

**Central styring
af
spredte autonome netværk**

Erik Bidstrup
bidstrup@diku.dk

Jesper Dangaard Brouer
hawk@diku.dk

Per Marker Mortensen
permm@diku.dk

2. Juli 2001

Resumé

Dette projekt omhandler etablering og drift af et regionalt kollegienetværk i København. Projektet identificerer forskellige problemstillinger ved etablering og drift af det regionale kollegienetværk. Udvalgte problemstillinger er analyseret og der er lavet designbeskrivelser af løsningsforslag. Løsningsforslagene inkluderer system til kildeidentifikation af trafik, samt system til valg blandt multiple internetgateways. Projektet afsluttes med en handlingsplan henvendt til Foreningen Kollegienet København, der forventes at gå videre med projektet.

Indhold

1	Indledning	1
2	Identifikation af problemstillinger	4
2.1	Parternes ønsker og behov	4
2.2	Vægtning af parternes ønsker og behov	6
2.3	Problemstillinger	6
2.4	Juridiske spørgsmål	7
2.4.1	Kildeidentifikation	7
2.5	Administration	7
2.5.1	Administrationens opgaver	7
2.5.2	Database	7
2.5.3	Distribution	7
2.6	Stabilitet	8
2.6.1	Ensartet hardware	8
2.6.2	Netværksovervågning	8
2.6.3	Server sikkerhed	8
2.7	Billigt Internet	8
2.7.1	Deling af internetforbindelse	9
2.7.2	Multiple internetgateways	9
2.8	Flexibilitet	9
2.9	Fairness	9
2.9.1	Trafikaccounting	9
2.9.2	Fair queing	10
2.10	Sikkerhed	10
2.10.1	Firewall	10
2.10.2	Intrusion Detection	10
2.10.3	Virusscanning	10
2.11	Nemt	10
2.11.1	Autokonfiguration	10
2.11.2	Oplysningssystem	11
2.11.3	Brugerstatus/konfigurations system	11
2.12	Hjemmeside, e-mail o.s.v.	11
3	Analyse og design	12
3.1	Udvælgelse af problemstillinger	12
3.2	Overordnede krav og antagelser	13
3.2.1	Autonomt-netværk	13

3.2.2	IPv4 vs. IPv6	13
3.3	Generelt om fjerndrift	14
3.3.1	Netværks sikkerhed	14
3.3.2	Eksisterende protokoller og værktøjer	16
3.3.3	Push vs. Pull	17
3.3.4	SSH - secure shell	17
3.3.5	Cfd - Cfengine dæmonen	18
3.3.6	CVS - Concurrent Versions System	18
3.3.7	Rsync	19
3.3.8	IPsec - IP security protocol	19
3.3.9	FTP - File Transfer Protokol	20
3.3.10	SNMP - Simple Network Management Protokol	20
3.4	Kildeidentifikation	21
3.4.1	Placering af kontrolpunkt	21
3.4.2	Identifikation af bruger	21
3.4.3	IPsec	21
3.4.4	Proxy med authentication	22
3.4.5	Identifikation baserede på IP/MAC-adresse	22
3.4.6	Pakkeidentifikation	22
3.4.7	Login	23
3.4.8	Udlogging	23
3.4.9	Automatisk udlogging	23
3.4.10	Sikkerhed	24
3.4.11	Design	25
3.5	Multiple internetgateways	28
3.5.1	Hvorfor benytte multiple internetgateways	28
3.5.2	Valg af gateway per kilde, trafikkarakteristika mm.	28
3.5.3	Hvor ligger de tekniske problemer	29
3.5.4	Krav til funktionalitet	30
3.5.5	Border Gateway Protocol, BGP4	31
3.5.6	Skift af IP-adresse på klient	32
3.5.7	Network Address Translation, NAT og NAT	33
3.5.8	Design af mulig løsningsmodel	35
3.6	Bifrostmaskinen	38
3.6.1	Operativsystem	38
3.6.2	Softwarevedligeholdelse	39
3.6.3	Konfigurationsfiler	39
3.7	Åbne/lukke system	41
3.7.1	Pakkefiltrering	41
3.7.2	Grund elementer	44
3.7.3	Tilstands-opdateringer	44
3.7.4	Grundreglerne	44
3.7.5	Design af regelkæderne	44
3.7.6	Oplysningssystem	45
3.7.7	Åbne/lukke API	46

4.1	Kildeidentifikation	48
4.1.1	Implementationen i forhold til design	48
4.1.2	Programbeskrivelse	49
4.1.3	Afprøvning	50
5	Handlingsplan	51
6	Konklusion	53
	Litteratur	54
A	Kode: portlukker2k	56
B	Kode: lukmigud	59
C	Kode: lukmig	66
D	Arbejdsbeskrivelse	69
D.1	Formål	69
D.2	Baggrund	69
D.3	Motivation og målsætning	70
D.4	Aflevering	70

Figurer

1.1	Skitse af det regionale kollegienetværks vigtigste komponenter. De centralt styrede firewalls/routere er symboliserede som en kasse med et ‘B’, internetgateways er symboliseret med et ‘I’.	2
3.1	Generelle netværks sikkerheds trusler. (Kilde: Klaus Hansens noter, fra sikkerhedskursus)	15
3.2	De overordede moduler i implementeringen af Bifrost. Stiplede pile angiver datastrøm. Fuldt optrukne pile angiver retningen af ‘kald’ — det kan være funktionskald i en API eller kald over netværk. På figuren er vist et automatisk udlogningsmodul, der anvender ICMP ping som bekræftet i afsnit 3.4.9. Et automatisk lukningsmodul, der anvender ‘aktiv trafik’ vil kommunikere med Åbne/lukke modulet frem for klienten.	26
3.3	IPv4 pakkeheader, fra [23]	29
3.4	Simpelt senarie med multiple internetgateways.	30
3.5	Simpelt senarie med syv BGP4 routere.	31
3.6	Skematisk afbildning af NAT’s virkemåde. Tiden er afbildet langs den vertikale akse. De tre involverede maskiner X, Y og Z er symboliseret med fede vertikale streger. Datagrammerne er symboliseret med kasser, hvor afsenderadresser er øverst og modtageradressen er nederst.	33
3.7	Indbygget regelkæder. Kilde [17]	42
3.8	Bruger defineret regelkæder. Kilde [17]	43
4.1	Principskitse af vores implementation. Sammenhold eventuelt med vores design i figur 3.2.	49

Kapitel 1

Indledning

Denne rapport er resultatet af et projektarbejde i faget “Projektet i datanet” foråret 2001. Projektdeltagerne er: Erik Bidstrup, Jesper Dangaard Brouer samt Per Marker Mortensen. Vejleder på projektet har været Per Høgh.

Projektet handler om forskellige tekniske problemstillinger ved etablering og drift af et regionalt kollegienetværk i København.

Projektets baggrund er tanken om et regionalt kollegienet i Københavnsområdet. Vi, projektgruppens medlemmer, har i nogle år være tilknyttet Foreningen Kollegienet København, Kbhkol¹. Kbhkol arbejder for at oprette et regionalt kollegienetværk mellem de enkelte kollegier i København.

Dette netværk skal blandt andet fungere som internetforbindelse for de enkelte kollegier. Formålet med netværket er at give billigere, bedre, hurtigere og nemmere adgang til Internet for kollegianerne.

Det regionale kollegienetværk tænkes administreret fra centralt hold. Derved fjernes behovet for lokal ekspertise og arbejdskraft på de enkelte kollegier. Dette, håber foreningen, vil give flere kollegier mulighed for at tilbyde Internet til sine kollegianere.

Det regionale kollegienetværk tænkes realiseret ved hjælp af tre vigtige komponenter.

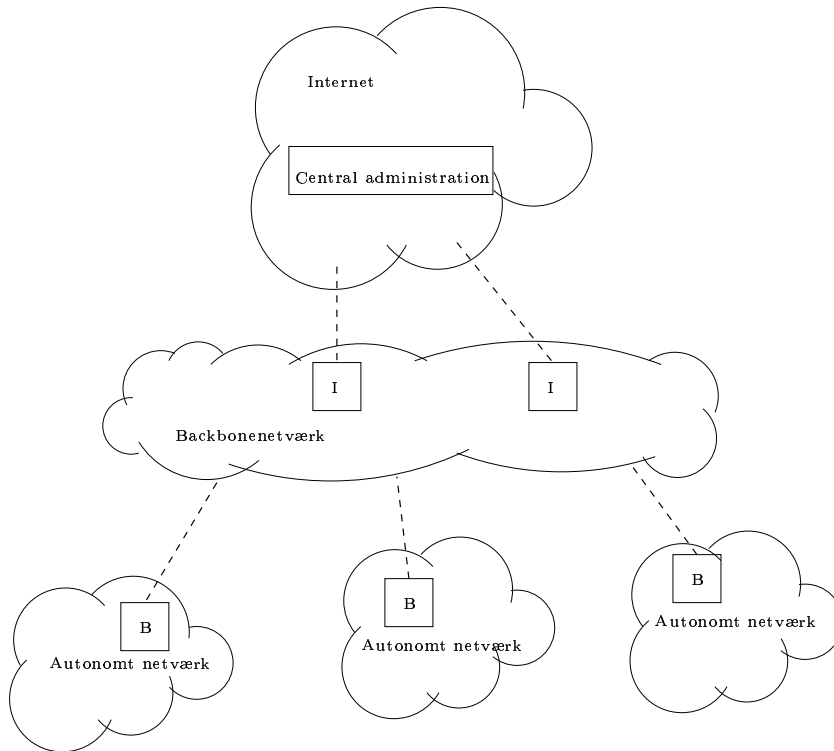
Lån af backbonenetværk Københavns Universitet (KU) har tilbudt de kollegier, som kan tilsluttes eller er tilsluttet KU's netværk, mulighed for at route trafik mellem hinanden gennem KUs netværk. Der er med andre ord mulighed for at benytte KU's netværk som backbone netværk mellem kollegierne.

Billig Internetforbindelse Studerende ved institutioner tilknyttet Forskningsnettet² har for tiden lov til at benytte forskningsnettets internetforbindelse gratis via institutionernes netværk. Der vil måske i fremtiden blive afkrævet betaling, men under alle omstændigheder forventes denne forbindelse at være billig og meget hurtig.

Centralt styrede firewalls/routere Der tænkes opsat en centralt styret firewall/router

¹<http://www.kbhkol.dk/>

²<http://www.forskningsnettet.dk/>



Figur 1.1: Skitse af det regionale kollegienetværks vigtigste komponenter. De centralt styrede firewalls/routere er symboliserede som en kasse med et 'B', internetgateways er symboliseret med et 'I'.

på hver enkelt kollegie. Denne enhed skal klare alt fra styring af de enkelte kollegianeres adgang til Internet til autokonfiguration af kollegianernes computere.

Figur 1.1 viser en skitse af de vigtigste komponenter.

*Det primære mål for vores projekt er at arbejde med den centralt styrede firewall/router. Denne er kernen i hele det regionale netværk og den forventes at indeholde mange avancerede tekniske problemstillinger. Vi kalder dette stykke udstyr for **Bifrost**³.*

Projektets mål kan ridses op således:

- Klarlægge alle involverede parters ønsker, behov, mål samt krav vedrørende de tekniske og administrative aspekter af det regionale kollegienetværk.
- Identificere problemstillinger ved det regionale kollegienetværk med baggrund i de klarlagte ønsker, behov, mål samt krav.
- Analysere udvalgte problemstillinger og give løsningsforslag til samme.
- Give designbeskrivelse af udvalgte løsningsforslag.

³Navnet er opstået samtidig på Egmont kollegiet i København og ved kollegierne omkring Lund i midten af 1990'erne. Ligeledes eksisterer der en minimal Linux distribution ved navn Bifrost (<http://bifrost.slu.se/>).

- Implementere enkelte af løsningsforslagene.
- Udarbejde en handlingsplan som kbhkol kan benytte som en teknisk ledesnor for hvad der er af tekniske problemstillinger, som skal løses før det regionale kollegienetværk kan etableres.

Vi ønsker at generalisere mest mulig i vores analyse af problemstillingerne, da vores løsning skal kunne anvendes af såvel eksisterende som nyetablerede netværk. De fleste kollegienetværk er opbygget uafhængigt af hinanden, de er derfor meget forskellige i opbygning. Ud over dette ved man aldrig, hvad fremtiden byder. Mere generelt kan man **opfatte kollegienetværk som autonome netværk**, hvis opbygning kan variere. Vi vælger derfor at betegne et kollegienetværk som et autonomt netværk, og vi mener herved, at vi stiller færrest mulige krav til opbygningen af det netværk som ligger bag den centralt styrede Bifrost maskine. **Heraf titlen: “Central styring af spredte autonome netværk”.**

Vores mål er at løse problemstillinger fra den virkelige verden. Vi vil derfor ligge vægt på, at vore løsningsforslag kan benyttes i de konkrete omgivelser. Vi ønsker, at vores løsning til problemstillingerne resulterer i et funktionelt, hurtigt, billigt og vedligeholdelsesfrit netværk med nem mulighed for tilkobling. Vi vil derfor i stor udstrækning benytte standardkomponenter til løsning af problemstillingerne.

Fagligt forventer vi, at vores arbejde med vægtning af løsningsmuligheder vil give os et godt indblik i, hvorledes den datalogiske disciplin datanet viser sig i den virkelige verden. Vi håber dette vil ruste os godt til videre arbejde med datanet i andre sammenhænge.

Rapporten er delt op i følgende kapitler:

Indledning Dette kapitel.

Identifikation af problemstillinger Her identificeres problemstillinger ved det regionale kollegienetværk med baggrund i ens klarlæggelse af ønsker, behov, mål samt krav.

Analyse af problemstillinger og design af løsninger Her analyseres udvalgte problemstillinger og der gives løsningsforslag til disse. Der gives designbeskrivelse af et udvalgt løsningsforslag per problemstilling.

Implementation og afprøvning Her beskrives implementationen og afprøvning af enkelte løsningsforslag.

Handlingsplan Her gives en teknisk handlingsplan for etablering af det regionale kollegienetværk. Handlingsplanen er beregnet for kbhkol til brug for foreningens videre arbejde.

Konklusion Her konkluderes på projektarbejdet.

Kapitel 2

Identifikation af problemstillinger

I dette kapitel vil vi forsøge at identificere problemstillinger ved etablering og drift af et regionalt kollegienetværk i København, herunder den centralt styrede firewall/router. Vi vil gøre dette ved først at beskrive ønsker, behov og krav for de forskellige parter i et regionalt kollegienetværk. Vi vil også komme ind på, hvorledes de forskellige parter ønsker og behov bliver vægtet i forhold til hinanden.

2.1 Parternes ønsker og behov

Der er mange forskellige parter i et regionalt kollegienetværk i København:

Kollegianerne En enkelt beboer på et kollegiet, der i denne sammenhæng forventes at have egoistiske interesser.

Kollegierne Et kollegie i netværket. Et kollegies interesser er summen af kollegiets administrations interesser og kollegianernes interesser som helhed. Kollegiets interesse vil oftest blive varetaget af kollegiets administration, beboerråd og bestyrelse.

Kbhkol Foreningen Kollegienet København som er ophavsmænd til ideen om et regionalt kollegienetværk. Kbhkol varetager medlemmernes, altså kollegiernes, samlede interesser.

Backbonenetværk De regionale netværk, som tænkes benyttet som backbone i det regionale kollegienetværk, f.eks. Københavns Universitet netværk.

Internetudbydere Internet trafik der ikke tillades på forskningsnettet eller backbonenetværket, skal via en kommerciel internetudbyder.

Driftorganisationen Den organisation, der skal varetage den centrale drift af det regionale kollegienetværk. Da organisationen er finansieret af kollegierne skal driftorganisationen som udgangspunkt følge disse ønsker. Driftorganisationens ønsker er dog typisk af teknisk karakter og derfor uafhængig af kollegiernes ønsker.

Vi vil i det følgende beskrive de forskellige ønsker, behov, mål, interesser samt krav (i det følgende samlet benævnt som behov) som de forskellige parter kunne tænkes at have. Løsningen skal, hvis den skal blive en succes, kunne accepteres af alle parter.

Kollegianerne :

- Nemt og hurtigt at blive tilsluttet netværket.
- Sikkerhed/trykthed: Man kan forvente, at dette vil blive efterspurgt som en firewall.
- Generelt fleksibelt, løsningen skal fungere sammen med min computer og mine applikationer (læs: alle computere og alle applikationer).
- Billigt og hurtigt, disse to ting vægtes individuelt, men de fleste kan blive enige om at det er gode egenskaber.
- Fair fordeling af båndbredde, den enkelte ønsker ikke at betale for noget andre bruger. De kollegianere, der bruger meget båndbredde er dog ikke interessede i fair fordeling. Herunder mulighed for betaling efter forbrug.
- Stabilitet.
- Mulighed for at hjemmeside, e-mail o.s.v.

Kollegierne :

- Nem drift, da kollegiernes administration kan være konservativt indstillet er det vigtigt, at deres arbejdsdag påvirkes mindst muligt af, at løsningen installeres.
 - Ingen behov for at ansætte teknisk kompetence
 - Ingen behov for at ansætte sekretær til papirarbejde
 - Eventueller procedurer er simple og regelmæssige.
 - Drift af fysisk forbindelse og andre forpligtigelser varetages eksternt.
 - Fast pris, da kollegiet er afhængig af, at kunne lægge et budget er de afhængige af, at prisen kan berammes forud for et regnskabsår.
- Lav pris
 - Løsningen skal kunne anvendes uden at et eventuelt etableret lokalnet skal udvides væsentligt.
- Fair fordeling af båndbredde.
- Kontrol over juridiske problemstillinger, kollegiet ønsker ikke at blive stillet til ansvar for en kollegianers handlinger.
 - Mulighed for at kunne placere ansvaret for misbrug på misbrugeren.
 - Opsporing af ureglementeret brug af lokal og internet.

Kbhkol :

- Foreningens formål er at sikre og fremme unge uddannelsessøgendes adgang til Internet.

- Mulighed for statistik over forbrugsmønstre til brug i overvejelser om eventuel betaling.
 - Kollegieanerens internetforbrug sammenholdt med studieretning, de forskellige institutioners trafikmængde.
 - Trafik til og fra uddannelsesinstitutionerne versus resten af Internet.

Backbonenetværk :

- Sikkerhed mod at kollegierne overbelaster backbonenetværket.
- Ønsker at kollegierne, der tilsluttes, bruger samme løsning når de tilslutter sig.
- Ønsker en organisation, der formidler kontakten til kollegierne.
- Kontrol med hvem som har adgang til internetgateway.

Internetudbyder :

- Mulighed for at kunne spore trafik til afsender.
- Kontrol med adgang til internetgateway.

Driftsorganisation :

- Driftsorganisationen vil som overordnet mål have et netværk, der er let og overskueligt at administrere, det regionale kollegienetværk.
- Mindst mulig support.
- Mindst mulig driftsarbejde.

