anger man protokolltyp, multicastadress samt portnummer

Om allt funkar som det ska så borde dataströmmen öppnas och spelas upp inom några sekunder.

Figur 24: Visar uppspelandet av en ström i VLC



Figur 25: Visar i wireshark att klienten skickar iväg en IGMP join och hur videoströmmen sedan startar

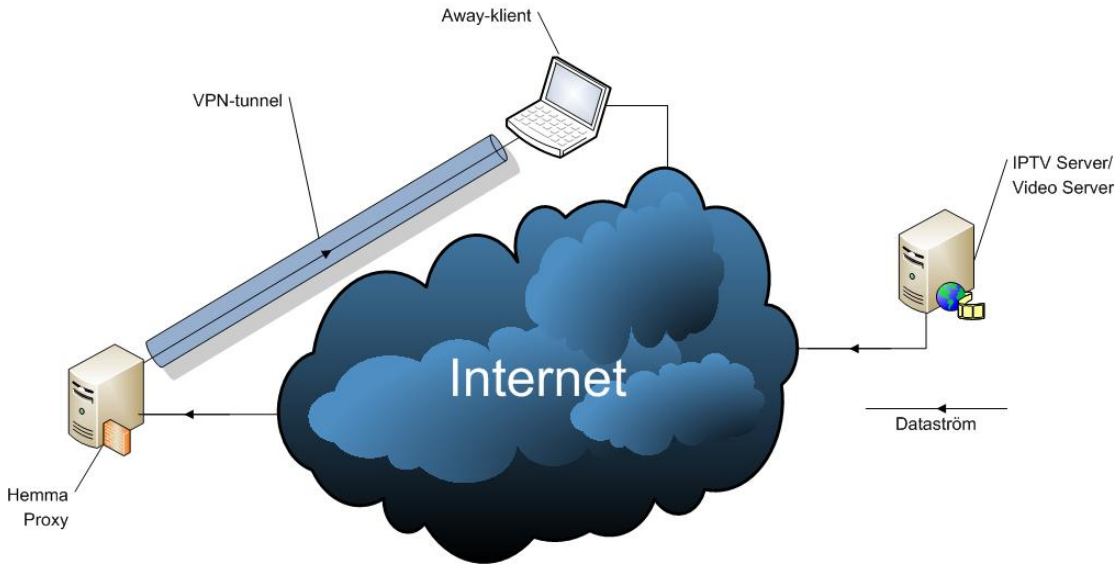
4 Analys

4.1 Lösning

Vi har genom hela examensarbetet haft en grundläggande filosofi om vad vår slutgiltiga lösning uppfylla. Under projektets gång har vi försökt uppnå dessa mål:

- Användarvänligt
- Kostnadseffektivt
- Säkert
- Okomplicerad

Vi anser att vi åstadkommit detta och skapat en stabil och billig produkt som inte kräver expertis inom området för att applicera för eget bruk. Lösningen är användarvänlig i den mån att applikationerna inte kräver speciellt underhåll efter det att produkten färdigställts. Efter det att lösningen är i bruk räcker det med en enkel knapptryckning för att komma igång.



Figur 26: Visar hur en dataström tar sig genom vår lösning

Kostnaden för lösningen är minimal, då den enda faktiska utgiften är hårdvaran för proxy och klienten. För vardagligt bruk av IPTV och andra tjänster behövs inte någon högre kapacitet på serverhårdvaran. Det är först vid modifikation av data som krav ställs på prestandan, som t.ex. omkodning av en mediaström. Den mjukvara som används är opensource vilket betyder att de kan laddas ner och användas gratis.

Vår lösning bygger på ett virtuellt privat nätverk som krypteras med SSL/TLS vilket är en standard inom internetkommunikation. Detta gör att lösningen är säker att använda, då trafiken i den virtuella tunneln är krypterad. Så länge nycklarna för

VPN:et behandlas på ett smart sätt så att utomstående inte har en möjlighet att förvärva dem kommer trafik i tunneln vara skyddad.