2.2 Vægtning af parternes ønsker og behov

De forskellige parterers behov er i nogle tilfælde modstridende og vi bliver derfor nød til at lave en vægtning af dem.

Nogle af parterne bør have deres behov opfyldt. F.eks. vil det være u hensigtsmæssigt, hvis driftsorganisationens behov, der sikrer stabil drift, ikke blev opfyldt. Et andet eksempel er backbonenetværket som for at give os adgangen til deres netværk nødvendigvis skal have alle krav opfyldt.

Vores opgave er at skabe overblik over de samlede behov og vælge løsninger som maksimerer summen af parternes tilfredshed. Vi vil, hvor det er relevant diskutere vægtningen mere eksplicit.

2.3 Problemstillinger

Vi vil her beskrive forskellige tekniske problemstillinger, der er afledt af parternes behov. For hver af problemstillingerne vil vi komme med en beskrivelse samt en forklaring på, hvilke parterers behov, som ligger til grund for problemstillingen.

2.4 Juridiske spørgsmål

Kollegierne kræver, at ansvaret for eventuel kriminel aktivitet kan placeres på en beboer. Det giver ligeledes internetudbydere og backboneleverandører tryghed, at kollegierne er afklarede omkring og kan håndtere placering af juridisk ansvar.

2.4.1 Kildeidentifikation

Ved kildeidentifikation forstår vi det at kunne spore enhver datastrøm på nettet tilbage til en kollegianer. Kan kildeidentifikation implementeres kan de juridiske problemer løses.

En udvidelse af dette er at vi for hver kollegianer har mulighed for at logge dennes trafik.

Kildeidentifikation er endvidere en forudsætning for løsning af de problemstillinger, der behandles i det efterfølgende og vil blive nævnt, hvor det er relevant.

2.5 Administration

Kollegierne ønsker at flytte administrationen bort fra kollegiet. Løsningen på dette er en central administrationsenhed, der besidder den nødvendige tekniske ekspertise. Såfremt den centrale administration organiseres godt kan der opnås stordriftsfordele, hvilket medfører billigere Internet til kollegianerne.

2.5.1 Administrationens opgaver

Administrationen forventes at varetage alle opgaver lige fra teknisk udvikling til administrativt arbejde som registrering af kollegianere, der vil tilsluttes nettet.

2.5.2 Database

Oplysninger om kollegianerne kan passende placeres i en database. Der er mulighed for, at denne kan fyldes ved at samle læreanstaltnes matrikkeldata. Derved kan der spares mange penge til administration.

2.5.3 Distribution

En forudsætning for at opnå økonomiske fordele ved stordrift er, at arbejdsbyrden på det administrative personale vokser langsommere end antallet af kollegier og kollegianere. Dette opnås dels ved at samle mest mulig teknik centralt, dels ved at sikre, at det udstyr, der er placeret decentralt på kollegierne, er ensartet. Kan der implementeres et system, der tilbyder central installation, opdatering og monitorering af alle kollegier

samtidigt opnås, at arbejdsbyrden ved disse opgaver er stort set uafhængigt af antallet af kollegier.

2.6 Stabilitet

Det er et krav fra kollegianerne, at systemet er stabilt og troværdigt. Dette kan under ingen omstændigheder garanteres, men central drift kan øge stabiliteten voldsomt.

2.6.1 Ensartet hardware

Der skal udarbejdes hardware som er ensartet for alle kollegier. Man kan derved have et lager af ekstraenheder, som hurtigt kan bringes ud, autokonfigureres og overtage opgaverne fra en defekt box.

2.6.2 Netværksovervågning

Netværksovervågning af netværksforbindelser og netværksudstyr samt vigtige maskiner sikrer at fejl opdages og derfor kan rettes hurtigt. Kombineret med et centralt vedligeholdelsessystem kan man i nogle tilfælde opnå, at en fejl kun skal optræde eet sted, før den er rettet alle steder.

2.6.3 Server sikkerhed

En forudsætning for stabil drift er, at servere og andet udstyr ikke kan kompromiteres og derved sættes ud af drift af crackere. Central administration kan i en hvis grad nedsætte risikoen for dette ved dels løbende at overvåge maskinerne, dels ved at sikre at eventuelle sikkerhedshuller opdages og lappes hurtigt.

Et Intrusion Detection system vil kunne detektere og stoppe indbrudsforsøg før de får betydning for driften og derved også betyde bedre stabilitet.

2.7 Billigt Internet

Kollegianerne ønsker naturligvis at deres internetadgang er billigst muligt. En måde at opnå dette på som allerede er behandlet er central drift. Derudover kan der spares penge ved at kollegierne køber internetforbindelser samlet, som de kan dele og derved opnå mængderabat.

Endelig er der mulighed for, at grupper af kollegianere kan blive tilbudt gratis Internet. Et aktuelt eksempel på det sidste er studerende ved institutioner, der hører under forskningministeriet, som tilbydes gratis internetadgang via Forskningsnettet.

2.7.1 Deling af internetforbindelse

En forudsætning for at kollegierne kan dele en internetforbindelse er, at dataene kan routes fra kollegierne til det sted, hvor internetforbindelsen rent fysisk er placeret.

2.7.2 Multiple internetgateways

For at et kollegie som helhed skal kunne tilsluttes en internetgateway, der er forbeholdt nogle få kollegianere er det et krav, at der kan skelnes mellem disse. Dette er en del af kildeidentifikationsproblematikken.

Ved indførelse af en internetgateway, som er forbeholdt en delmængde af kollegianerne, er vi nødt til at tilbyde en alternativ gateway til resten af kollegianerne.

Løsningen skal kunne sikre, at alle brugere individuelt bliver tilknyttet og bruger en optimal/korrekt internetgateway.

2.8 Flexibilitet

Med flexibilitet mener vi, at løsningen skal fungere sammen med mest muligt hard- og software, det vil sige at løsningen i mindst mulig grad skal stille krav til den soft- og hardware, der kan anvendes på kollegierne.

Et eksempel på dette er, at vi forlanger, at det er muligt at kontakte kollegianernes maskiner udefra, således at kollegianerne har mulighed for at køre serversoftware og andre applikationer, der kræver dette.

Derudover vil vi kræve, at vores løsning kan udvides til at understøtte nye internetteknologier som QoS, når disse bliver almindeligt tilgængelige.

2.9 Fairness

Et problem når et stort antal brugere skal dele en begrænset båndbredde er at nogle brugere bevidst eller ubevidst udsulter de øvrige brugere som vil opleve at netforbindelsen ikke er responsiv. F.eks. er det normalt at åbne mange samtidige TCP-forbindelser ved filoverførsel for at få tildelt større del af båndbredden. Dette er til stor gene for brugere af realtidsapplikationer som telefoni, shell-forbindelse eller spil.

2.9.1 Trafikaccounting

Derudover må betaling til internetudbydere (dataoverførsel) forventes at være en betragtelig udgift. Det er derfor et ønske, at systemet kan bogføre den enkelte kollegians dataoverførsel. Kildeidentifikation er naturligvis en forudsætning for at dette kan implementeres.

2.9.2 Fair queing

Fair queing er en bred betegnelse for metoder, der kan implementeres på en router for at give brugerne lige deling af båndbredden. Der ønskes lige fordeling mellem kollegianere. Dette forudsætter kildeidentifikation for at kunne blive implementeret fuldt ud, men en næsten optimal løsning kan opnås ved at sikre lige deling af båndbredden mellem afsendercomputer.

2.10 Sikkerhed

Kollegianerne ønsker, at deres computer beskyttes mod trusler fra Internet og andre kollegianere. Under vores forudsætning om autonome netværk er det sidste ikke muligt at implementere fuldt ud.

2.10.1 Firewall

Sikring mod trusler fra Internet kan til en hvis grad opnås ved at tilbyde firewallfaciliteter til brugerne på de autonome netværk. Da firewallingen i høj grad er en afvejning mellem flexibilitet og sikkerhed skal brugerne til en hvis grad selv kunne kontrollere, hvad de vil beskyttes imod.

2.10.2 Intrusion Detection

Log af trafik til og fra Internet giver stort statistisk materiale til at søge efter mønstre, der minder om forsøg på crackning af kollegianernes computere (f.eks. port- og net-scanning). Kan crackningsforsøg detekteres kan de stoppes til fordel for brugerne.

2.10.3 Virusscanning

Man kunne forstille sig, at det i et vist omfang ville være muligt at scanne trafik ind og ud af kollegiet for vira.

2.11 Nemt

Kollegianerne ønsker, at det er nemt at tilslutte sig nettet. Nem tilslutning til nettet medfører også, at der kan spares penge til support.

2.11.1 Autokonfiguration

Ved at tilbyde autokonfiguration af netværks opsætningen på maskiner, som tilsluttes lokalnettene, kan man for mange platforme næsten eliminere behov for særlig opsætning af klienterne. Derved undgår man problemer forårsaget af misforståelser og tastefejl.

2.11.2 Oplysningssystem

Har en kollegianer glemt at logge sig på eller har fejlkonfigureret sin computer skal hun, hvis det er muligt, automatisk informeres herom. Dette kan for eksempel ske ved, at der vises en særlig hjemmeside med en fejlbesked.

2.11.3 Brugerstatus/konfigurations system

Der kan oprettes et interface, hvor brugerne kan få oplyst deres status i forhold til systemet. Dette kunne for eksempel være at se sit eget trafik forbrug i tilfælde af at der er trafikkvote på kollegiet. Derudover kan hun have mulighed for at konfigurere forskellige options, der tilbydes af systemet. Hvis kildeidentifikation er implementeret kan dette bruges til at identificere brugeren, der automatisk vil få vist information, der er relevant for netop hende.

2.12 Hjemmeside, e-mail o.s.v.

Hjemmeside, e-mail og ligende ydelser, som ikke har noget med netværkets opbygning at gøre, men er afhængige af netværket, vil ikke blive behandlet i denne opgave. Vi skal blot nævne, at der også her er mulighed for stordriftsfordele ved at lade kollegierne deles om en serverpark.

Kapitel 3

Analyse og design

Analyse af problemstillinger og design af løsninger

I dette kapitel vil vi behandle udvalgte problemstillinger fra kapitel 2. Vi vil analysere de enkelte problemstillinger, opstille krav til en løsning, give løsningsforslag, udvælge et givent løsningsforslag samt give en designbeskrivelse af det valgte løsningsforslag.

Vores mål er at løse problemstillinger fra den virkelige verden. Vi vil derfor ligge vægt på, at komme med løsningsforslag, som kan benyttes i de konkrete omgivelser. Vi ønsker, at løse problemstillingerne godt, hurtigt, nemt, billigt og vedligeholdelsesfrit. Vi vil derfor i stor udstrækning benytte standardkomponenter til løsning af problemstillinger.

Før vi går igang med at behandle de udvalgte problemstillinger, vil vi bruge lidt tid på, at etablere nogle generelle forudsætninger, krav og antagelser. Først vil vi naturligvis udvælge nogle af problemstillingerne.

3.1 Udvalgelse af problemstillinger

På grund af projektets omfang er det ikke realistisk at behandle alle de identificerede problemstillinger i dette kapitel. Vi har derfor udvalgt fire problemstillinger vi vil gå i dybden med. De øvrige problemstillinger vil vi ikke behandle i dette kapitel, vi vil dog i kapitel 5 komme ind på deres indflydelse på etableringen af det regionale kollegienetværk.

Følgende problemstillinger vil vi behandle *dybdegående*:

- Kildeidentifikation
- Multiple internetgateways
- Åbne/lukke system
- Bifrostmaskinen

Kriterier for udvælgelsen af problemstillingerne har været: kerneproblemstillinger, faglig relevans og interesse.

Vi vil så vidt muligt forsøge ikke at blande de enkelte problemstillinger sammen. Der er dog en vigtig fællesnævner mellem mange af problemstillingerne, som vi vil beskrive først: Fjerndrift. Fællesnævneren omhandler, hvorledes vi på en sikker og forsvarlig måde kan overføre data mellem den centrale administration og de decentralt placerede Bifrost maskiner.

3.2 Overordnede krav og antagelser

Med antagelser og krav, mener vi ting, som vi forlanger af omgivelserne

3.2.1 Autonomt-netværk

I stedet for kollegienetværk benytter vi begrebet *autonomt* netværk. Dette er den mest generelle form for lokalnetværk, vi kan forstille os, at kunne arbejde med.

Om de autonome klienter antager vi:

- De har en TCP/IP protokolstak implementeret (se [11]).
- Mindst et af de mest basale internetprogrammer som f.eks. telnet eller ftp kan afvikles fra klienten.

Da et af vore primære mål er at tilbyde Internetadgang og da Internettet er baserede på TCP/IP, lægger det første krav i praksis ikke nogen restriktioner på netværket.

Da vi ikke gør nogle antagelser om hvilket operativsystem klienterne bruger sikrer vi os dels at løsningen er fremtidssikret, dels at den kan bruges på meget simple klienter som f.eks. printere og håndholdte.

Om netværket antager vi:

- Netværket er ikke routet, det medfører at klienternes hardwareadresse (MAC adresse) kan ses af de datagrammer Bifrost modtager.

Vi kan naturligvis ikke sikre, at en bruger ikke sætter flere maskiner op og foretager routning mellem disse. Vi vil derfor tage højde for dette i analysen og vores løsning skal acceptere eller håndtere dette på en passende måde.

3.2.2 IPv4 vs. IPv6

Internet Protocol version 6 (IPv6, se [19]) løser nemt og elegant nogle af de problemstillinger, vi er stillet overfor. Vi har dog valgt, at se bort fra IPv6 i dette projekt, da den er umuligt at benytte i de konkrete omgivelser. Vores omgivelser inkluderer masser af gammel hard- og software, som ikke understøtter IPv6, men kun IPv4 (se [11]). Vi vil derfor i dette projekt kun benytte teknologier, som kan benyttes i et IPv4 miljø.

3.3 Generelt om fjerndrift

I dette afsnit vil vi gøre nogle mere generelle overvejelser omkring fjerndrift af Bifrost. Designbeskrivelse og mere specifikke valg vil vi udsætte til behandlingen af de enkelte problemstillinger.

Vi har et grundlæggende ønske og behov om fjerndrift af Bifrost. De enkelte Bifrost maskiner er fysisk placeret på de enkelte kollegier, altså decentralt placeret. Vi forestiller os, at der etableres et centralt driftscenter, hvis opgave er at administrere og vedligeholde de decentralt placeret maskiner. Central styring af disse, altså fjerndrift, er en vigtig opgave, som vil kunne udmønte sig i en hurtigere og mere effektiv service over for “kunderne”.

Ved central vedligeholdelse, skal al information ligge centralt. Enhederne skal være ens og næsten selvkonfigurerende via kontakte til den centrale styring. De autoritative filer skal altid forefindes centralt, så den enkelte Bifrost maskine kan genetableres ud fra den centrale information. Data opsamlet eller skabt på Bifrost bør opsamles centralt.

Vi vil indlede dette afsnit med en diskussion af de sikkerhedsmæssige aspekter ved at overføre data mellem den centrale administration og de decentralt placerede Bifrost.

Herefter vil vi beskrive forskellige eksisterende protokoller og værktøjer, som kan benyttes til dataoverførslen mellem centralen og Bifrost.

3.3.1 Netværks sikkerhed

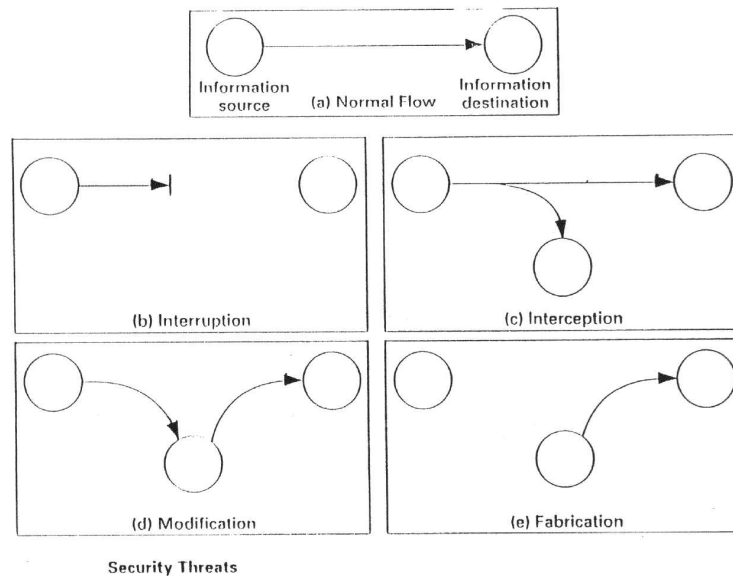
Vores primære kommunikationsvej mellem Bifrost maskinerne og den centrale administration går via 3.-parts netværk. Det er derfor interessant, at undersøge de sikkerhedsmæssige aspekter ved dataoverførsel mellem disse.

For at gøre dette, skal de klassiske netværks trusler overvejes, når vi skal overføre data sikkert og forsvarligt mellem to parter.

Generelle netværkssikkerhedstrusler (se figur 3.1):

- **Opsnapning** af data. (Eng.: *Interception*)
- **Afbrydelse**: Data forhindres i at komme frem. Betegnes normalt “Denial of Service” (DoS) angreb. (Eng.: *Interruption*)
- **Modifikation** af data: Klassisk “man-in-the-middle” problematik. (Eng.: *Modification*)
- **Fabrikation**: Opdigtning eller replay af data. (Eng.: *Fabrication*)

Opsnapning kan foregå på to forskellige måder. (1) Data kan opsnapes, når det sendes over netværket. (2) Data kan hentes direkte fra “serveren”, hvis man udgiver sig for en “klient”, der har lov til at hente data. Situation (1) kan løses ved at kryptere data, som sendes over nettet, således at det er uforståeligt for udenforstående. Situation (2) skal “klienten” autentificere overfor “serveren”, dvs. kunne vise den er den, som



Figur 3.1: Generelle netværks sikkerheds trusler. (Kilde: Klaus Hansens noter, fra sikkerhedskursus)

den påstår. Dette kan ske via en delt hemmelighed eller et "challenge-respons" som kun "klienten" kan svare på (offentlig nøgle systemer er ofte brugt). Vi skal i vores analyse og design, være pragmatiske og overveje, hvilken betydning det har, at 3.part opsnapper og læser vores data. *Hvor fortrolig er vores data?*

Afbrydelse af kommunikationsvejen til Bifrost maskinerne, er vi yderst sårbare overfor. Da kommunikationen går over 3.parts netværk, har vi svært ved at forhindre problemet.

Man kunne overveje, at benytte en alternativ backup-kommunikationsvej til Bifrost maskinerne, men i praksis vurderer vi dette alt for omstændigt og omkostningsfuldt, i forhold til sandsynligheden for afbrydelser.

En anden overvejelse er, at hvis Bifrost maskinens forbindelse til backbonenetværket er afbrudt, kan den alligevel ikke udføre sin funktion som gateway. En partitionering af backbonenetværket er stadigvæk problematisk.

Det er i denne forbindelse vigtigt, i vores design, at overveje, hvor afhængige Bifrost maskinerne skal være af de centrale servere. Vi skulle nødtigt have en situation, hvor alle Bifrost maskiner "stopper", fordi de centrale maskiner bliver utilgængelige.

I vores analyse og design skal vi overveje; *Hvad vores tilgængeligheds krav er.*

Modifikation af data undervejs, imellem afsender og modtager. Vores kommunikationsvej mellem afsender og modtager går igennem 3.parts netværk. 3.part har derfor rig mulighed for at modificere data.

En ofte brugt metode til at modificere data, uden at afsender og modtager opdager noget, kaldes for "man in the middle" metoden. Princippet går ud på, at narre både modtager og afsender til at tro de snakker direkte med hinanden, men i virkeligheden kommunikerer de henover 3.part. Fordelen ved dette angreb, er at krypteret kommu-

nikation også kan modificeres. Dette opnås ved at 3.part dekrypterer, modificerer og krypterer på ny. Dette er kun muligt, fordi 3.part har narret parterne til at etablere krypteringensnøglen med 3.part og ikke hinanden.

Integriteten af data skal selvfølgelig overholdes, og man er derfor naturligvis aldrig interesseret i at ens data modificeres af 3.part.

For at sikre sig mod modifikation, har man behov for en protokol, der sikrer integritet og autentifikation. Det er ikke nødvendigt at kryptere indholdet. Integritet sikres ofte ved brug af secure-hash funktioner. Korrekte autentifikation og verifikation af modparten er også essentiel at etablere, ellers kan vi blive udsat for “man in the middle” angrebet.

Fabrikation af netværksforespørgsler bruges til at snyde modtageren, til at udføre handlinger eller give udvidet adgang.

Hvis protokollen blot er semi-kompliceret, er den nemmeste metode til fabrikation, at “afspille” tidligere opsnappet trafik.

Måden protokoller normalt sikrer sig mod “replay” angreb, er ved at bruge tidsstempler og sekvensnumre.

3.3.2 Eksisterende protokoller og værktøjer

Som tidligere nævnt vil vi helst benytte standardkomponenter ved løsning af problemstillingerne. Vi vil derfor ikke udvikle/designe vores egne protokol til dataoverførslen. Vi vil i stedet benytte eksisterende værktøjer og protokoller.

I de følgende afsnit vil vi diskutere og beskrive eksisterende værktøjer og protokollerne de benytter sig af. Vi vil forsøge at holde beskrivelsen på et overordnet niveau for at begrænse afsnittets omfang og for ikke at komme ind på design og implementations detaljerne.

Vi vil i beskrivelsen fokusere på overførsel af data fra den centrale administration til Bifrost. Ved overførsel i den modsatte retning er problemstilling tilsvarende.

Vi er kun interesserede i at benytte værktøjer og protokoller, der ikke kræver interaktion for at overføre data. Det er essentielt for skalerbarheden, at Bifrost maskinerne er “selvkonfigurerende” eller “selvkørende” uden interaktion fra driftscenteret.

Frem for at stille krav direkte i forhold til de i afsnit 3.3.1 beskrevne sikkerhedstrusler, vil vi i stedet beskrive, hvilke problemer de enkelte værktøjer og protokoller lider under eller løser.

Vi har valgt at kigge nærmere på følgende værktøjer og protokoller:

- SSH og SCP (med SSH-keys)
- Cfd (med SSL/TLS overførelse)
- CVS (via pserver eller SSH)
- Rsync
- IPsec

- FTP (anonymous)
- SNMP

Først vil vi dog se på push og pull mekanismer til overførsel af data.

3.3.3 Push vs. Pull

Når vi snakker om overførsel af filer/data, er der to overordnede begreber vi benytter os af, push og pull, som beskriver hvem, der starter overførslen.

Med vores senarie kan man forklare situationen på følgende måde:

Push er, hvis centralen sender/opdaterer en fil på Bifrost og

Pull er, hvis Bifrost henter en fil som centralen “udbydder”.

Ulempen ved udelukkende at benytte en *Push* mekanisme til distribution, er situationen hvor en Bifrost maskine er utilgængelig mens der push'es, hvorved Bifrost maskinen ikke opdateres. Med *pull* mekanismen kan maskinen opdatere, når den har tid, og i forbindelse med vigtige synkroniserings punkter, såsom ved opstart. Ulempen ved *pull* er, at vi fra centralt hold ikke ved, hvorvidt alle Bifrost maskiner er “opdateret”.

Hvis Bifrost maskinerne, som beskrevet, skal være “selvkonfigurerende”, ved at hente data fra centralen, så har vi behov for en *Pull* funktionalitet.

3.3.4 SSH - secure shell

SSH er en forkortelse for Secure Shell, og giver mulighed for at oprette en krypteret shell adgang til andre maskiner, hvor SSH er installeret. SSH er nærmest blevet et standardprogram på næsten alle UNIX'er, om ikke andet kan man gratis downloade og installere OpenSSH¹. Det er primært shell adgang SSH tilbyder, men fil kopiering og tunneling af trafik er også muligt.

For at SSH kan opfylde vores krav om automatisering, skal man oprette SSH-nøgler, der fungerer som et offentligt nøgle system. D.v.s. man kan tillade login uden brug af password, ved at ligge brugerens offentlige-nøgle på serveren. Ud fra denne kan serveren således autentificere brugeren, ved at verificere om brugeren er i besiddelse af den private nøgle. Denne form for autentifikation giver en rigtig god sikkerhed. Til gengæld er den private nøgle yderst vigtigt at hemmeligholde.

SSH er et værktøj, som kan det “hele”, vi har fil kopiering, tunneler og fuld shell adgang. Med et så kraftigt værktøj, skal vi overveje “trust” modellen, hvem skal have adgang til hvad.

Push funktionalitet: Hvis vi giver den centrale server SSH adgang til alle Bifrost maskinerne, udsætter vi ikke den centrale servers private-nøgle for nogen specielt store trusler. (Alle Bifrost maskinerne stoler på den centrale servers private-nøgle)

¹<http://www.openssh.org/>

Pull funktionalitet: Hvis vi giver alle Bifrost maskinerne SSH adgang til den centrale server, så kan vores private-nøgler på Bifrost maskinerne nemmere kompromitteres. Kollegierne har fysisk adgang til Bifrost maskinerne og kan potentielt læse harddisken, ved blot at tilkoble den fysisk til en anden maskine. (Den centrale server stoler på alle Bifrost maskinernes private-nøgler)

Hvis vi ønsker SSH *pull* funktionalitet fra Bifrost maskinerne, bør dette kun ske via en mindre privilegeret bruger-konto på den centrale server.

3.3.5 Cfd - Cfengine dæmonen

Cfengine er en softwarepakke til vedligeholdelse af konfigurationsfiler (hvilket beskrives i afsnittet omkring konfigurations filer på side 39). Det vi er interesseret i at udnytte i denne sammenhæng, er cfengines dæmon "cfd".

cfd har 2 funktioner:

- Udbyde filer, som kan tilgås fra en remote maskine. Overførelsen kan ske sikkert hen over SSL/TLS (benytter SSLeay libs).
- Remote aktivering, d.v.s. en anden maskine kan bede *cfd* om at udføre et lokalt predefineret script (normalt cfengine.conf).

I vores situation kunne vi lade den centrale maskine udbyde filer via *cfd*, således at Bifrost maskinerne laver et *pull* af filerne. På alle Bifrost maskinerne kunne vi konfigurere *cfd* til at udfører vores pull-script og tillade centralen privilegier til remote aktivering. Herved opnår vi en *pull-on-demand* funktionalitet, som svare til en *push* funktionalitet set for centralen.

3.3.6 CVS - Concurrent Versions System

CVS (Concurrent Versions System) er et versions kontrol system, der holder styr på gamle versioner af filer, og logger alle rettelser med informationer omkring hvem, hvad og hvornår. CVS bruges normalt til software udvikling i grupper. Under hele vores projekt forløb har vi gjort brug af CVS. Den videre udvikling af dette projekt i Kbhkol regi, vil formentligt også benytte CVS, d.v.s. vores egen software vil formenlig allerede ligge i et CVS arkiv.

Det, vi kan udnytte med CVS, er distributions mekanismen. Når man skal rette/læse CVS arkivet, henter man indholdet til sin lokale maskine. CVS kan benytte mange forskellige underliggende autentifikation og transport mekanismer.

- *ext* - Dette kan være rsh, SSH eller ligende. Kravet er blot, at klienten har en måde at udfører kommandoer på serveren, samt input og output sendes via klienten.
- *pserver* - Der findes mange implementationer, ofte med mindre funktionalitet end hele CVS. Brugernavn, password og data sendes i klartekst. Denne mekanisme bruges ofte til at give offentlig læse adgang til CVS arkivet.

- *kserver* - Via en kerberos service.
- Generisk interface, så der er muligt at lave diverse nye autentifikation og transport mekanismer (*pserver* benytter denne).

Hvis Bifrost maskinerne skal have CVS adgang, skal dette, af oplagte sikkerhedsmæssige grunde, kun være læse adgang. Med denne udnyttelse af CVS, har vi kun en *pull* funktionalitet ("On-demand-push" er stadigvæk mulig ved brug af *cfid*).

Problemerne ved brug af SSH, som autentifikation og transport mekanisme, er beskrevet i afsnittet om SSH. Automatisk *rsh* er for usikker. Med *pserver* skal vi overveje hvor fortrolig vores data er.

3.3.7 Rsync

Rsync (se [26]) er et værktøj, som kan benyttes til at synkronisere to mapper. Rsync kan aktiveres således, at synkronisering foregår over et netværk. Rsync kan enten benytte sig af SSH til remote aktivering, eller den kan startes som dæmon, der tilgås af en klient.

Hvis rsync benyttes som en dæmon, er der ingen sikkerhed ved overførsel af data.

Det specielle ved rsync er, at den forsøger at overføre mindst mulig data for at synkronisere de to mapper. Dette gøres ved kun at overføre data der har ændret sig. Rsync er derfor meget funktionel og kan bruges til hurtigt at synkronisere selv meget store datamængder.

Problemerne ved brug af SSH, som autentifikation og transportmekanisme, er beskrevet i afsnittet om SSH.

3.3.8 IPsec - IP security protocol

IPsec adskiller sig væsentligt fra de andre protokoller, ved at ligge på netværks-laget, hvor de andre protokoller benytter applikations-laget. D.v.s. at de andre protokoller kan afvikles oven på IPsec og derved have gavn af alle IPsec's fordele. IPsec tager sig af kryptering og autentificering, og løser derfor problemerne for protokoller der ikke indeholder disse sikkerhedsfunktionaliteter.

Via IPsec kan vi etablere et VPN (Virtual Private Network) mellem Bifrost maskinerne og det centrale driftscenters maskiner. Dog kræver dette væsentlig mere konfiguration end de veletablerede applikations protokoller.

Med IPsec kan man ikke snakke om pull vs. push, da den blot danner grundlag for transport af andre protokoller.

IPsec har også indbygget en autentifikations mekaniske, der kunne benyttes til kildeidentifikation. Grundet den ringe udbredelse af IPsec er dette formenlig ikke muligt i praksis (nærmere beskrevet i afsnit 3.4.3).

De efterfølgende protokol beskrivelse, er generelt for usikre, men ved brug af IPsec er deres sikkerhedsproblemer løst.

3.3.9 FTP - File Transfer Protokol

FTP er en klartekstprotokol, og må betegnes som usikker. Vi skal overveje om vores data er fortroelig. Hvis alle må læse data, er det acceptabelt at give “anonymous” adgang til data via FTP.

3.3.10 SNMP - Simple Network Management Protokol

Den mest anvendte version af SNMP (se [2]) benytter klartekst password (kaldet “community”). Nyere version af SNMP understøtter bedre sikkerhedsfunktioner, men det er de færreste der understøtter de nyere versioner.

SNMP benytter UDP-pakker af den begrænset størrelse, og har kun primitiverne Get, Set og Trap. Vi vil da heller ikke benytte SNMP til filoverførelser. SNMPS primære formål for dette projekt ville være at udbyde trafik tællere af forskellige art til brug for trafik statistik

I kombination med IPsec, kunne man rent teoretisk overveje at benytte SNMP som fjernprotokol til åbne/lukke systemet.

3.4 Kildeidentifikation

Det autonome netværk er befolket med en række computere og en række individer. Da netværket er autonomt kan disse bevæge sig frit. Maskinerne skifter fysisk placering og individer bevæger sig mellem computerne.

Med kildeidentifikation menes, at vi til en given datastrøm ind eller ud af et autonomt netværk ønsker, at kunne identificere det individ på det autonome netværk, som er afsender eller modtager af datastrømmen.

Kildeidentifikation er en forudsætning for løsning af de andre problemstillinger, der behandles i denne opgave herunder multiple gateways og trafikaccounting.

Kravet om klientuafhængighed og flexibilitet vil kun kunne opfyldes, hvis vi kan finde en metode, som udelukkende opererer på netværkslaget (lag 3 i OSI modellen) eller lavere. Dette er vanskeligt, hvorfor vi nogle steder i dette afsnit bliver nødt til, at bøjne krav for at evt. at finde bedre løsninger.

3.4.1 Placering af kontrolpunkt

Bifrost er placeret mellem det autonome- og backbonenetværket. Det vil sige, at vi kan inspicere og ændre data, der strømmer mellem backbone- og lokal-netværket, men at vi ikke har kontrol med, hvad der sker på lokalnetværket (heraf betegnelsen autonomt-netværk).

3.4.2 Identifikation af bruger

I det følgende bruges termen *bruger* om en person, der har fysisk adgang til at bruge lokalnettet. For at skelne mellem disse tilknyttes de en unik identifikation (brugernavn). Målet er at al data, der passerer Bifrost, kan specificeres som oprindende fra eller destineret til en registreret bruger.

3.4.3 IPsec

En forkroment løsning kunne være baserede på IPsec. Det vi ønsker, er autensitet, som er en delmængde af, hvad IPsec (se [5]) tilbyder. For at bruge IPsec skulle alle brugere have en privat nøgle i et offentligt nøglesystem, som de kan bruge til at identificere sig over for Bifrost med. Desværre er IPsec i praksis implementeret på et meget lille antal klienter og vanskeligt tilgængeligt for den almene bruger. Det strider mod vores to mål om, at løsningen skulle være klient uafhængig og nem at bruge. Derudover kan det være vanskeligt for brugerne at tage deres nøgle med, hvis de skal benytte en anden computer midlertidigt.

3.4.4 Proxy med authentication

En mulighed er, at tvinge brugerne til at benytte sig af applikationsproxyservere. Disse kan f.eks. understøtte socks5 (se [8]). Socks5 serveren kan kræve autentikation af klienten, dette kan være ved hjælp af brugernavn og password. Dette vil ikke give brugerne mulighed for at tilgå deres maskiner fra Internet og opfylder derved ikke vores krav om flexibilitet. Derudover kræves applikationer, der understøtter Socks, hvilket også er et problem.

3.4.5 Identifikation baserede på IP/MAC-adresse

I et miljø med fysisk kontrol over brugere og computere ville problemet kunne løses ved at udstyrer hver bruger med en computer og hver computer med et fast kendt IP-nummer. Alle pakker, der modtages af Bifrost fra lokalnettet, har IP-afsenderadresse indkodet. Tilsvarende har alle pakker, der sendes fra Bifrost til lokalnettet, en IP-modtageradresse. På et netværk med fysisk kontrol vil man derfor for alle pakker, der passerer Bifrost, kunne finde afsender-/modtagercomputeren ud fra pakkens IP-adressen og til computeren kan man knytte en bruger.

Denne løsning er ikke hensigtsmæssig på et autonomt netværk af flere grunde. Dels forventes der at være offentligt tilgængelige computere tilsluttet netværket, som ikke kan knyttes til en bestemt bruger, dels kan man forvente at brugerne vil bytte computere. Men mest, fordi brugere med onde hensigter har gode muligheder for at omgå systemet. Dette kan gøres ved at søge efter IP-adresser, som ikke er i brug og bruge dem uden at være registreret.

Man kunne forsøge at begrænse dette ved også at registrere MAC-adresserne på det netværksudstyr brugeren benytter til tilslutning. Dette giver en vis sikkerhed mod brugere, der kommer til at bruge, en forkert IP-adresse ved et uheld, men giver så godt som ingen beskyttelse mod beviste forsøg på at omgå systemet. Et andet problem er, at den ikke kan forventes at fungerer sammen med autokonfigurations metoder som f.eks. DHCP.

3.4.6 Pakkeidentifikation

Metoden i afsnit 3.4.5 lever op til alle krav om brugervenlighed og flexibilitet. Derfor er den sidste metode, vi vil beskrive, en udvidelse af denne.

Kan en IP-adresse til alle tider knyttes til en bruger uanset dennes færden er vort mål nået.

Vi skal altså løbende vedligeholde en tabel, der for hver IP-adresser knytter denne til en bruger. Vi kalder denne tabel for kildetabel.

For at gøre dette indfører vi en konceptuel ind- og udlogning. Ved indlogging identificerer brugeren sig for Bifrost og meddeler hvilken IP-adresse hun bruger. Ved udlogging meddeler brugeren Bifrost, at hun ikke længere bruger en given IP-adresse.

Kan IP-adressen på en datapakke, der vil passere Bifrost, ikke knyttes til en bruger

afvises datapakken.

3.4.7 Login

Oprettelse i tabellen sker ved, at en bruger identificerer sig for systemet. Dette kan generelt gøres ved at brugeren opretter en dataforbindelse til Bifrost, hvorigennem brugernavn og password sendes. Da IP-adressen, som brugeren benytter, i autentikeringsøjeblikket fremgår af dataforbindelsen kan disse knyttes til brugernavnet.

I praksis kan dette ske via en vilkårlig protokol, der giver mulighed for at brugeren kan autorisere sig med brugernavn og password.

Systemet kan give mulighed for autentikering via flere protokoller, således at brugeren selv vælger. Ved at tilbyde autentikering via standartprotokoller som: telnet, ssh, http, ftp, smb o.s.v. opnår vi, at der findes klientsoftware til de fleste platforme.

Ved brug af passende modulopbygget design kan disse metoder supplere hinanden således, at brugeren kan vælge den autentikationsmetode, der passer hende bedst.

3.4.8 Udlogging

Udlogin kan ske på samme metode som indlogging. Ved udlogging er det dog ikke nødvendigt for brugeren at opgive brugernavn og password. Det eneste Bifrost skal vide er, at brugeren ønsker at logge ud, samt hvilken IP-adresse, hun ønsker at logge af fra.

Som beskrevet ovenfor kan IP-adressen findes som kildeadressen på forespørgslen. Udloggingen kan derfor fra brugerens synspunkt f.eks. ske ved, at hun besøger en given hjemmeside.

3.4.9 Automatisk udlogging

Slukker en bruger sin computer uden at logge ud efterlades IP-adressen åben og en ondsindet person kan potentielt overtage den og derved udgive sig for at være brugeren, der ikke har logget ud. F.eks kan brugeren glemme at logge ud eller hendes computer kan gå i stykker, så hun ikke har mulighed for at logge ud.