Lösningen bygger på att tre grundläggande programvaror:

- openSUSE
- OpenVPN
- IGMPProxy

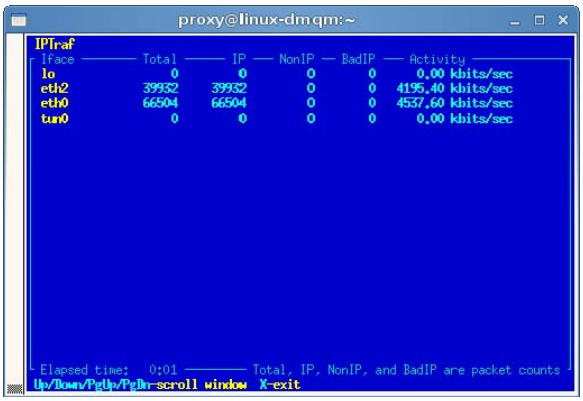
Genom att använda oss av så lite programvara eller konfiguration som var möjligt skapade vi en okomplicerad produkt. Detta gjorde att vi kunde eliminera onödiga problem med applikationer som inte var vitala för lösning, och på så sätt undvika en mer komplicerad konfiguration som kunde förvirra användaren.

4.1.1 Mätningar

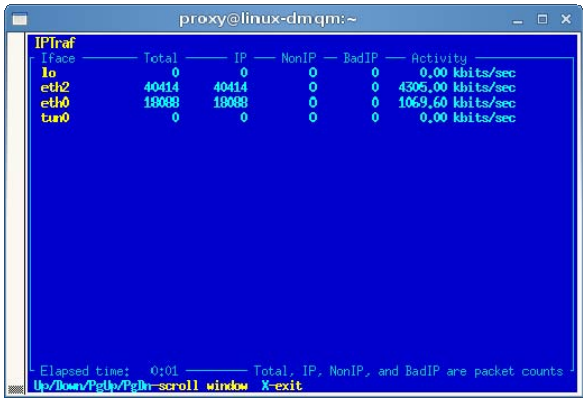
Vi har som tidigare nämnt begränsat våra mätningar till att enbart testa lösningen funktionalitet i form av mindre modifieringar av dataströmmen. Nedan visar bilder från iptraf, ett statistik program som tillåter dig att analysera nätverkstrafik. Bilderna visar vår proxy och dess två nätverksinterfaces, eth0 är interfacet där VPN-tunneln är uppkopplad och eth2 är interfacet där mediaströmmen kommer in.

Som det visas i figur 29 nedan så skickas det fler paket på eth0 än på eth2. Detta beror på att eth0 (tunneln) kapslar in datan i krypterade paket vilket gör att mindre payloaddata får plats i paketen, och detta medför att mer trafik skapas.

I figur 30 ser vi hur bredbandsanvändningen på tunnelinterfacet har minskat drastiskt. Detta är exakt samma dataström som tas emot av proxyn, men innan trafiken skickas vidare i tunneln så kodas dataströmmen om med 50% reducerad bildstorlek. Till synes har bredbandsanvändningen på tunneln minskat med ~75%, vilket kan vara en effektiv lösning om man är begränsad till en mindre bandbredd.



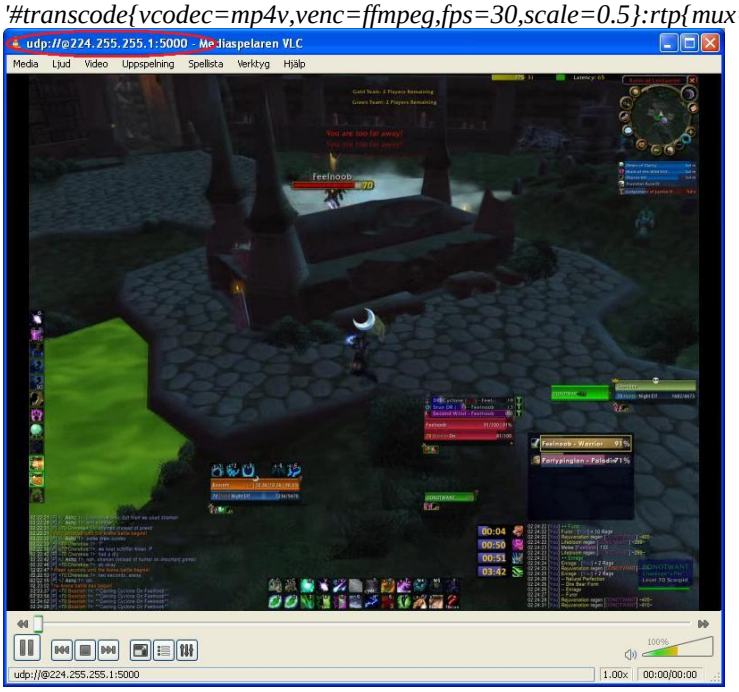
Figur 27: Visar bandbreddsanvändning av dataström utan omkodning



Figur 28: Visar bandbreddsanvändning av dataström med omkodning. Skala 50

Med hjälp av VLC kodar vi alltså om dataströmmen som kommer in från 224.255.255.1:5000 (se figur 30), skalar ner den med 50% och skickar vidare på 224.255.254.1:50004 (se figur 29**)**. Resultatet visas nedan. Notera att den omkodade dataströmmen ligger ~1-2s efter originalet på grund av realtidskonvertering till det nya formatet.