For at undgå dette, er det hensigtsmæssigt at indføre automatisk udlogging af en bruger, når hendes maskine slukkes eller frakobles netværket. Fungerer den automatiske lukning godt, behøver brugerne ikke at logge af, når en computer forlades, men kan nøjes med at slukke den.

Ophør af brug af en IP-adresse kan detekteres på flere måder. Der er stor forskel på, hvor hurtigt de forskellige metoder kan forventes at reagere. Reaktions tiden er af betydning for sikkerheden, da vi i værste tilfælde kan risikere, at en ondsindet bruger afventer at en anden bruger slukker sin maskine og overtager dennes IP-adresse inden den automatiske udlogging når at reagere.

For alle metoder gælder det, at de er en afvejning mellem sikkerhed og brugervenlighed.

- *timeout* kræver nyt “login” efter en given tidsperiode. Det må forventes, at være til stor gene for brugeren at blive afbrudt med jævne mellemrum for skulle logge ind. Derudover giver metoden ikke mulighed for at brugeren kan kontakte sin maskine fra Internet, da hun vil blive logget af, når hun forlader maskinen. Fordelen med metoden er, at hvis en ondsindet bruger overtager IP-adressen, vil denne kun kunne bruges, i en tidsbegrænset periode, indtil der kræves nyt login.
- *aktiv trafik*, hvis der ikke sendes noget trafik fra IP'en igennem firewallen, i et givent tidsinterval, antager vi at forbindelsen skal lukkes. Denne metode har, som *timeout* metoden, det problem, at den ikke giver brugeren mulighed for at tilgå sin maskine fra Internet. Den er tilgængelig meget langsom til at reagere og i det tilfælde at en ondsindet overtager IP-adressen kan denne bruges ind til brugeren igen tænder sin computer.
- Brugeren skal have en *aktiv TCP-forbindelse* til firewallen. Når denne forsvinder betegnes det som “logout”. Dette vil i praksis betyde at brugeren skal logge på via en TCP-forbindelse. Så snart TCP-forbindelsen afbrydes betragtes det som logout.

Denne metode giver et højt niveau af sikkerhed, da man typisk kan forventet, at alle TCP-forbindelser bliver afbrudt ved lukning af computeren. Skulle en forbindelse ikke blive lukket vil den stadig være vanskelig at overtage.

Denne metode har en række problemer. Den forudsætter, at indlogging foregår via TCP og at klienten holder forbindelsen åben. Dette udelukker en række af de nævnte standart klienter, der kan bruges til indlogging, herunder http-browsere. Derudover kræver den, at brugeren har en applikation kørende, det kan være generende for brugeren.

Et potentielt problem ved metoden er, at det er resourcekrævende for Bifrost maskinen at opretholde disse TCP-forbindelser.

- *ICMP ping* [12], brugers IP-adresser pinges. Ved manglende svar antages at brugeren ikke længere benytter IP-adressen og der foretages logout. Denne metode er ikke helt klient-uafhængig, da man kan forestille sig klienter, der ikke svarer på ping (praktisk problem med “personlige” firewalls under windows). Tilgængelig er den fuldstændig uafhængig af, hvilken metode, der bruges til åbning. Sikkerheden hænger sammen med hvor ofte Bifrost kan pinge brugerens maskine. Vi har dog ikke mulighed for at sikre, at en ond bruger laver et program, der pinger oftere end Bifrost.

Man kan eventuelt vælge at kombinere løsningerne. *Timeout* eller *aktiv trafik* kan benyttes ved offentlige maskiner, da man derved også beskytter brugeren mod at forlade maskinen uden at logge ud eller slukke. *ICMP ping* giver størst flexibilitet for brugerne og kan benyttes for personlige maskiner.

3.4.10 Sikkerhed

Vi antager, at det ikke er muligt for en ondsindet bruger at overtage en andens IP-adresse mens, denne bruges. Det vil i hvert tilfælde ikke umiddelbart være muligt uden

at genere andre og derved blive opdaget.

Det største sikkerhedsproblem ser ud til at ligge i tidrummet fra en bruger er holdt op med at bruge sin IP-adresse til brugeren logges ud.

Registrering af MAC-adresse

En mulighed for at forbedre sikkerheden er ikke kun registrere brugerens IP-adresse men også registrere MAC-adressen.

Effektiviteten af denne metode kan diskuteres. Den ondsindede bruger kan hurtigt omgå dette ved udover at skifte IP-adresse også at skifte MAC-adresse.

Det vil formodenligt være mere effektivt at bruge MAC-adressen til at detektere misbrug. Bliver en IP-adresse brugt i forbindelse med to forskellige MAC-adresser inden for et kort tidsrum er der sandsynligvis noget galt. En forudsætning er, at den onde bruger ikke ved, at vi holder øje med MAC-adresserne. En anden ulempe ved detektion frem for prevention er, at det kræver dyre mandetimer at reagere på en alarm.

Det gælder, at disse metoder kun i begrænset omfang er anvendelige på et routet netværk, da det kun vil være routerens MAC-adresser, der er tilgængelige fra Bifrost.

Snifning

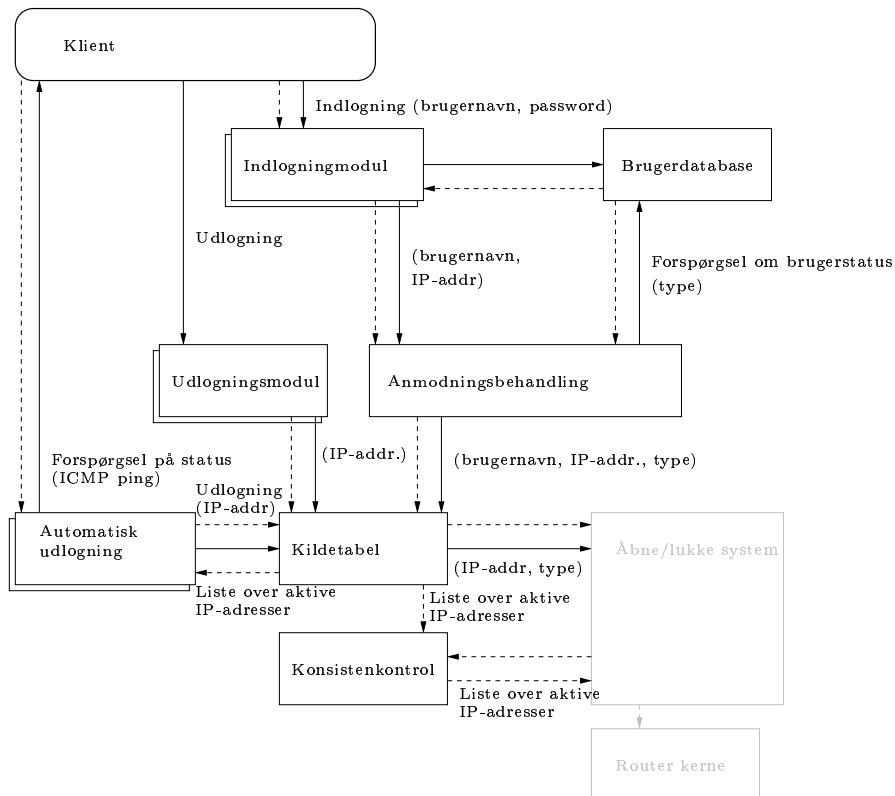
Vi har ind til videre valgt helt at se bort fra, at det autonome netværk kan give mulighed for at en ondsindet bruger kan "sniffe" netværkstrafikken og derved opsnappe andre brugeres password. Dette kan undgås ved kun at tilbyde autentikeringsprotokoller, der tilbyder kryptering såsom https og ssh. Dette vil dog i en eller anden grad gå ud over platform uafhængigheden.

3.4.11 Design

Vi ønsker, at vores løsning let kan udvides med nye måder at implementere automatisk lukning og nye protokoller til ind- og udlogning. Vi lader derfor vores løsning være modulopbygget så vidt det er muligt. For at opnå at vores løsning er mindst mulig afhængig af den omkringliggende soft- og hardware lader vi kommunikation med routeren foregå via et abstraktionslag, vi har valgt at betegne som *åbne/lukke system*. Dette beskrives nærmere i afsnit 3.7. I dette afsnit vil vi blot opfatte åbne/lukke systemet som et API med to kald:

- `open(IP-adresse, MAC-adresse, tilstand/type)`
- `close(IP-adresse)`

Hvor *tilstand/type* angiver parametre for, hvordan brugerens trafik skal behandles, når den passerer firewallen. Det kan for eksempel være, hvilken type firewall brugeren ønsker og hvilken gateway, der skal benyttes.



Figur 3.2: De overordede moduler i implementeringen af Bifrost. Stiplede pile angiver datastrøm. Fuldt optrukne pile angiver retningen af ‘kald’ — det kan være funktionskald i en API eller kald over netværk. På figuren er vist et automatisk udlogningsmodul, der anvender ICMP ping som bekræftet i afsnit 3.4.9. Et automatisk lukningsmodul, der anvender ‘aktiv trafik’ vil kommunikere med Åbne/lukke modulet frem for klienten.

Det er oplagt at lade indlogging, udlogging og automatisk udlogging være selvstændige moduler. Disse moduler skal kun indeholde den funktionalitet, der er typisk for modulet.

Hvorledes modulerne, der forklares i det følgende, samarbejder, er skitseret i figur 3.2. Indlogningsmodulernes opgaver er at:

- tage imod brugernavn og password
- finde IP- og MAC-adresse
- kontrollere brugernavn og password
- sende beked om indlogging videre

Man kan diskutere om kontrol af brugernavn og password skal ligge i indlogningsmodulerne. Ved at lade kontrollen være en del af modulet, opnår vi at der findes mange standartprotokoller, hvortil der er implementeret servere, som kan varetage modulets opgaver.

Indlogningsmodulet lader vi samarbejde med et modul vi kalder *admodningbehandling*. Dette moduls opgave er, at indhente information fra relevante steder og tage stilling til, hvordan brugerens trafik skal behandles. Dette kan være om brugeren ønsker firewallfunktionalitet, hvilken internetgateway brugeren skal benytte og om brugeren skal nægtes adgang på grund af for eksempel misbrug af nettet. Denne ekstra information om brugeren bliver lagt i det, der på figuren er benævnt *type*, og sendt videre sammen med brugernavn, MAC-adresse og IP-adresse.

Udlogningsmodulerne har samme funktionalitet som indloggingmodulerne, de skal blot ikke tage imod og kontrollere brugernavn og password. Da MAC-adressen kan udledes fra IP-adressen, er modulernes eneste opgave at finde IP-adressen og sende den videre.

Modulerne, der varetager automatisk udlogging skal bruge en liste over IP-adresser på brugere, der er logget ind. Der er ikke gjort nogen antagelser om modulernes virkemåde. Modulet skal lige som udloggingmodulet give en IP-adresser, når det har fundet en bruger, der skal logges af.

Vi definerer et modul, som vi kalder *kildetabel*, hvis opgave er, at vedligeholde kildetabellen og *pushe* ændringerne i denne til kerneabstraktionsmodulet. Ændringerne i kildetabellen modtages fra ind- og udlogningsmodulerne.

Vi kommer frem til at kildetabelmodulet skal indholde følgende kald:

- `indlogging(brugernavn,MAC-addr., IP-addr., type)`
- `udlogging(IP-addr.)`
- `dump_aktive()`, giver en liste over aktive brugere og IP-adresser.

For at sikre konsistens mellem kerneabstraktionsmodulet og kildetabellen, kan det blive nødvendigt at implementere et konsistenskontrolsmodul som med et fast interval tager et udtræk af listen af aktive IP-adresser fra kildetabelmodulet og åbne/lukke systemet for at sammenholde disse og korrigere, hvor det er nødvendigt.

3.5 Multible internetgateways

I dette afsnit vil vi analysere et autonomt netværks muligheder for at benytte multible internetgateways. Vi vil herunder se på, hvilke muligheder der er for, at valget af gateway sker på baggrund af kilde, trafikkarakteristika, tidspunkt, mm.

3.5.1 Hvorfor benytte multible internetgateways

Der kan være mange grunde til, at et autonomt netværk ønsker at benytte sig af multible gateways til Internet. Af grunde kan nævnes:

- Redundans
- Kapacitet
- Hastighed
- Pris
- Tilhørsforhold

Som eksempel kunne man tænke sig et autonomt netværk med to forbindelser til Internet. En langsom forbindelse, som er billig per trafikmængde, og en hurtig forbindelse, som er dyrere per trafikmængde. Det autonome netværk vil så kunne fordele deres trafik mellem disse to forbindelser afhængig af trafikkarakteristika. Store filoverførsler kunne eksempelvis sendes via den langsomme/billige forbindelse, mens f.eks. interaktive data kunne sendes over den hurtige, men dyrere, forbindelse.

Et andet konkret eksempel på brug af multible internetgateways er forskningsnettets tilbud om at give studerende ved forskningsnetinstitutioner (institutioner tilknyttet forskningsnettet) gratis adgang til forskningsnettets internetforbindelse. Det autonome netværk kunne således have to internetgateways: en til forskningsnettet og en til en anden internetudbyder. Trafik fra studerende ved forskningsinstitutioner kan så gratis sendes via forskningsnetforbindelsen, mens den øvrige trafik må sende via den anden internetudbyder.

3.5.2 Valg af gateway per kilde, trafikkarakteristika mm.

Som det ses af forrige afsnit, kan der være gode grunde til at ville fordele trafik imellem multible internetgateways. Valget af gateway kan ske ud fra mange forskellige parametre, eksempelvis:

Kilde Internetgateways kan nemt være forbeholdt bestemte kildegrupper.

Trafikkarakteristika Der er et stort antal delparametre i en pakkeheader, trafikken kan karakteriseres ud fra, f.eks. TOS (type of service), applikationsprotokoltype. Valg af gateway kan ske ud fra disse (se figur 3.3).

Belastning Belastningen af forskellige internetgateways kan være en god parameter for valg.

Hastighed Hastigheden af internetgateways er også en god parameter for valg.

Pris Prisen per trafikmængde har betydning for mange.

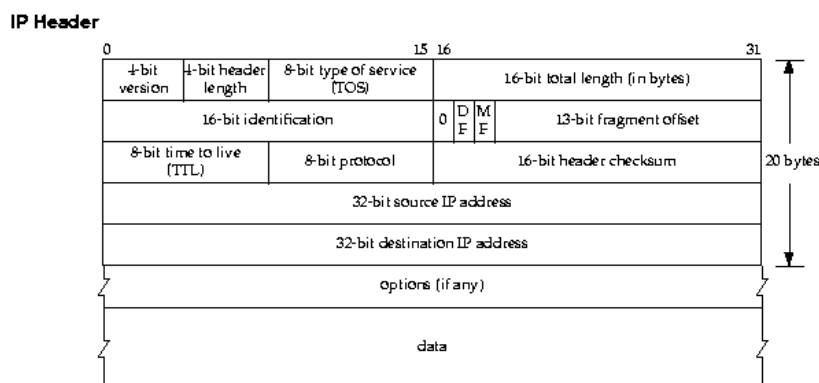
Tidspunkt Om natten er der langt færre, der benytter Internet forbindelsen.

Mange af disse parametre er kun kendt af Bifrost. Det vil derfor være mest hensigtsmæssigt, hvis valg af gateway foretaget på Bifrost er transparent for kilden. Kilden skal selvfølgelig have mulighed for at konfigurere hvordan valg skal foregå, men selve valgene bør ske automatisk og transparent for kilden.

3.5.3 Hvor ligger de tekniske problemer

Hvorfor er multiple internetgateways et problem? Hvorfor kan man ikke bare vælge imellem dem? Disse spørgsmål vil vi her prøve at give et svar på, ved at give en teknisk forklaring på hvordan routing virker under IPv4 (Internet Protocol version 4, se [11, 1]).

På et IPv4 netværk tages der fornyet beslutning om routning i alle routere (knuder i netværket) når en pakke passerer. Dette benævnes på engelsk “Hop-by-hop routing paradigm”. Normalt undersøges modtageradressen og der vælges en vej ud fra denne. Valget sker ud fra en routetabel. Denne tabel indeholder information om hvilken vej der skal vælges for at nå en bestemt IP-adresse. Det vil sige, at det er modtageradressen, som bestemmer en pakkes vej igennem Internet.

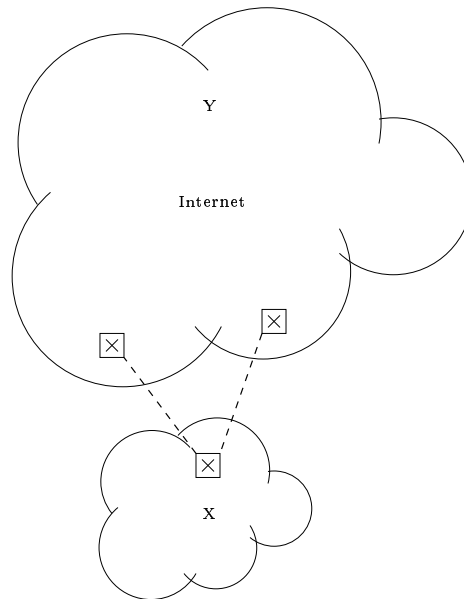


Figur 3.3: IPv4 pakkeheader, fra [23]

På figur 3.3 ses headeren af en IPv4 pakke.

Sendes en pakke fra afsenderadresse X til modtageradresse Y, vil en svarpakke fra Y følge en vej som er bestemt ud fra X. Dette er ikke nødvendigvis er den samme vej som pakken nåede Y via.

Figur 3.4 viser et simpelt senarie med multiple gateways.



Figur 3.4: Simpelt senarie med multiple internetgateways.

Ved på Bifrost at foretage routing baseret på andre parametre end modtageradresse er der ikke noget problem i at få trafik fra X til Y til at blive routet via den valgte internetgateway. Problemet er, at svarpakker fra Y til X kommer tilbage via den internetgateway, hvortil routerne på vejen, mener X er tilsluttet. Der foregår altså en asynkron routing.

Der er umiddelbart to måder at omgå disse problemer på:

- Ændre på indholdet af routetabellen på alle de mellemliggende routere.
- Benytte en afsenderadresse, som i de mellemliggende routere svarer til den internetgateway, som ønskes benyttet.