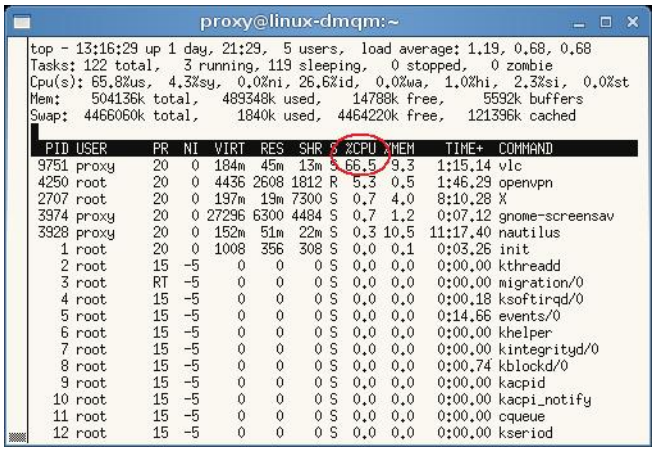
 # vlc udp://@224.255.255.1:5000 --sout '#transcode{vcodec=mp4v,venc=ffmpeg,fps=30,scale=0.5}:rtp{mux=ts,dst=224.255.254.1}'



Figur 30: Visar original av dataström

Figur 29: Visar ström skalad 50%

Vad som kanske inte syns i bilderna ovan är att bildkvalitén givetvis försämras. Vad vi vinner i prestanda förlorar vi i kvalité. Ett annat problem som vi tidigare nämnt är att omkodningsprocessen är krävande på hårdvaran. Vid vår enkla omkodning ovan så användes en större del av processorkapaciteten till just denna uppgift. Detta gav oss insikten att för tyngre omkodningar krävs kraftfullare hårdvara.



Figur 31: Visar CPU-användning av VLC vid omkodning av dataströmmen

5 Slutsats

Vi ville skapa en produkt som gjorde det möjligt använda sig av allt man kan när man är hemma, även om man inte är det. Detta har vi uppnått och vi är nöjda med resultatet.

I slutändan när vi ser tillbaka på den tid som spenderats på projektet kan vi konstatera att vårt första intryck av MC-forwarding skiljde sig ganska mycket från hur det ser ut i verkligheten. När vi startade projektet hade vi uppfattning att forwardering av trafik, oavsett av vilken typ, var den samma. Tyvärr förlitade vi oss på våra tidigare erfarenheter angående forwarding, vilket resulterade i att tid spenderades onödigt på vad som i slutändan ledde oss in i en återvändsgränd. Sen att ämnet inte verkar vara speciellt populärt gjorde vårt jobb ännu svårare, och den information vi kunde hitta vara oftast väldigt gammal.

Den första tiden spenderades på att försöka få igång MC-forwarding genom iptables, manipulation av routingtabeller och aktivering av MC-forwarding. Efter många försök utan större framgångar insåg vi att det inte var möjligt att MC-trafik skickas vidare genom iptables, och är avslaget som standard. Denna parameter kan enbart ändras av kerneln själv. Tyvärr så finns det inte möjlighet för en användare att aktivera denna funktion, utan detta måste göras av en programvara som körs i kernel mode.

IGMPProxy blev vår räddning, detta program gjorde precis det vi ville. Programmet krävde lite konfiguration, om än smått förvirrande information tillhandahölls hur programmet fungerade. Detta gjorde till en början att vi övergav programmet och letade nya alternativ och även fortsatte förgäves med att få IPtables fungerande. Efter vidare efterforskning insåg vi att IGMPProxy var det rätta programmet för vårt ändamål. När vi förstått hur konfigurationen faktiskt fungerade var det enkelt att få igång MC-forwarding och få vår dataström till klienten.