3.5.4 Krav til funktionalitet

For at give størst mulig funktionalitet, vil vi her opstille nogle krav til, hvad vi mener et system til valg mellem multiple internetgateways skal opfylde:

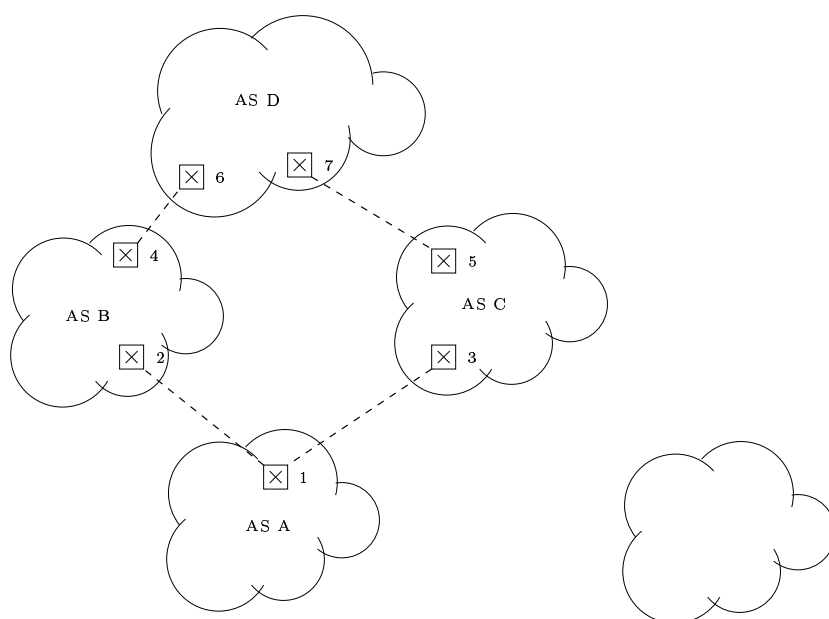
- Hver kilde skal kunne vælge frit mellem internetgateways.
- Mulighed for tilslutning fra Internet til maskine på lokalnettet.
- Må gerne fungere transparent for kilden.
- Må gerne give mulighed for at benytte flere internetgateways samtidig til forskellige former for trafik.
- Skift mellem internetgateways skal foregå relativt hurtigt (eventuelt umiddelbart).

I det følgende vil vi beskrive nogle forskellige teknologier, som på forskellig måde kan give mulighed for valg mellem multiple internetgateways.

3.5.5 Border Gateway Protocol, BGP4

Border Gateway Protocol 4, BGP4 (se [15, 14, 24, 25, 3]), er en protokol, som kan benyttes til at udveksle routeinformation imellem routere. BGP4 er en såkaldt “Exterior Gateway Protocol” i modsætning til “Interior Gateway Protocol” (se [9] for et eksempel). BGP4 benyttes på størstedelen af Internet til udveksling af routeinformation imellem de forskellige parter på Internet. Internt i de enkelte parters netværk benyttes normalt en “Interior Gateway Protocol”.

På figur 3.5 ses en simpelt senarie med syv BGP routere.



Figur 3.5: Simpelt senarie med syv BGP4 routere.

BGP benytter begrebet “AS”, eller “Autonomous System”, for hver af de logisk adskilte parter i netværket. På figur 3.5 er der således syv forskellige AS’er.

Routeinformation udveksles ved at “nabo”-routere opretter TCP-forbindelser imellem hinanden. På figur 3.5 oprettes der således en TCP-forbindelse mellem følgende routerpar 1-2, 1-3, 4-6, 5-7. Desuden oprettes der normalt også en TCP-forbindelse imellem BGP-routere internt i et AS, altså imellem 2-4, 3-5 samt 6-7. Dette gøres for give enkelte BGP-routere et ens billede af routeinformationen i et AS.

Et AS kaldes en “multihomed AS”, hvis der er forbindelser til mere end en AS og der samtidig er forbud mod transit trafik. Et AS kaldes et “transit AS”, hvis der er forbindelse til mere end en AS og transit trafik er tilladt (eventuelt med visse begrænsninger).

Routeinformation udveksles således kun med “nabo”-routere, og en route vil derfor propagere med en vis forsinkelse gennem Internet.

En BGP-router har mulighed for at annoncere router til dens “nabo”-routere. Disse router indeholder blandt andet følgende:

IP-Prefiks IP-præfiks for de IP-adresser routen dækker (se [13, 10]).

NEXT-HOP IP-nummer på den router som skal modtage trafik for IP-prefikset.

AS-PATH Lister over de AS'er routeinformationen har været igennem.

Routerne kan også tilbagekaldes, hvis det bliver nødvendigt.

En BGP-router behandler alle indkommende router i forhold til en lokal politik og sender de behandlede router videre til dens “nabo”-routere. På denne måde propagere routeinformation rundt på Internet. For at mindske antallet af router foretages aggression af router, hvis det er muligt.

Umiddelbart lyder det som om BGP4 løser problematikken med at kunne vælge mellem multiple internetgateways. Der er dog et større problem ved at benytte BGP4 til dette.

Hvis valg af internetgateways skal kunne foretages frit, betyder det, at der skal annonceres en route for hver eneste IP-adresse. Dette er et stort problem for antallet af router i BGP-routerne rundt om i verden. Det er op til hver enkelt BGP-router, om den ønsker at acceptere en routeinformation, og om den ønsker at sende routeinformationen videre. Router til enkelt IP-adresser bliver derfor normalt ikke accepteret eller sendt videre.

Vi har, ved samtaler med teknikere hos internetudbydere i Danmark, fået oplyst, at kun router med et IP-præfiks mindre end eller lig med /20 med garanti bliver accepteret af alle BGP routere i verden. Routeinformationen skal altså gælde mindst 4096 sammenhængende IP-adresser.

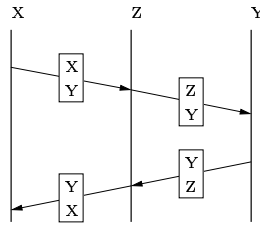
Det er derfor ikke muligt, at benytte BGP til frit at vælge imellem forskellige internetgateways. BGP giver heller ikke mulighed for at benytte flere internetgateways samtidig.

3.5.6 Skift af IP-adresse på klient

I afsnit 3.5.3 kom vi frem til, at en måde at løse problemstillingen ved valg af internetgateway er at benytte en IP-adresse som i forvejen bliver routet som ønsket.

Der er flere problemer ved gøre dette direkte på klienten:

- Det er besværligt at skifte internetgateway.
- Det tager tid at skifte internetgateway.
- Det er ikke muligt at gøre transparent for kilden.
- Det giver mere end en IP-serie på lokalnettet. Dette kan give problemer med en del protokoller f.eks.: SMB (CIFS), rwho.
- Det giver problemer ved autokonfiguration. Fra hvilken IP-serie skal en IP-adresser tildeles.



Figur 3.6: Skematisk afbildning af NAT's virkemåde. Tiden er afbildet langs den vertikale akse. De tre involverede maskiner X, Y og Z er symboliseret med fede vertikale streger. Datagrammerne er symboliseret med kasser, hvor afsenderadresser er øverst og modtageradressen er nederst.

- Det øger behovet for IP-adresser.
- Det giver større administration af IP-adresser.

Skrift af IP-adresse direkte på klinerne er derfor ikke en god løsning.

3.5.7 Network Address Translation, NAT og NAPT

Network Address Translation (NAT, se [21, 20]) giver mulighed for at lave en oversættelse af IP-adresser, når trafik passerer igennem f.eks. Bifrost. NAT kan derfor benyttes til at give trafik en ny afsender IP-adresse som bliver routet som ønsket.

På figur 3.6 ses en skematisk afbildning af hvordan NAT virker. En pakke modtages på Bifrost (Z) fra lokalnettet, afsenderadressen omskrives og pakken sendes ud på Internet. Når der kommer svar fra Internet omskrives adressen igen til den oprindelige. Det er så godt som transparent for kilden, at der sker denne adresseoversættelse. Der er dog visse protokoller, som ikke fungerer godt med NAT (se [4]), og det er derfor ikke helt uproblematisk at benytte sig af NAT. Det drejer sig f.eks. om en protokol som H.323 som benyttes til "voice over IP".

NAT laver en "en til en" adresseoversættelse, det vil sige at en bestemt adresse på lokalnettet svarer til en bestemt adresse på Internet. Denne mapping kan både fungere dynamisk eller statisk.

Der findes en variant af NAT kaldet NAPT (også kendt som masquerading og PAT, se [21]), som ikke laver en "en til en", men en "mange til en" oversættelse. NAPT er relativt avanceret og benytter sig af en statutabel over åbne forbindelser sammen med en oversættelse af portnummeret for at opnå en "mange til en" opsættelse. Der er mange protokoller som ikke umiddelbart virker ordenligt med NAPT. I de fleste tilfælde er der dog mulighed for at udbygge NAPT med protokolafhængige moduler som sikrer at metoden virker. En vigtig egenskab ved NAPT er, at forbindelser kun kan etableres i en bestemt retning, fra lokalnettet ud og til Internet.

"Mange til en" oversættelsen mindsker behovet for IP-adresser, men giver som sagt flere komplikationer med protokoller som ikke virker.

En løsning baseret på NAT giver følgende egenskaber:

- Mulighed for direkte tilslutning fra Internet til maskine på lokalnettet.
- Hver kilde frit valg mellem internetgateways.
- Fungerer transparent for kilden.
- Til en hvis grad mulighed for at benytte flere internetgateways samtidig.
- Skift mellem gateways kan ske umiddelbart.

og følgende problemer:

- Kan give problemer med DNS, da en maskine reelt har flere IP-adresser: En på lokalnettet og en på Internet.
- Har problemer med enkelte applikationsprotokoller.
- Øger behovet for IP-adresser.

En løsning baseret på NAT giver følgende egenskaber:

- Frit valg mellem internetgateways, for hver kilde.
- Fungere transparent for brugeren.
- En hvis grad mulighed for, at benytte flere internetgateways samtidig.
- En hvis grad af sikkerhed for maskinen på lokalnettet, da det ikke er muligt for maskiner på Internet at etablerer forbindelser til den.
- Skift mellem gateways kan se umiddelbart.
- Er meget økonomisk med IP-adresser.

og følgende problemer:

- Giver *ikke* mulighed for direkte tilslutning fra Internet til maskine på lokalnet.
- Har generelt problemer med en del applikationsprotokoller.

Umiddelbart lader en løsning baseret på NAT til at være både brugbar og funktionel. Eventuelt kan løsningen kombineres med NAT for at mindske behovet for IP-adresser.

Der er dog et mindre problem med DNS-opslag og med enkelte applikationsprotokoller. Problemet med DNS-opslag kan løses ved at benytte en DNS-extension til NAT-modulet der ændrer på indholdet af DNS svarpakker, der er på vej ud gennem NAT-routeren (se [22]). Denne extension kræver dog, at DNS-serveren er placeret på lokalnettet. Problemet med de problematiske applikationsprotokoller kan delvis løses ved at lave protokolafhængige extension til NAT modulet.

3.5.8 Design af mulig løsningsmodel

Sammenholder man fordele og ulemper ved de beskrevne teknologier, lader det til at en løsning baseret på både NAT, NAPT med DNS extension er den mest funktionelle.

For at beskrive en sådan løsning nærmere er det først nødvendigt at kortlægge behovet for IP-adresser:

- NAPT kræver, at hver internetgateway har minimum en IP-adresse tilknyttet.
- NAT kræver, at hver internetgateway har et antal IP-adresser tilknyttet. Antallet afhænger af hvor mange, der skal benytte internetgatewayen via NAT.
- Lokalnettet kræver en IP-adresse per maskine. Lidt mindre kan dog gøre det, hvis ikke alle maskiner kører samtidig og der benyttes en autokonfigurationsprotokol som DHCP.

Et stort spørgsmål er, hvilken type IP-adresser, der skal benyttes på lokalnettet. RFC1918 [16] definerer nogle grupper af IP-adresser, som kan benyttes frit internt i organisationer, men som ikke bliver routet på Internet. Disse benævnes ofte som private IP-adresser.

Vælger vi at benytte private IP-adresser på lokalnettet, har vi ikke problemer med at få det antal vi har behov for. Det kræver dog, at vi benytter NAT eller NAPT på al trafik fra lokalnettet til Internet.

En alternativ løsning er at benytte IP-adresser, som er tilknyttet en af de internetgateways, det autonome netværk benytter sig af. Dette giver mulighed for, at trafik fra lokalnettet til Internet kan sendes uden brug af NAT og NAPT. Valget af internetgateway er dog i denne situation lagt fast, men man kunne tænke sig at dette kunne benyttes i situationer, hvor en applikationsprotokol ikke virker med NAT eller NAPT.

Da det i mange tilfælde er svært at få alle de IP-adresser man ønsker, vil vi lade det være op til den konkrete situation, om der skal benyttes private IP-adresser eller normale.

NAT kan benyttes på to forskellige måder:

Statisk Der eksisterer en fast entydig korrespondance (mapping) mellem en IP-adresse på lokalnettet og en IP-adresse på en internetgateway.

Dynamisk Der eksisterer ikke nogen fast entydig korrespondance (mapping), men der oprettes en midlertidig korrespondance i det øjeblik en bruger logger på Bifrost (se afsnit 3.4 for information om indlogging)

Statisk mapping kræver et stort antal IP-adresser, da hver internetgateway skal have IP-adresser til alle brugere som må benytte gatewayen. Statisk mapping sikrer, at en given bruger ved valg af en given internetgateway altid har den samme IP-adresse på Internet.

Dynamisk mapping kræver langt færre IP-adresser per internetgateway, da der kun skal være IP-adresser til det maksimale antal samtidige brugere. Dynamisk mapping betyder, at en bruger ikke har en fast IP-adresse på Internet.

Statisk og dynamisk mapping af adresser i NAT kan fungerer samtidig.

Tildelingen af IP-adresser på lokalnettet kan også ske statisk eller dynamisk. Dynamisk tildeling kræver mindst administration for alle parter og er derfor at fortrække for hovedparten af brugerne. Dynamisk tildeling af IP-adresser betyder, at det ikke umiddelbart er muligt for en bruger at kende en anden brugers IP-adresse. For at løse dette bør etableres et nogle DNS-services på det autonome netværk:

maskinnavn.kollegienavn.kbhkol.dk Det bør til en hver tid være muligt at lave et navneopslag på maskinnavnet og få oplyst den dynamisk tildelte IP-adresse. Dette lader sig nemt gøre ved brug af en udvidelse til DHCP-servicen. I windows netværk kan WINS også benyttes til dette.

brugernavn.kollegienavn.kbhkol.dk Det bør til en hver tid være muligt at lave en navneopslag på brugernavnet og herved få oplyst IP-adressen, hvorfra brugeren sidst er logget ind. Dette bør være nemt at samkøre med kildeidentifikationssystemet beskrevet i afsnit 3.4.

Disse to DNS-services giver hver bruger mindst to entydige FQDN's (Fully Qualified Domain Names), og det bør derfor ikke være problematisk for brugeren at have en dynamisk tildelt IP-adresse.

For også at give brugerne entydige FQDN's på Internet, er det nødvendigt at benytte den omtalte DNS extension til NAT og NAPT. Herved får brugerne FQDN's som er unikke for hele Internet.

Vores løsning skal for alle internetgateways tilbyde følgende opsætninger:

- Dynamisk IP-adresse på lokalnettet, NAPT.
- Dynamisk IP-adresse på lokalnettet, NAT med dynamiske adresser.
- Dynamisk IP-adresse på lokalnettet, NAT med statiske adresser.
- Dynamisk IP-adresse på lokalnettet direkte tilsluttet en bestemt internetgateway.
- Statisk IP-adresse på lokalnettet, NAPT
- Statisk IP-adresse på lokalnettet, NAT med dynamiske adresser.
- Statisk IP-adresse på lokalnettet, NAT med statiske adresser.
- Statisk IP-adresse på lokalnettet direkte tilsluttet en bestemt internetgateway.

Den direkte tilslutning til en internetgateway fungerer selvsagt kun med en bestemt internetgateway per kollegie og kun hvis IP-adresserne på lokalnettet er tilknyttet denne internetgateway.

Brugeren skal have mulighed for at konfigurere hvilken af disse opsætninger han ønsker at benytte. Standartopsætningen bør være: *Dynamisk IP-adresse på lokalnettet, NAT med dynamiske adresser*. Dette vil sammen med de omtalte DNS services give standardbrugeren en flexibel internetopkobling, hvor følgende er opfyldt:

- Frit valg af internetgateway.
- Mulighed for at benytte langt de fleste almindelige applikationsprotokoller.
- Mulighed for et etablerer forbindelser fra Internet til maskinen på lokalnettet.
- Unikt FQDN, der opdateres løbende i forhold til den aktuelle situation.

3.6 Bifrostmaskinen

I dette afsnit vil vi beskæftige os med Bifrostmaskinen og den centrale vedligeholdelse af denne.

De primære komponenter der skal overvejes i forbindelse med Bifrost er:

- Operativsystem
- Softwarevedligeholdelse
- Konfigurationsfiler

Ved design af Bifrost vil vi stræbe efter to mål:

1. Høj opetid
2. Billig drift og etablering

De to mål er som udgangspunkt modstridende, hvorfor det på mange måder er et spørgsmål om prioritering. Vi kan dog i mange tilfælde udnytte stordriftsfordele til at optimere begge ting samtidig.

3.6.1 Operativsystem

Vi vil med det samme ligge os fast på *Linux som OS platform*.

Windows platformen er afskrevet på forhånd, grundet den meget ringe mulighed for fjernstyring og fjernkontrol. En vigtig byggeklods for os er trafikfiltrering, hvilket Windows ikke har "indbygget". Windows er ikke OpenSource, hvilket resulterer i manglende fleksibilitet på mange områder. Uden at gå i detaljer er dette en fleksibilitet vi har behov for.

Vi kunne uden tvivl have valgt en af de andre UNIX kloner som OS. Eksempelvis ville OpenBSD, NetBSD eller FreeBSD være nogle ude mærkede alternativer, der også opfylder vores behov for trafikfiltrering, fleksibilitet og fjerndrift.

Grunden til at valget så prompte falder på Linux, er primært at alle gruppens medlemmer har størst erfaringer med Linux, fremfor de andre UNIX kloner. Desuden har vi alle positive praktiske erfaringer med Linux som firewall og router. I skrivende stund er alle DIKU's firewalls Linux baserede maskiner ², og Grønjord- og Egmont-kollegiets firewalls er ligeledes Linux baserede ³.

²konfigureret og udviklet af medforfatter til denne rapport Jesper Dangaard Brouer

³konfigureret og udviklet af de andre medforfattere til rapporten Per Marker Mortensen og Erik Bidstrup

Distribution

Valget af Linux distribution er ikke specielt vigtig for afviklingen af vores system. Hvilken Linux kerne den understøtter er dog et krav, da det hænger sammen med den indbygget firewall funktionalitet.

Det, der er relevant, er hvor nemt distributionen kan vedligeholdes. Driftsorganisationen er selvfølgelig interesseret i så nem vedligeholdelse som muligt. Tilgængeligheden af sikkerheds-updates fra OS-distribution er her en faktor. Nem, hurtig og evt. automatisk installation er en anden.

Vi har kigget på et svensk projekt, med et pudsigt navnesammenfald, de kalder deres Linux distribution for “Bifrost”⁴.

Distributionen overholder alle de krav vi har, når vi skal sætte en Linux firewall op. Deres projekt beskrivelse stemmer overens med alle vores behov.

Citat fra deres hjemmeside:

The goal of this project is to find out stability, performance, filter capabilities, administration, computer security, scalability and development possibilities of a Linux based streamlined router/firewall system.

3.6.2 Softwarevedligeholdelse

Softwaren vi installerer, må som udgangspunkt forventes at være behæftet med fejl og mangler. Det er derfor nødvendigt, at vi kan opdatere denne nemt og derved billigt.

En metode, der gør softwareopdatering nem, betyder også at nye funktionaliteter billigt kan tilføjes.

En opdatering som slår fejl er katastrofal, da vi i værste fald risikerer, at skulle sende en tekniker ud til alle de autonome netværk for at geninstallere Bifrost. Dette kan komme til at strække sig over et langt tidsrum, da administrationen som udgangspunkt er meget lavt bemanded.

3.6.3 Konfigurationsfiler

Vedligeholdelse af konfigurationsfiler på tværs af mange maskiner er et tilbagevendende problem for systemadministratorer. Ved manuel vedligeholdelse, oplever man to primære problemer. (1) Inkonsistens mellem konfigurationsfiler på de forskellige maskiner, og (2) tidsforbruget ved at rette filerne til i hånden. Den manuelle løsning skalerer derfor overhovedet ikke.

Manuel vedligeholdelse strider imod vores design principper omkring ingen lokale rettelser på Bifrost maskinen. En vigtig forudsætning for at driftsafdelingen kan vedligeholde et stort antal maskiner, er ikke at skulle logge ind på hver eneste maskine. Den enkelte Bifrost maskine *skal* kunne genetableres ud fra den centrale information.

⁴<http://bifrost.slu.se/index.en.html>

Det eneste værktøj vi kender der kan vedligeholde konfigurations filer på en større skala er programmet Cfengine ⁵.

Som garant for Cfengines formåen, kan nævnes at i DIKU's edbafdeling vedligeholdes der et stort antal forskellige UNIX systemer. Alle maskiners konfigurationsfiler vedligeholdes via Cfengine.

Kort fortalt, er Cfengine et høj niveau sprog der beskriver rettelser og ændringer til en given konfigurationsfil. Rettelsen sker på baggrund af hvilken "klasse" maskinen tilhører.

Citat fra Cfengines hjemmeside:

Cfengine, or the configuration engine is an agent/software robot and a high level policy language for building expert systems to administrate and configure large computer networks. Cfengine uses the idea of classes and a primitive intelligence to define and automate the configuration and maintenance of system state, for small to huge configurations.

Cfengine "sproget" benytter primitiver såsom:

```
AppendIfNoSuchLine "Teksten indsættes hvis den ikke eksistere i filen"
```

Vi vil ikke her komme nærmere ind på, hvordan cfengine scripts virker eller opbygges.

Cfengine scriptene skal udføres på den pågældende maskine. Dette udgør ikke noget problem på DIKU's integrerede UNIX system, hvor filerne blot exporteres via NFS og udføres via Cron. For at løse dette problem, i vores system, skal vi vælge mellem de distributions værktøjer, vi har behandlet i afsnit 3.3.2 på side 16. Dette valg vil vi dog først foretage i et eventuelt implementationsafsnittet.

⁵Cfengine er et officielt GNU program, des hjemmeside er: <http://www.iu.hioslo.no/cfengine/>

3.7 Åbne/lukke system

Åbne/lukke systemet er et abstraktions niveau eller API (Applikation Programmers Interface) oven på et eller andet faktisk filtreringssystem. Basalt set skal systemet kunne åbne og lukke for routning af IP-trafik på baggrund af kilden.

Åbne/lukke systemets API benyttes af kildeidentifikations modulet (se afsnit 3.4.11). Principielt bør det kun være kildeidentifikations modulet, der opdaterer/ændrer de dynamiske tilstande (kildetabellen), ellers er kildeidentifikationsmodulet ikke synkroniseret med den faktiske filtertilstand.

3.7.1 Pakkefiltrering

Systemet kunne godt implementeres v.h.a. routetabellen på gatewayen, men vi har af fleksibilitets hensyn valgt kun at kigge på pakkefiltrering (betegnes alment som packet-firewalling eller screening-router).

Vores krav til pakkefiltrerings funktionen (firewallen), i prioriteret rækkefølge:

- Gratis, og helst inkorporeret i kernen af operativsystemet.
- Skal naturligvis kunne filtrere trafik.
- Mulighed for gruppering af regler: Regelsæt.
- Nem og flexibel måde at genbruge regelsæt.
- Mulighed for at ændre, hvilket regelsæt en given IP skal benytte.
- En struktureret måde at organisere reglerne på.
- Mulighed for at springe irrelevante regelsæt over, dvs. at det ikke for alle pakker er nødvendig at gennemløbe alle filterregler.
- Redirigering af trafik (f.eks. tvungen proxy eller informationsside).
- Mulighed for NAT (Network Address Translation).

Vi har kigget på følgende gratis firewall implementationer:

- ipfwadm - fra Linux kerne 2.0.x
- ipchains - fra Linux kerne 2.2.x
- netfilter/iptables - fra Linux kerne 2.4.x
- ipfilter - fra⁶ distributionerne NetBSD, FreeBSD og OpenBSD⁷.

⁶Virker også på mange andre OS'er fx. SunOS v4.1.1-4.1.4, Solaris/Solaris-x86 v2.3-8, IRIX v6.2, HP-UX 11.00(beta)

⁷Er fjernet fra OpenBSD distributionen pga. licensproblemer

Det er kun *ipfwadm*, der ikke opfylder de ovenstående krav. Den understøtter ikke gruppering af regler, dvs. regelsæt, og derved heller ikke mulighed for at springe irrelevante regelsæt over.

Der er ingen af gruppens medlemmer, der har erfaring med *ipfilter*, og vi vil derfor umiddelbart fravælge at benytte dette værktøj. Alle gruppens medlemmer har praktiske erfaringer med både *ipchains* og *iptables*. En fordel ved *iptables* fremfor *ipchains* er, at *iptables* understøtter filtrering på MAC-adresse niveau. En anden fordel er, at *iptables* understøtter fuld NAT, og ikke kun NAPT som *ipchains*.

Vores umiddelbare valg falder derfor på *iptables* til pakkefiltrering / firewalling.

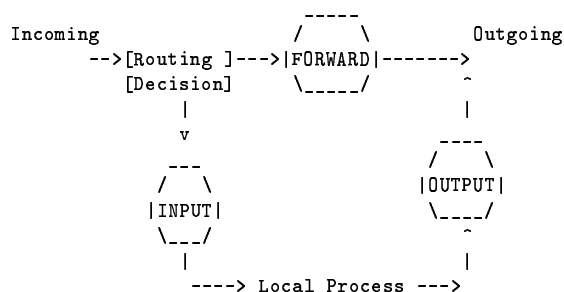
I implementationen vil vi dog stræbe efter, at opbygge systemet så modulært, at filtreringsfunktionen kan udskiftes.

Filtrerings mekanismen i netfilter/iptables

Måden, vi vil organisere filterreglerne på, er ved at gruppere dem i nogle regelsæt. I netfilter/iptables grupperes reglerne i *regelkæder*, der hedder “chains” i deres terminologi.

Der er tre indbyggede *regelkæder*: INPUT, OUTPUT og FORWARD. Udover disse er det muligt at lave bruger-defineret regelkæder. Netfilter/iptables er meget modul opbygget og det er muligt at indlæse kerne moduler, der giver extra faciliteter så som NAT kæder.

De indbyggede grund regelkæder, kan illustreres på følgende måde (Kilde: Netfilter dokumentationen [17]):



The three circles represent the three chains mentioned above. When a packet reaches a circle in the diagram, that chain is examined to decide the fate of the packet. If the chain says to DROP the packet, it is killed there, but if the chain says to ACCEPT the packet, it continues traversing the diagram.

Figur 3.7: Indbygget regelkæder. Kilde [17]

For “ipchains” kendere skal det pointeres, at pakker kun sendes igennem INPUT og OUTPUT, hvis det er til en lokal netværks adresse. Kun ikke lokale netværks adresser sendes igennem FORWARD. Med “ipchains” blev alle pakker sendt igennem både INPUT, FORWARD og OUTPUT.

Princippet med *regelkæder*, er at pakker kan sendes igennem en “kæde” af regler.

Reglerne er et filter, der matcher på felterne i IP-headeren (se figur 3.3 på side 29). Hvis pakken ikke matcher reglen, så sendes pakken blot videre til næste regel i kæden. Hvis pakken matcher reglen, skal reglen afgøre, hvad pakkens *skæbne* skal være. Udover ACCEPT og DROP, er det interessante at en pakkes *skæbne* kan være at blive *sendt videre til en anden regelkæde*.

Denne facilitet er yderst anvendelig, og opfylder/løser mange af vores krav til firewall funktionaliteten. Specielt nem og flexibel måde at genbruge regelsæt på, samt mulighed for at ændre, hvilket regelsæt en given IP skal benytte.

Som tidligere nævnt er netfilter/iptables meget modul opbygget, og der findes derfor mange former for pakke *skæbner*, der i deres termonologi hedder "TARGET". De indbyggede TARGET's er ACCEPT, DROP, QUEUE, RETURN eller "user-defined chain". Der findes også udvidelser til pakke *skæbner* eller TARGET. De udvidelser, der følger med netfilter/iptables i Linux kerne distributionen 2.4.x, er (uden at komme ind på deres funktionalitet) LOG, MARK, REJECT, TOS, MIRROR, SNAT, DNAT, MASQUERADE, REDIRECT (for beskrivelse se man iptables(8)). (Brug af disse udvidelse, krævet at Linux kernen kan loadet det passende kerne modul)

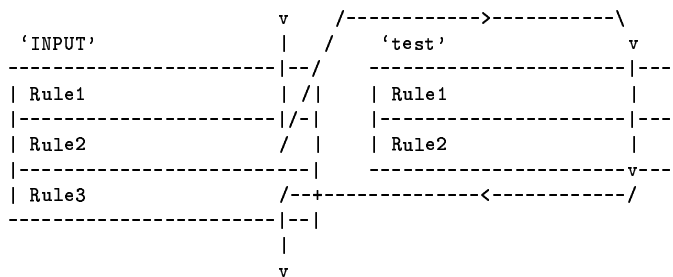
Regelkæder illustreres på følgende måde i netfilters dokumentation [17]:

```

'INPUT'                                     'test'
-----                                     -----
| Rule1: -p ICMP -j DROP |                 | Rule1: -s 192.168.1.1 |
|-----|                                     |-----|
| Rule2: -p TCP -j test  |                 | Rule2: -d 192.168.1.1 |
|-----|                                     |-----|
| Rule3: -p UDP -j DROP  |                 |
-----                                     -----
    
```

Consider a TCP packet coming from 192.168.1.1, going to 1.2.3.4. It enters the INPUT chain, and gets tested against Rule1 - no match. Rule2 matches, and its target is test, so the next rule examined is the start of test. Rule1 in test matches, but doesn't specify a target, so the next rule is examined, Rule2. This doesn't match, so we have reached the end of the chain. We return to the INPUT chain, where we had just examined Rule2, so we now examine Rule3, which doesn't match either.

So the packet path is:



Figur 3.8: Bruger defineret regelkæder. Kilde [17]

3.7.2 Grund elementer

Vores åbne/lukke system består af 2 grund elementer.

1. Et API til dynamisk at ændre filterings reglerne på kildebasis. Kan generaliseres som *tilstands-opdateringer/ændringer*.
2. De forskellige regelsæt kilderne kan vælge imellem. Kaldes i det efterfølgende for *grundreglerne*.

Der er en grundlæggende forskel på hvorledes opdateringen, overførelsen og vedligeholdelsen af disse to typer firewall elementer skal foregå på. Grundreglerne vedligeholdes centralt, og tilstands-opdateringer sker lokalt på Bifrost maskinen.

3.7.3 Tilstands-opdateringer

Tilstands-opdateringerne er dynamiske og benytter et lokalt API, og opdateringer sker ved interaktion med kildeidentifikations modulet. API'et udvikler vi selv, og den autoritative udgave vil derfor ligge på den centrale server, hvorefter opdateringen antages at ske som et led i vedligeholdelse og opdatering af software på Bifrost maskinen.

3.7.4 Grundreglerne

Grundreglerne vedligeholdes centralt. Distributions mekanismen er forholdsvis banal, og vi skal i implementationen overveje værktøjerne, der er blevet beskrevet i afsnit 3.3.2.

Grundreglerne, de faktiske iptables regler, skal oprettet på Bifrost maskinerne via et eller andet script. Man kan godt vedligeholde dette firewall-script manuelt i hånden, men det ville være praktisk at have en semi-automatisk generering. En form for konfigurationsfiler, som beskriver filterreglerne ville være praktisk.

Opdatering af grundreglerne kan give en del komplikationer, når dette sker på et kørende system. Jf. beskrivelsen i næste afsnit.

3.7.5 Design af regelkæderne

Vi vil gøre udvidet brug af netfilter/iptables regelkæder og muligheden for at "hoppe" mellem regelkæderne.

Vi vil strukturere de forskellige "services" eller "tilstande" (betegnes herefter som *service*), ved brug af netfilter/iptables regelkæder. Dvs. at der skal designes/laves en regelkæde per *service*.

Når en IP-adresse skal ændre status, og benytte en anden *service*, skal vi blot ændre reglen for den givne IP-adresse til at benytte en anden regelkæde.

Set fra åbne/lukke modulets side er IP-adressen vores primærnøgle. Det er op til kildeidentifikations modulet, at koble IP-adressen til en given kilde/bruger. MAC-adressen kan ikke benyttes som primærnøgle, pga. der kan komme alt for mange tabelindgange. MAC-adressen kan dog stadig bruges i en senere filter regel.

En måde, at implementere IP-adressen som primærnøgle, er at oprette en regel i FORWARD kæden per IP-adresse. Der skal heraf være en regel for hver IP-adresse på det autonomenetværk/lokalnetværket.

Vi ønsker at FORWARD kæden skal referere til en kæde per IP-adresse. Fremfor direkte at bestemme i FORWARD kæden, hvilket regelsæt en given IP-adresse skal benytte. Fordelen ved hver IP-adresse har sin egen kæde er, at det er nemmere at vedligeholde.

Ændringen af regelsæt for en given IP-adresse, kan gøres på følgende måde uden, at der opstår *race-conditions* i forbindelse med skiftet.

Istedetfor at slette indholdet af tabellen, indsættes der en ny "hop" regel foran den oprindelige, hvorefter nr.2 regel slettes. Herved opnår vi, at skiftet til en ny tilstand vil forløbe uden et tredje mellemstadie.

Denne metode løser desuden en faktisk/praktisk "race-condition" i netfilter-implementationen. Man kan ikke slette en filter regel, hvis der er en pakke der "passerer" reglen i samme øjeblik. Grunden er, at Linux kernen på det tidspunkt har allokeret en "skb"-netværks buffer, der er tilknyttet reglen (se [7, 6]). Netfilter håndterer dette ved at have nogle interne "tællere" på om reglen er i brug (se [18]).

Hvis vi skal opdatere grundreglerne på et kørende system, kan vi komme ud for samme problematik som netop beskrevet. Man kan godt lave den store forkromede og kompliceret løsning, der kan klare problemet runtime. Men vi vil formentlig vælge den mere pragmatiske løsning, at fjerne alt og opbygge regelsættene fra bunden, hvorefter kildeidentifikations modulet kan genindføre IP/bruger mappingen. Dette grunder i antagelsen om, at rettelser og opdateringer af grundreglerne vil ske tilpas sjældent.

3.7.6 Oplysningssystem

Når brugerne på en eller anden måde ikke har adgang igennem systemet, ville det være praktisk at kunne informere dem herom.

Da vi kun vil informere dem om, at de ikke har adgang igennem Bifrost, er det naturligt først at informere dem når de prøver at tilgå noget uden for det autonome netværk. Da denne trafik går igennem Bifrost har vi rig mulighed for at manipulere med deres forespørgelser.

Vores firewall har mulighed for at dirigere trafikken, så vi f.eks kan placere dem på en passende informations hjemmeside, uanset hvad de forsøger at vælge i deres browser.

Der kan være forskellige grunde til, at brugeren ikke har adgang, såsom:

- Brugeren er ikke logget ind.
- Brugeren har nået sin max grænse for trafik.
- Kontoen er lukket pga. misbrug, etc.

3.7.7 Åbne/lukke API

Alle *tilstands-opdateringer* kan principielt benyttes et fælles funktionskald.

```
setstate(IP, MAC, service, tilstand) -> errorcode
```

Argumenterne IP og **service** er påkrævet.

Argumentet **MAC** angiver MAC-adressen og er ikke påkrævet, men vil kunne give en extra form for sikkerhed.

Notationen “-> **errorcode**” betyder at funktionen returnerer en “**errorcode**”, dvs. en fejlkode, der beskriver resultatet af kommandoen.

Argumenterne **service** og **tilstand** beskriver basalt set, hvilket regelsæt IP-adressen tilknyttes. Principielt kunne vi nøjes med et argument til at beskrive det ønskede regelsæt, men det er mere overskueligt og fleksibelt at dele det op i **service** og **tilstand**.

Service	Tilstand
OPEN	Forskningsnet, ISP1, ISP2
CHANGE	Forskningsnet, ISP1, ISP2
ADD	Forskningsnet, ISP1, ISP2
INSERT	Forskningsnet, ISP1, ISP2
INFORM	ikke-logget-ind, trafik-max, konto-lukket
CLOSE	-
USERDEF	firewall1, firewall2

Tabel 3.1: Service og Tilstand

Beskrivelse af tabel 3.1:

OPEN, **CHANGE**, **ADD** og **INSERT** minder meget om hinanden og er egentlig blot afarter af samme funktion, nemlig at ændre tilstanden for en IP-adresse.

OPEN , bruges når det er første gang IP-adressen registreres. Man ved derfor, at man kan slette IP-adressens regel tabel.

CHANGE , ved man er et tilstands skift, fra en tidligere tilstand til en ny tilstand.

ADD , der tilføjes en tilstand efter i en eksisterende tilstand.

INSERT , der indsættes en tilstand foran en eksisterende tilstand.

INFORM , et oplysningssystem (se afsnit ovenfor), der bruges til at informere brugeren om, at adgangen igennem Bifrost er lukket.

CLOSE , skal selvfølgelig lukke forbindelsen. Og opfattes som at IP-adresse bindingen ophører. Rediregering til en passende login side, vil formentlig være på sin plads.

USERDEF , vi har tidligere snakket om bruger-defineret sikkerhedsniveau på firewal-len.

Selvom funktionskaldet kan udtrykkes som et kald, vil det formenligt være praktisk at lave nogle wrapper funktioner i API'et. Såsom:

```
open(IP, MAC, tilstand) -> errorcode
close(IP)                -> errorcode
```

Grunden til, at vi ikke laver de forskellige “services” som specifikke funktionskald, er at systemet er nemmere at udvides med nye “services” og “tilstande”, når vi benytter `setstate` funktionen.