Så här i efterhand när man ser tillbaka på projektet så inser man att om vi skulle återskapa miljön skulle det gå relativt fort. Men tiden det tog att forska och ta fram information om området var betydligt underskattat av oss, och tog mycket längre tid än vad vi beräknat till en början. Vi känner att detta projekt har gett oss en inblick i hur routing och mediatekniker på internet fungerar och hur mycket jobb det kan vara att få även de minsta detaljerna att fungera.

5.1 Framtida möjligheter

Det finns utrymme för utveckling av produkten, men mycket av detta har varit utanför ramen av vad vi ville uppnå med projektet. Som vi nämnde tidigare så testade vi enbart att omkoda en dataström som kom in till proxyn. Detta skulle kunna automatiseras och eventuellt kunna styras genom ett grafiskt gränssnitt från klienten,

istället för att manuellt behöva skriva in koden på proxyn. Det finns även mycket småscript som man skulle kunna implementera på serversidan för att förenkla processer som skapande av certifikat/nycklar, användare samt konfigurationer.

För IPTV är vår lösning väldigt skalbar, men om vi börjar prata video-on-demand så är den begränsad p.g.a. att multicast används. Vår lösning inriktade sig dock på privat bruk, och har det som mål. I större nätverk skulle däremot en peer-to-peer lösning kunna utvecklas, och istället för att en server belastas med all trafik så delar användarna på jobbet och fungerar ungefär som bittorrent. Vi ser fler och fler företag som håller på att utveckla just denna funktionalitet.

Källförteckning

[1] Internet Protocol Suite (TCP/IP), http://en.wikipedia.org/wiki/Internet_Protocol_Suite Datum: 2009-09-11

[2] J. Postel, User Datagram Protocol (UDP), IETF, 28 Aug. 1980 http://tools.ietf.org/rfc/rfc768.txt

[3] Streaming http://en.wikipedia.org/wiki/Streaming_media Datum: 2009-09-23

[4] H. Schulzrinne, S. Casner, R. Frederick, och V. Jacobson, Real Time Transport Protocol (RTP), IETF, Network Working Group, Request for Comments:3550, July 2003 http://www.ietf.org/rfc/rfc3550.txt

[5] Real Time Control Protocol (RTCP) http://en.wikipedia.org/wiki/RTP_Control_Protocol Datum: 2009-09-23

[6] H. Schulzrinne, A. Rao, och R. Lanphier, Real Time Streaming Protocol (RTSP), IETF, Network Working Group, Request for Comments: 2326, April 1998. http://tools.ietf.org/rfc/rfc2326.txt

[7] Multicast http://en.wikipedia.org/wiki/Multicast Datum: 2009-10-17

[8] A. Adams, J. Nicholas, och W. Siadak, Protocol Independent Multicast (PIM), IETF, Network Working Group, Request for Comments: 3973, Jan. 2005. http://tools.ietf.org/rfc/rfc3973.txt

[9] Unicast http://en.wikipedia.org/wiki/Unicast Datum: 2009-10-17

[10] Internet Group Management Protocol (IGMP) http://en.wikipedia.org/wiki/Internet_Group_Management_Protocol Datum: 2009-10-17

[11] IGMPproxy http://software.opensuse.org/search?baseproject=openSUSE%3A11.2&p=1&q=igmpproxy Datum: 2009-10-20

[12] IPTV, IPTV Information, 2008. http://www.iptvinformation.net/IPTV+FAQ.aspx Datum: 2009-10-29

[13] VLC Media Player http://en.wikipedia.org/wiki/VLC_media_player Datum: 2009-10-18

[14] Bradley Mitchell, VPN Solutions and Key Features, About.com Guide
http://compnetworking.about.com/od/vpn/a/what_is_a_vpn.htm
Datum: 2009-09-11

[15] OpenVPN, OpenVPN Technologies Inc.
http://openvpn.net/index.php/open-source.html
Datum: 2009-09-11

[16] OpenVPN HowTo, OpenVPN Technologies Inc.
http://openvpn.net/index.php/open-source/documentation/howto.html
Datum: 2009-09-22

[17] Secure Sockets Layer (SSL)
http://sv.wikipedia.org/wiki/SSL
Datum: 2009-09-23

[18] Transport Layer Security (TLS)
http://en.wikipedia.org/wiki/Transport_Layer_Security
Datum: 2009-09-23

[19] Angelos D. Keromytis, John Ioannidis, Jonathan M. Smith, Implementing IPsec, In proceedings of GlobeCom Conference, IEEE, Nov. 1997, pp. 1948-1952.
http://www1.cs.columbia.edu/~angelos/Papers/ipsec.pdf
Datum: 2009-12-02