Forespørgelser omkring tilstanden. Kildeidentifikations modulet bør have det korrekte billede af tilstanden i åbne/lukke systemet. Men til verifikation, debugging og konsistens check, er det praktisk at kunne aflæse tilstanden for en given kilde.

```
getstate(IP, MAC, service) -> (tilstand, errorcode)
getstate(IP, MAC)          -> (service, tilstand, errorcode)
getstate(IP)               -> (service, tilstand, errorcode)
dumpstate() -> alt
```

For at et eksternt modul kan afgøre om kilden og dens tilknyttet IP-adresse er aktiv, kan man blot observere en trafik tæller. Derfor er det praktisk, at tilbyde en sådan funktion.

```
getcounter(IP) -> (trafikmængde, errorcode)
```

Kapitel 4

Implementation

I dette kapittel beskrives implementationen og afprøvning af enkelte løsningsforslag.

Formålet med implementationen er i første omgang, at demonstrere at de metoder, der er beskrevet i rapporten fungerer i praksis.

Derudover kan implementation hjælpe til at finde mangler og forglemmelser i vores analyse og design.

4.1 Kildeidentifikation

Vi har valgt at implementere kildeidentifikation, da den er en forudsætning for, at mange af de elementer vi beskriver i kapitel 2 kan fungerer.

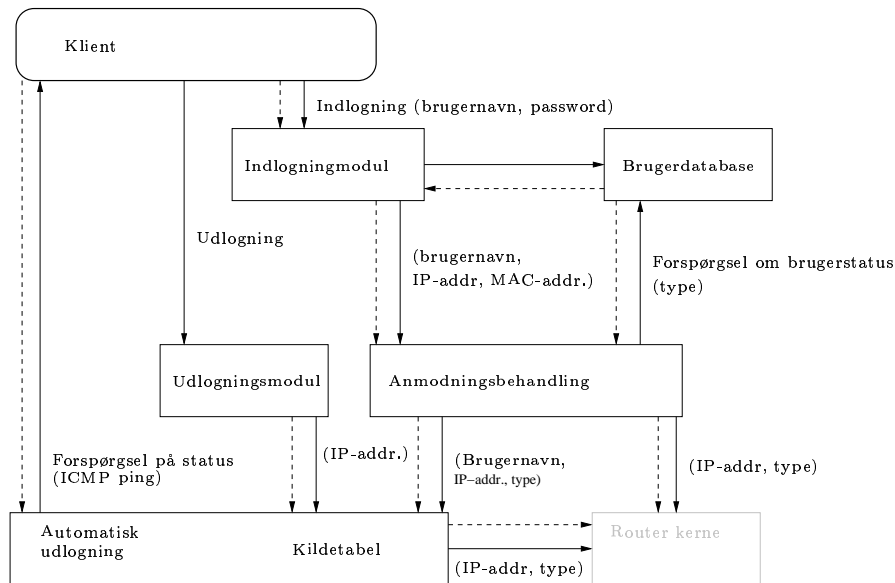
En stor del af implementationen er gået forud for udarbejdelsen af denne rapport — herunder designafsnittet (afsnit 3.4.11). Designafsnittet er altså skrevet på baggrund af de erfaringer, vi har gjort os under implementeringen og designet afviger derfor for den faktiske implementation.

Skal systemet tages i brug i større målestok vil det være naturligt at implementere det i en form, der er konsistent med designet, da den eksisterende implementation ikke er åben for udvidelser.

Programmerne er skrevet i Perl og kører under Linux.

4.1.1 Implementationen i forhold til design

En principdiagram for vores implementation ses i figur 4.1. Denne kan sammenholdes med den tilsvarende figur 3.2 i afsnit 3.4.11. De væsentligste forskelle er, at kildetabelen og det automatiske udlukningsmodul er smeltet sammen. Derudover er router abstraktionslaget udgået således, at kommunikation foregår direkte med routerkernen. På grund af u hensigtsmæssigheder i implementationen af kildetabelmodulet, gik der urimelig lang tid fra en bruger havde logget ind til denne var aktiveret i routerkernen. Derfor sker kommunikationen med routerkernen ved indlogging uden om kildetabelmodulet.



Figur 4.1: Prinsipkitse af vores implementation. Sammenhold eventuelt med vores design i figur 3.2.

4.1.2 Programbeskrivelse

Som ind og udlogningsinterface har vi afprøvet ssh og telnet. Begge dele var i forvejen implementeret i den Linux-distribution vi anvender.

Udlogningsmodulet

Udlogningsmodulet (bilag C) er hovedsageligt et wrapperlag, der trækker indformation ud fra telnet/ssh og sender disse videre til kildetabelmodulet. Udtræk af brugernavn, MAC- og IP-adresse sker i vores implementation på en lidt særpræget måde, da ssh og telnet ikke giver disse eksplicit, men ligger dem som en tilstand i operativsystemet før lukningmodulet kaldes. Udtrækket sker i funktionen `findid` i bilag C.

Kommunikation med kildetabelmodulet sker ved, at udlogningmodulet skriver til en fil som læses af kildetabelmodulet. Fordelen ved denne kommunikationsform er, at den er let at implementere og letter aflusning, da indholdet i filen til alle tider kan aflæses på et kørende system. Ulempen er, at overførselsen ikke sker øjeblikkeligt.

Anmodningsbehandlingsmodulet

Anmodningsbehandlingsmodulet (bilag B) skal lige som udlogningmodulet trække indformation om brugernavn, MAC- og IP-adresse ud fra telnet/ssh. Dette sker ligeledes i funktionen `findid`. I funktionen `aavenforbindelse` sker den egenlige anmodningsbehandling. Vores brugerdatabase er implementeret som simple filer, hvilket fungerer udmærket, så længe den ikke skal distribueres via netværk. Funktionen returnerer det, der på figuren kaldes status i, form af to strenge. Disse er begge en kommando sekvens

til routerkernen, der henholdsvis åbner og lukker for routning. Lige som udlogningsmodul kommunikterer dette modul med kildetabelmodul via en fil.

Kildetabelmodul

Kildetabelmodulets funktionalitet ligger i funktionerne `laesnye` og `luk` som ses i bilag A. Det tager imod oplysning om ind- og udlogninger fra udlognings- og anmodningsbehandlingsmodul og vedligeholder en tabel over kilder i en fil. Funktionen `luk`'s eneste funktion er, at kalde routerkernen med strengen, der blev dannet i anmodningsbehandlingsmodul.

Automatisk lukning

Kernen i det automatiske lukningmodul ligger i den sidste `while`-løkke i bilag A: alle IP-adresser i kildetabellen gennemløbes og pinges. Svarer en IP-adresse ikke på pinget logges den tilsvarende bruger ud.

4.1.3 Afprøvning

Programmet har været under udvikling på Grønjordskollegiet, hvor det har været afprøvet og der løbende er blevet rettet fejl. Programmet, der er vist i bilag A–C, er sat i drift på Grønjordskollegiet, og har kørt i en længere periode, uden at der er blevet fundet fejl i dets funktionalitet.

Kapitel 5

Handlingsplan

I dette kapitel vil vi give en kort tekniske handlingsplan for, hvordan Foreningen Kollegiet København kan komme videre med etablering og drift af et centralt styret kollegienetværk i København.

For at etablere et minimalt regionalt kollegienetværk i København er følgende tekniske komponenter essentielle:

- System til central installation, opdatering samt monitorering af firewall/router på de enkelte kollegier. (Bifrostmaskinen)
- System til at lave kildeidentifikation af trafik der passerer firewall/routeren.
- System til valg blandt multiple internetgateways.
- Database over brugere.

Denne rapport giver en analyse af problemstillinger ved disse tekniske komponenter. Rapporten laver ikke en analyse af hvordan komponenterne kan fungere sammen, en sådan analyse er essentiel at lave før etableringen af kollegienetværket. Foreningen Kollegiet København bør derfor starte med at lave en sådan analyse.

Foreningen Kollegiet København bør herefter påbegynde udarbejdelsen af grundige designbeskrivelser af de enkelte komponenter. Denne rapport indeholder knap så detaljerede designbeskrivelser, som kan bruges som udgangspunkt for udarbejdelsen.

Til slut bør foreningen påbegynde en implementation af de enkelte komponenter. Denne rapport indholder en implementation af et system til kildeidentifikation, som kan benyttes til inspiration.

Foreningen Kollegiet København bør herefter kunne påbegynde etableringen, af det regionale kollegienetværk i København.

Foreningen bør dog hurtigst muligt efter herefter (og gerne før) tilføje følgende tekniske komponenter til netværket:

- System til Autokonfiguration

- System til Brugerstatus/konfigurations system
- System til trafikaccounting.
- System til fairqueuing.

Disse komponenter er essentielle for langvarig drift, af et kollegienetværk og specielt essentielle set i forhold til brugernes oplevelse af kollegienetværket.

Kapitel 6

Konklusion

Vi har klarlagt de involverede parters ønsker og behov ved etableringen af et regionalt kollegienetværk i København.

Med baggrund i disse behov har vi identificeret en række tekniske problemstillinger i forbindelse med opbygningen af et regionalt kollegienetværk.

Vi har udvalgt og analyseret centrale problemstillinger og givet løsningsforslag til disse.

Udvalgte af løsningsforslagene munder ud i et designforslag. Netop et af disse designforslag har vi implementeret og afprøvet i et eksisterende netværksmiljø.

Vores arbejde har resulteret i en handlingsplan der beskriver hvordan foreningen Kollegienet København kan forsætte arbejdet på den tekniske del af etableringen af et regionalt kollegienetværk i København.

Generelt har vores arbejde været tilfredsstillende. Vi ville dog gerne have givet en kortfattet analyse, af de resterende problemstillinger for at gøre behandlingen mere fuldstændig.

Som udgangspunkt lagde vi stor vægt på at løse problemer fra den virkelige verden, vi burde derfor have lagt større vægt på implementationen. Dette viste sig desværre i praksis at være for omfattende set i forhold til projektets omfang. Rapportens primære vægt ligger i identifikation og analyse af problemstillingerne.

Vores arbejde har vist, at det rent teknisk er muligt, at etablere et regionalt kollegienetværk i København, som opfylder alle parters væsentlige behov. Rapporten giver dog ikke en komplet opskrift på etableringen af et regionalt kollegienetværk i København.

Litteratur

- [1] F. Baker. RFC1812 requirements for IP version 4 routers. Technical report, RFC Editor, Juni 1995.
- [2] J. Case, M. Fedor, M. Schoffstall, and J. Davin. RFC1157 a simple network management protocol (snmp). Technical report, RFC Editor, May 1990.
- [3] E. Chen and T. Bates. RFC1998 an application of the BGP community attribute in multi-home routing. Technical report, RFC Editor, August 1996.
- [4] M. Holdrege and P. Srisuresh. RFC3027 protocol complications with the IP network address translator. Technical report, RFC Editor, January 2001.
- [5] S. Ken and R. Atkinson. RFC2401 security architecture for the internet protocol. Technical report, RFC Editor, November 1998.
- [6] Harald Welte laforge@gnumonks.org. The journey of a packet through the linux 2.4 network stack. Technical report, Internet, Oktober 2000. <http://www.gnumonks.org/ftp/pub/doc/packet-journey-2.4.html>.
- [7] Harald Welte laforge@gnumonks.org. skb - linux network buffers. Technical report, Internet, Oktober 2000. <http://www.gnumonks.org/ftp/pub/doc/skb-doc.html>.
- [8] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. RFC1928 SOCKS protocol version 5. Technical report, RFC Editor, March 1996.
- [9] Ron McCarty. R.I.P RIP? *SysAdmin*, 10(3):61–62, March 2001.
- [10] J. Mogul and J. Postel. RFC950 internet standard subnetting procedure. Technical report, RFC Editor, August 1985.
- [11] J. Postel. RFC791 - internet protocol. Technical report, RFC Editor, September 1981.
- [12] J. Postel. RFC792 - internet control message protocol. Technical report, RFC Editor, September 1981.
- [13] Y. Rekhter. RFC1817 cidr and classful routing. Technical report, RFC Editor, August 1995.
- [14] Y. Rekhter and P. Gross. RFC1772 application of the border gateway protocol in the internet. Technical report, RFC Editor, March 1995.

- [15] Y. Rekhter and T. Li. RFC1771 a border gateway protocol 4 (BGP-4). Technical report, RFC Editor, March 1995.
- [16] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. RFC1918 address allocation for private internets. Technical report, RFC Editor, February 1996.
- [17] Rusty Russell. Linux 2.4 packet filtering howto. Technical report, Internet, 2000. <http://netfilter.samba.org/unreliable-guides/packet-filtering-HOWTO/index.html>.
- [18] Rusty Russell. Linux netfilter hacking howto. Technical report, Internet, 2000. <http://netfilter.samba.org/unreliable-guides/netfilter-hacking-HOWTO/index.html>.
- [19] R. Hinden S. Deering. RFC2460 - internet protocol, version 6 (IPv6) specification. Technical report, RFC Editor, December 1998.
- [20] P. Srisuresh and K. Egevang. RFC3022 traditional IP network address translator (traditional NAT). Technical report, RFC Editor, January 2001.
- [21] P. Srisuresh and M. Holdrege. RFC2663 ip network address translator (NAT) terminology and considerations. Technical report, RFC Editor, August 1999.
- [22] P. Srisuresh, G. Tsirtsis, P. Akkiraju, and A. Heffernan. RFC2694 DNS extensions to network address translators (DNS_ALG). Technical report, RFC Editor, September 1999.
- [23] W. Richard Stevens. *The Protocols*, volume 1 of *TCP/IP Illustrated*. Addison Wesley, 1994.
- [24] P. Traina. RFC1773 experience with the BGP-4 protocol. Technical report, RFC Editor, March 1995.
- [25] P. Traina. RFC1774 BGP-4 protocol analysis. Technical report, RFC Editor, March 1995.
- [26] Andrew Tridgell. *Efficient Algorithms for Sorting and Synchronization*. PhD thesis, The Australian National University, April 2000. http://samba.org/~tridge/phd_thesis.pdf.

Bilag A

Kode: portlukker2k

```
#!/usr/bin/perl
#
use strict;
my $logdir = "/var/portlukker/";
my $logfil = join("", $logdir, "aabninglog");
my $macfil = join("", $logdir, "macnumre");
my $aabnefil = join("", $logdir, "aabne");
my $kontaktfil = join("", $logdir, "kontaktfil");
my $luklogfil = join("", $logdir, "lukninglog");
my $ipfwadm = "ipfwadm";
my $arp = "arp";
my $ping = "ping";
my $forsoeg = 10;
my %gode;
my $ip;

$ENV{'PATH'} = '/bin:/usr/bin:/sbin:/usr/sbin';

#
# Forsoeager at udfoere shellkommandoer indtil succes
#
sub multiexec {
    my @kommandoer = split(';', @_);
    my $count;
    my $success;
    my $returnval = 0;
    foreach my $kommando (@kommandoer)
    {
        $success = 0;
        for ($count = 1 ; $count < 10 && $success == 0 ; $count++)
        {
            if(system('sh', '-c', $kommando) == 0)
            {
                $success = 1;
            }
            else
            {
                open DEBUGLOG, ">>/var/portlukker/multiexecdebug";
                printf DEBUGLOG ("kunne ikke udfoere: %s,%s,%s forsoeg %d\n",
                    scalar localtime, time, $kommando, $count);
            }
        }
        close DEBUGLOG;
        sleep(1);
    }
}
if($success == 0)
{
    $returnval = 1;
}
}
```

```

    return($returnval)
}

sub ping
{
    my($i, $ipnr, $cmd, $result);
    $ipnr = $_[0];
    $cmd = sprintf('%s -c 1 %s', $ping, $ipnr);
    # print "pinger ", $ipnr, "\n";
    for($i = 0, $result=0; ($i < $forsoeg) && !($result); $i++)
    {
        $result = `$cmd`;
        $result = ($result =~ /. *1 packets received.*/s);
    };
    # skriv i logfilen hvis en maskine kunne pinges paa andet forsoeg
    if ($i>1 && $i!=$forsoeg)
    {
        open debug, ">>/var/portlukker/debug";
        print debug (scalar localtime, " n", $i, " ", $ipnr, "\n");
        close debug;
    };
    # print ($result ? "OK!\n" : "Not OK!\n");
    # print $i;
    return($result);
}

sub maksimum
{
    my($stal1, $stal2);
    $stal1 = $_[0];
    $stal2 = $_[1];
    if($stal1 > $stal2)
    {return $stal1;}
    else
    {return $stal2;}
}

sub luk
{
    my($ipnr, $cmd, $mac, $hostname, $user);
    $ipnr = $_[0];
    # print "lukker: ", $ipnr, "\n";
    $gode = $gode{$ipnr};
    $cmd =~ s/.*?;(.*?)$1/;
    multiexec($cmd);
    # print "med commandoen '", $cmd, "'\n";

    $_=$gode{$ipnr};

    if(/ (\w\w:\w\w:\w\w:\w\w:\w\w:\w\w) (\w+) (.+?);.*/)
    {
        $mac=$1;
        $user=$2;
        $hostname=$3;
    }
    else
    {
        $mac="";
        $user="";
        $hostname="";
    }
    open luklog, ">>$luklogfil";
    printf luklog ("lukket: %s,%s,%-12s,%15s,%-12s,%-17s\n", scalar localtime,
    time, $user, $ipnr, $hostname, $mac);
    close luklog;
    delete $gode{$ipnr};

    while(! open aabne, ">$aabnefil")
    {
        open debug, ">>/var/portlukker/debug";
        print debug (scalar localtime, " kan ikke skrive aabne ved lukning ",

```

```

        $ipnr, "\n");
    close debug;
};

foreach $ipnr (keys %gode)
{
    print aabne $ipnr, $gode{$ipnr}, "\n";
};
close aabne;
};

sub laesnye
{
    my ($i, $ipnr);
    open nye, "<$kontaktfil";
    $i=0;
    while(<nye>)
    {
if (/^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})( .....*)/)
    {
        $gode{$1} = $2;
        $i=1;
    };
if (/^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) lukkes.* && exists $gode{$1})
    {
        luk($1);
    };
    };
    open nye, ">$kontaktfil";
    close nye;

    if($i)
    {
open aabne, ">$aabnefil";
foreach $ipnr (keys %gode)
{
    print aabne $ipnr, $gode{$ipnr}, "\n";
};
close aabne;
    }
};

#
# Main
#
open nye, "<$aabnefil";
while(<nye>)
{
    /^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})(.*/);
    $gode{$1} = $2;
};
close nye;

while(1)
{
    laesnye;
    foreach $ip (keys %gode)
    {
select(undef, undef, undef, 0.20); #sleep x secs
if (!(~z "$kontaktfil"))
{
    laesnye;
};
if (!(ping $ip))
{
    luk($ip);
};
    };
};
}

```

Bilag B

Kode: lukmigud

```
#!/usr/bin/perl

use strict;
my $logdir = "/var/portlukker/";
my $logfil = join("", $logdir, "aabninglog");
my $macfil = join("", $logdir, "macnumre");
my $macskiftfil = join("", $logdir, "macskiftlog");
my $aabnefil = join("", $logdir, "aabne");
my $kontaktil = join("", $logdir, "kontaktil");
my $who = "who -m";
my $nslookup = "nslookup";
my $arp = "arp";
$ENV{'PATH'} = '/usr/local/bin:/bin:/usr/bin:/usr/sbin:/sbin';

my($aabencmd, $lukcmd);
my($bruger, $ipnr, $hostnavn, $mac);
my(@macnrlist, @gamlemac);
my($maclinie);

#
# Forsoeger at udfoere shellkommandoen indtil succes
#
sub multiexec {
    my @kommandoer = split(';', @_);
    my $count;
    my $success;
    my $returnval = 0;
    foreach my $kommando (@kommandoer)
    {
        $success = 0;
        for ($count = 1 ; $count < 10 && $success == 0 ; $count++)
        {
            if(system('sh', '-c', $kommando) == 0)
            {
                $success = 1;
            }
            else
            {
                open DEBUGLOG, ">>/var/portlukker/multiexecdebug";
                printf DEBUGLOG ("kunne ikke udfoere: %s,%s,%s forsoeg %d\n",
                    scalar localtime, time, $kommando, $count);
            }
        }
        close DEBUGLOG;
        sleep(1);
    }
}

if($success == 0)
{
    $returnval = 1;
}
```

```

    }
    return($returnval)
}
#
# flg funktion returnerer brugernavn, ipnummer, hostnavn, mac
#
sub findid {
    my($result, $brugernavn, $adresse, $cmd, $result, $ip, $mac);

# anvend who til at finde bruger og hostnavn

    $result = `who`;
    $_ = $result;
    /.*!(\S*)\s.*\((.*)\)/;
    $brugernavn = $1;
    $adresse = $2;
    $_ = $adresse;

# Udfoer et nslookup hvis den returnerede host ikke er en ip-adresse.

    if (!/\d+\.\d+\.\d+\.\d+/)
    {
# Nedenstaaende linje er kun noedvendig paa steder hvor who husker 16
# tegns hostnavne
#   $adresse =~ s/(.+?)\..*/$1/;
#   $cmd = sprintf('%s %s', $nslookup, $adresse);
#   $result = `$cmd`;

        if($result =~ s/.*Address.*Address:\s*(\d+\.\d+\.\d+\.\d+).*/$1/s)
        {
            $ip = $result;
        }
        else
        {
            die "Fejl i DNS";
        }
    }
    else
    {
        $ip = $adresse;
    }
};

# Find mac-adressen ud fra ip-nummeret

$cmd = sprintf("%s -na %s", $arp, $ip);
$result = `$cmd`;
if ($result =~ /.*at\s*(.....)\s\[ether\].*/s)
{
    $mac = $1;
}
else
{
    die "Kunne ikke finde macadresse";
};
return($brugernavn, $ip, $adresse, $mac);
};

#
# Skriver logon i $logfil
#

sub skrivlog
{
    open udlog, ">>$logfil";
    printf udlog ("aabnet: %s,%s,%-12s,%15s,%-12s,%-17s\n", scalar localtime,
        time, $bruger, $ipnr, $hostnavn, $mac);
    close udlog;
};

#
# Vedligeholder liste over macnumre som brugeren logger paa fra

```



```

## til firewallen
# Den returnerer den sreng der gives til shell ved aabning
# samt den steng der gives til shell ved lukning
#

sub aabenforbindelse
{
    my $subcmd, $aabencmd, $lukcmd;
    my $ipfwadm = "/sbin/ipfwadm";
    my($ipnr, $mac, $bruger);
    $ipnr = @_ [0];
    $mac = @_ [1];
    $bruger = @_ [2];
    my $kontofil="/var/portlukker/fnetbrugere";
    my $lukkedefil="/var/portlukker/lukkede";
    my $tbruger;
    my $fnet = 0;
    my $dynip = 0;
    if ($ipnr eq "195.249.184.10" ||
$ipnr eq "195.249.184.11")
    {
        die "Du kan ikke bruge lukmig ud fra denne host!";
    };

    print "\n\n";

    open LUKKEDE, "<$lukkedefil" || print "Kunne ikke aabne lukkedefil\n";

    while((defined($tbruger=<LUKKEDE>)) &&
chomp($tbruger) &&
!($tbruger eq $bruger))
    {
        };
    if ($tbruger eq $bruger)
    {
        print "Din konto er lukket, sandsynligvis p.g.a. overforbrug.\n\n";
        sleep(4);
        die "Din konto er lukket, sandsynligvis p.g.a. overforbrug.\n\n";
        };
    close LUKKEDE;
    open FNETBRUGER, "<$kontofil" || print "Kunne ikke aabne kontofil\n";
    while((defined($tbruger=<FNETBRUGER>)) &&
chomp($tbruger) &&
!($tbruger eq $bruger))
    {
        };
    if ($ipnr =~ /\d+\.\d+\.\d+\.\d+/ ||
$ipnr =~ /\d+\.\d+\.\d+\.\d+/ ||
$ipnr =~ /\d+\.\d+\.\d+\.\d+/)
    {
        $dynip = 1;
        };

    close FNETBRUGER;
    if ($tbruger eq $bruger)
    {
        if ($dynip == 1)
        {
            print "Da du er registreret som studerende under Forskningsministeriet.\n
Og du bruger dynamisk ipadresser bliver du sendt paa Forskernettet.\n";
            $fnet = 1;
        }
        else
        {
            print "Du er registreret som studerende under Forskningsministeriet.\n
Men da din ipadresse er statisk kan du ikke komme paa Forskernettet. \n
Skift til dynamisk ipadresse hvis du ikke har nogen grund til at have statisk.\n";
        };
    }
    else
    {
        };
    }
}

```

```

print "Du er ikke registreret som studerende under Forskningsministeriet\n
og kommer derfor paa den komerentielle forbindelse.\n";
}

if ($fnet == 1)
{
#aabnekommandoer
$subcmd = sprintf('/usr/local/sbin/eiptables -A portlukker
-j postportlukker -s 0/0 -d %s/32', $ipnr);
$aabencmd = $subcmd;
$subcmd = sprintf('/usr/local/sbin/eiptables -A portlukker
-j postportlukker -s %s/32 -d 0/0', $ipnr);
$aabencmd = join(" ; ", $aabencmd, $subcmd);
# $subcmd = sprintf('/usr/local/sbin/eiptables -t nat -A POSTROUTING
# -j SNAT --to-source 130.226.217.161 -s %s/32 -d 0/0', $ipnr);
# $aabencmd = join(" ; ", $aabencmd, $subcmd);
$subcmd = sprintf('ip rule add from %s table uni prio 100', $ipnr);
$aabencmd = join(" ; ", $aabencmd, $subcmd);
$subcmd = sprintf('ip route flush cache');
$aabencmd = join(" ; ", $aabencmd, $subcmd);

#lukkekommandoer
$subcmd = sprintf('/usr/local/sbin/eiptables -D portlukker
-j postportlukker -s 0/0 -d %s/32', $ipnr);
$lukcmd = $subcmd;
$subcmd = sprintf('/usr/local/sbin/eiptables -D portlukker
-j postportlukker -s %s/32 -d 0/0', $ipnr);
$lukcmd = join(" ; ", $lukcmd, $subcmd);
# $subcmd = sprintf('/usr/local/sbin/eiptables -t nat -D POSTROUTING
# -j SNAT --to-source 130.226.217.161 -s %s/32 -d 0/0', $ipnr);
# $lukcmd = join(" ; ", $lukcmd, $subcmd);
$subcmd = sprintf('ip rule del from %s table uni prio 100', $ipnr);
$lukcmd = join(" ; ", $lukcmd, $subcmd);
$subcmd = sprintf('ip route flush cache');
$lukcmd = join(" ; ", $lukcmd, $subcmd);

}
else
{
if ($dynip == 0)
{
#aabnekommandoer
$subcmd = sprintf('/usr/local/sbin/eiptables -A portlukker
-j postportlukker -s 0/0 -d %s/32', $ipnr);
$aabencmd = $subcmd;
$subcmd = sprintf('/usr/local/sbin/eiptables -A portlukker
-j postportlukker -s %s/32 -d 0/0', $ipnr);
$aabencmd = join(" ; ", $aabencmd, $subcmd);

#lukkekommandoer
$subcmd = sprintf('/usr/local/sbin/eiptables -D portlukker
-j postportlukker -s 0/0 -d %s/32', $ipnr);
$lukcmd = $subcmd;
$subcmd = sprintf('/usr/local/sbin/eiptables -D portlukker
-j postportlukker -s %s/32 -d 0/0', $ipnr);
$lukcmd = join(" ; ", $lukcmd, $subcmd);
}
else
{
#aabnekommandoer
$subcmd = sprintf('/usr/local/sbin/eiptables -A portlukker
-j postportlukker -s 0/0 -d %s/32', $ipnr);
$aabencmd = $subcmd;
$subcmd = sprintf('/usr/local/sbin/eiptables -A portlukker
-j postportlukker -s %s/32 -d 0/0', $ipnr);
$aabencmd = join(" ; ", $aabencmd, $subcmd);
$subcmd = sprintf('/usr/local/sbin/eiptables -t nat -A POSTROUTING
-j SNAT --to-source 195.249.184.2 -s %s/32 -d 0/0 -o eth2', $ipnr);
$aabencmd = join(" ; ", $aabencmd, $subcmd);
}
}
}
}

```

```

#lukkekommandoen
  $subcmd = sprintf('/usr/local/sbin/eiptables -D portlukker
    -j postportlukker -s 0/0 -d %s/32', $ipnr);
  $lukcmd = $subcmd;
  $subcmd = sprintf('/usr/local/sbin/eiptables -D portlukker
    -j postportlukker -s %s/32 -d 0/0', $ipnr);
  $lukcmd = join(" ; ", $lukcmd, $subcmd);
  $subcmd = sprintf('/usr/local/sbin/eiptables -t nat -D POSTROUTING
    -j SNAT --to-source 195.249.184.2 -s %s/32 -d 0/0 -o eth2', $ipnr);
  $lukcmd = join(" ; ", $lukcmd, $subcmd);
}
}

  return($aabencmd, $lukcmd);
};

### main

($bruger, $ipnr, $hostnavn, $mac) = findid; # disse variable bruges globalt

# Soerg for at disse maskiner ikke faar adgang til Internet
#
#
if (!controllermac ($bruger, $mac))
{
  open maclist, "<$macfil";
  @macnrlist = <maclist>;
  open macudlist, ">$macfil";
  foreach $maclineie (@macnrlist)
  {
if (!($maclineie =~ /^(.....)\s+$bruger$/))
{
  print macudlist $maclineie;
}
else
{
  push(@gamlemac, $1);
};
};
close macudlist;
open maclog, ">>$macskiftfil";
print maclog scalar localtime, " ", $bruger, " ", $hostnavn, "\n";
print maclog "< ", @gamlemac, "\n";
print maclog "\n> ", $mac, "\n\n";
close maclog;

#De foelgende linjer udskriver maclog til en linjeskriver tilsluttet /dev/lp1
#
#   if(open maclog, ">/dev/lp1")
#   {
# print maclog scalar localtime, " ", $bruger, " ", $hostnavn, "\n";
# print maclog "< ", @gamlemac, "\n";
# print maclog "\n> ", $mac, "\n\n";
# close maclog;
#   }
#   else
#   {
# 'mail -s "Bifrosts printer er nede" root@egmont-kol.dk';
#   };

  open macnumre, ">>$macfil";
  printf macnumre ("%s %s\n", $mac, $bruger);
  close macnumre;
};
skrivlog;

if (forbindelseaaben ($ipnr))
{
  print "forbindelsen er allerede Aaben\n";
}

```

```
    sleep(5);
    die "forbindelsen er allerede Aaben\n";
};

($aabencmd, $lukcmd) = aabenforbindelse($ipnr, $mac, $bruger);
#print "$aabencmd\n";
#print "$lukcmd\n";
if(multiexec($aabencmd) == 0)
{
    open kontakt, ">>$kontaktfil";
    print kontakt ($ipnr," ", $mac," ", $bruger, " ", $hostnavn, ";", $lukcmd, "\n");
    close kontakt;
    print "Du er nu tilsluttet Internet!\n";
}
else
{
    open DEBUGLOG, ">>/var/portlukker/lukmiguddebug";
    print DEBUGLOG "Kunne ikke udfoere:$aabencmd\n";
    close DEBUGLOG;
    print "Der opstod fejl udner tilslutning!\n";
}
sleep(5);
#print $aabencmd;
```

Bilag C

Kode: lukmig

```
#!/usr/bin/perl
#
use strict;
my $logdir = "/var/portlukker/";
my $logfil = join("", $logdir, "aabninglog");
my $kontaktil = join("", $logdir, "kontaktil");
my $aabnefil = join("", $logdir, "aabne");
my $who = "who -m";
my $nslookup = "nslookup";
my $arp = "arp";
$ENV{'PATH'} = '/usr/sbin:/usr/local/sbin:/usr/local/bin:/bin:/usr/bin';

my($bruger, $ipnr, $hostnavn, $mac);

#
# flg funktion returnerer brugernavn, ipnummer, hostnavn, mac
#
sub findid {
    my($result, $brugernavn, $adresse, $cmd, $result, $ip, $mac);

    # anvend who til at finde bruger og hostnavn

    $result = '$who';
    $_ = $result;
    /.*!(\S*)\s.*\((.*)\)/;
    $brugernavn = $1;
    $adresse = $2;
    $_ = $adresse;

    # Udfør et nslookup hvis den returnerede host ikke er en ip-adresse.

    if (!/\d+\.\d+\.\d+\.\d+/)
    {
        $adresse =~ s/(.+?)\././$1/;
        $cmd = sprintf('%s %s', $nslookup, $adresse);
        $result = '$cmd';

        if($result =~ s/.*Address.*Address:\s*(\d+\.\d+\.\d+\.\d+).*/$1/s)
        {
            $ip = $result;
        }
        else
        {
            die "Fejl i DNS";
        }
    }
    else
    {
        $ip = $adresse;
    }
}
```

```

};

# Find mac-adressen ud fra ip-nummeret

$cmd = sprintf("%s -na %s", $arp, $ip);
$result = `$cmd`;
if ($result="/.*at\s*(.....)\s\[ether\].*/s)
{
    $mac = $1;
}
else
{
    die "Kunne ikke finde macadresse";
};
return($brugernavn, $ip, $adresse, $mac);
};

sub forbindelseaaben
{
    my($ipnr, $ok);
    $ok = 0;
    $ipnr = @_ [0];
    open aabne, "<$aabnefil";
    while (<aabne>)
    {
        if (/ $ipnr .*/)
        {
            $ok = 1;
        };
    };
    close aabne;
    if (!$ok)
    {
        open aabne, "<$kontaktfil";
        while (<aabne>)
        {
            if (/ $ipnr .*/)
            {
                $ok = 1;
            };
        };
        close aabne;
    };
    return $ok;
};

#
# Skriver logon i $logfil
#

sub skrivlog
{
    open udlog, ">>$logfil";
    printf udlog ("aabnet: %s,%s,%-12s,%15s,%-12s,%-17s\n", scalar localtime,
        time, $brugger, $ipnr, $hostnavn, $mac);
    close udlog;
};

### main

($brugger, $ipnr, $hostnavn, $mac) = findid; # disse variable bruges globalt
skrivlog;

# Soerg for at disse maskiner ikke faar adgang til Internet
#
#

if (!forbindelseaaben ($ipnr))

```

```
{
  die "Forbindelsen er ikke åben\n";
};

open kontakt, ">>$kontaktfil";
print kontakt ($ipnr," lukkes\n");
close kontakt;

print "Din Internetforbindelse bliver lukket om kort tid!\n";
```

Bilag D

Arbejdsbeskrivelse

D.1 Formål

Formålet med dette projekt, er at identificere og analysere problemer ved central drift af nedenfor beskrevne netværk, og give løsningsforslag til disse problemer.

Endvidere vil vi implementere enkelte af løsningsforslagene, vi kan på nuværende tidspunkt af naturlige årsager ikke sige hvilke.

D.2 Baggrund

Netværket vi vil analysere i dette projekt, består af en række lokalnetværk der er spredt ud over et større geografisk område. De enkelte lokalnet er autonome, d.v.s. at der ikke er nogen kontrol over udstyret som er forbundet til netværket. Lokalnetværkene er forbundet til et antal backbonenetværk som kontrolleres af trediepart. De enkelte lokalnetværk er befolket af individer der kan have kontrol over udstyr tilknyttet lokalnetværket.

På backbonenetværkene er der et antal gateways til Internet. Det enkelte individ er tilknyttet en eller flere af disse gateways hvorigennem dennes trafik skal routes.

Det er ikke acceptabelt at give direkte adgang fra lokalnetværket til backbonenetværkene da det ikke vil være muligt at spore netværkstrafik til et enkelt individ hvilket blandt andet medfører at:

- der er ingen kontrol med hvor meget de enkelte individer bruger Internet og der er risiko for urimelig fordeling af ressourcerne
- der er ikke mulighed for at spore kriminel aktivitet, dette er et problem for den juridisk ansvarlige for netværket
- nogle Internetgateways må kun benyttes af bestemte individer. For at lokalnetværket skal have mulighed for at benytte disse er det krævet trafikken der routes til disse kan spores til et individ.

Det er kun muligt at opnå kontrol over trafikken ved at placere en enhed mellem lokalnettet og backbonenetværkene. Denne er primær fokus for projektet.

D.3 Motivation og målsætning

Den aktuelle problemstilling vi beskæftiger os med udspringer fra vores daglige tilknytning til kollegienet i København, herunder Foreningen Kollegienet København. Projektets formål er en generalisering af problemstillinger ved kollegienet.

Forskningsnettet i Danmark tillader at studerende ved forsknings institutioner gratis kan benytte Forskningsnettes Internetforbindelse. Ikke alle beboere på kollegier er studerende ved forskningsinstitutioner, men kollegierne har stadig interesse i at benytte denne gratis Internetforbindelse til dele af deres trafik.

Rent teknisk er det problematisk at benytte flere gateways til Internet, hvis man samtidig ønsker at give tovejs adgang til Internet.

Kollegierne ønsker en tilslutning til Internet som i drift er nem, billig og stabil. Det er derfor vigtigt at vores løsningsforslag er billige at implementere og ikke mindst billige at vedligeholde. Vi forventer at kunne opnå dette via central drift.

D.4 Aflevering

Vi ønsker at aflevere en rapport hvori problemstillingerne identificeres og analyseres og hvor kommer vi med løsningsforslag og kompromiser.

Vi ønsker desuden at implementere enkelte af løsningsforslagene i form af kildetekst og beskrivelse af dennes virkemåde. Kildetekst og beskrivelse ønsker vi at afleverer sammen med rapporten.

Vi vil også være interesseret i at lave en handlingsplan for videre udvikling på projektet set med relation til Foreningen Kollegienet København. Handlingsplanen ønsker vi at aflevere sammen med rapporten.