[20] IP Security (IPsec)
http://en.wikipedia.org/wiki/Ipsec
Datum: 2009-10-17

[21] S. Kent och K. Seo, Security Architecture for the Internet Protocol, IETF, Network Working Group, Request for Comments: 4301, Dec. 2005. http://tools.ietf.org/rfc/rfc4301.txt

[22] S. Kent och R. Atkinson, IP Authentication Header, IETF, Network Working Group, Request for Comments: 2402, Nov.1998.
http://www.ietf.org/rfc/rfc2402.txt

[23] Encapsulating Security Payload (ESP), Microsoft Corporation, 2009.
http://technet.microsoft.com/en-us/library/cc959510.aspx

[24] Internet Key Exchange (IKE)
http://en.wikipedia.org/wiki/Internet_key_exchange
Datum: 2009-10-17

[25] D. Harkins och D. Carrel , The Internet Key Exchange (IKE), IETF, Network Working Group, Request for Comments: 2409, Nov. 1998l.
http://tools.ietf.org/html/rfc2409

[26] T. Dierks and C. Allen, The TLS Protocol: Version 1.0, IETF, Network Working Group, Request for Comments: 2246, January 1999.
http://www.ietf.org/rfc/rfc2246.txt

Bilagor

Bilaga A – igmpproxy konfig

```
##------------------------------------------------------
## Enable Quickleave mode (Sends Leave instantly)
##------------------------------------------------------
quickleave

##------------------------------------------------------
## Configuration for tun0 (Downstream Interface)
##------------------------------------------------------
phyint tun0 downstream  ratelimit 0  threshold 1
      altnet 10.0.8.0/24
##------------------------------------------------------
## Configuration for eth2 (Upstream Interface)
##------------------------------------------------------
phyint eth2 upstream  ratelimit 0  threshold 1
      altnet 192.168.0.0/24
```

Bilaga B – OpenVPN server konfig

```
# Sätter IP-adressen som server lyssnar på efter inkommande
# anslutningar. Detta borde vara adressen du får av din ISP
local 10.0.10.1

# Anger port som servern lyssnar på och vilket protokoll som används
port 1194
proto tcp

# Typ av virtuellt interface. Tun används för routing
dev tun

# Sökväg till nycklar, certifikat och DH
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/server.crt
key /etc/openvpn/keys/server.key
dh /etc/openvpn/keys/dh1024.pem

# Det virtuella nätet som OpenVPN kommer använda sig av. Servern
# kommer att få IP-adressen 10.0.8.1
server 10.0.8.0 255.255.255.0

# Säger åt klienterna att använda VPN-servern som default gateway.
# Detta medför att all trafik tvingas att gå genom tunneln
push "redirect-gateway def1"

# Skickar med multicast routes till klienterna
push "route 224.0.0.0 240.0.0.0"

# Sparar en lista på namn och virtuellt IP till klienterna
ifconfig-pool-persist ipp.txt

# Pingar server var 10:e sekund för att kolla om den är uppe. Om
# inget svar fås efter 120 sekunder tolkas det som att servern är
# nere
keepalive 10 120

# Använder lzo-komprimering
comp-lzo

# Försöker bibehålla sessionen mellan omstarter
persist-key
persist-tun

# En kort logfil om nuvarande anslutningar
status openvpn-status.log

# Maximalt antal klienter som är tillåtna samtidigt
max-clients 100

# Beskriver hur detaljerad logfilen skall vara
verb 3
```

Bilaga C – OpenVPN klient konfig

```
# Anger att konfigurationsfilen tillhör en klient
client

# Typ av virtuellt interface. Tun används för routing
dev tun

# Anger vilket protokoll som används
proto tcp

# IP-nummer och port till VPN-server
remote 10.0.10.1 1194

# Binder inte klient till en speciell port
nobind

# Försöker bibehålla sessionen mellan omstarter
persist-key
persist-tun

# Sökväg till nycklar och certifikat, om klienten använder windows OS
# behövs \\ i sökvägen
ca "C:\\Program Files\\OpenVPN\\keys\\ca.crt"
cert "C:\\Program Files\\OpenVPN\\keys\\away.crt"
key "C:\\Program Files\\OpenVPN\\keys\\away.key"

# Använder lzo-komprimering
comp-lzo

# Beskriver hur detaljerad logfilen skall vara
verb 3
```

TRITA-ICT-EX-2009:219

www.kth.se