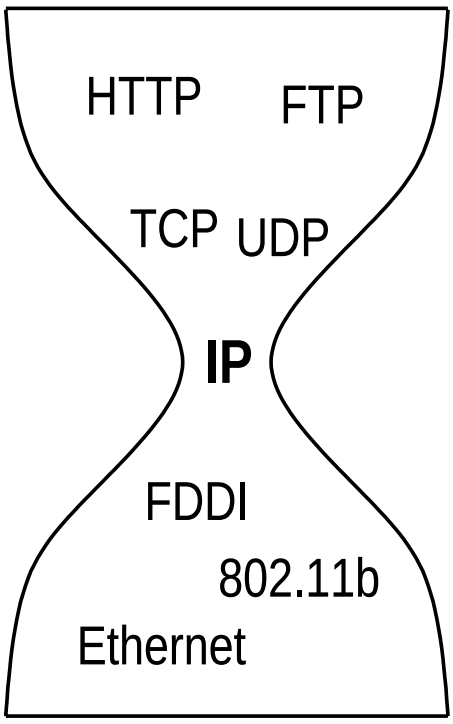
FOI-R--0309--SE
December 2001
ISSN 1650-1942

Användarrapport

Mats Persson

Säkerhet i aktiva nät



Ledningssystemteknik
581 11 Linköping

TOTALFÖRSVARETS FORSKNINGSINSTITUT

Ledningssystemteknik
Box 1165
581 11 Linköping

FOI-R--0309--SE

December 2001

ISSN 1650-1942

Användarrapport

Mats Persson

Säkerhet i aktiva nät

Utgivare Totalförsvarets Forskningsinstitut - FOI Ledningssystemteknik Box 1165 581 11 Linköping	**Rapportnummer, ISRN** FOI-R--0309--SE	**Klassificering** Användarrapport
	Forskningsområde 4. Spaning och ledning	
	Månad, år December 2001	**Projektnummer** E7023
	Verksamhetsgren 1. Forskning för regeringens behov	
	Delområde 41 Ledning med samband och telekom och IT-	
Författare/redaktör Mats Persson	**Projektledare** Alf Bengtsson	
	Godkänd av Lennart Nyström	
	Uppdragsgivare/kundbeteckning FM	
	Tekniskt och/eller vetenskapligt ansvarig Lennart Nyström	

Rapportens titel

Säkerhet i aktiva nät

Sammanfattning (högst 200 ord)

Rapporten beskriver aktiva nät och vad de kan användas till. De forskningsresultat som hittills framkommit beskrivs översiktligt och i rapporten görs även en analys och värdering av säkerheten i några implementationer av aktiva nät. Med aktivt nät menas ett nät vars funktionalitet kan ändras genom att programkod, så kallad mobil kod, skickas mellan nätets olika noder. På detta sätt kan man behandla nätet som en ”gemensam delad dator”. Redan idag kan man se tydliga tendenser på att nätet blir alltmer aktivt. Exempel på detta är brandväggar och webproxies. Dessa kan konfigureras och styras med snabb laddning av ny programkod. En nackdel är att denna form av programkod leder till större säkerhetsrisker.

Nyckelord

aktiva nät, IT-säkerhet, mobil kod

Övriga bibliografiska uppgifter	**Språk** Svenska
ISSN 1650-1942	**Antal sidor:** 24 s.
Distribution enligt missiv	**Pris: Enligt prislista** **Sekretess**

Issuing organization
FOI – Swedish Defence Research Agency
Command and Control Warfare Technology
P.O. Box 1165
SE-581 11 Linköping

Report number, ISRN
FOI-R--0309--SE

Report type
User report

Research area code
4. C4ISR

Month year
December 2001

Project no.
E7023

Customers code
1. Research for the Government

Sub area code
41 C4I

Author/s (editor/s)
Mats Persson

Project manager
Alf Bengtsson

Approved by
Lennart Nyström

Sponsoring agency
FM

Scientifically and technically responsible
Lennart Nyström

Report title (In translation)
Security in Active Networks

Abstract (not more than 200 words)
The report describes active networks and how they can be used. Current research results are briefly described and the report also contains an analysis and evaluation of the security in some implementations of active networks. The functionality of an active network can be changed by sending program code (mobile code) between the nodes in the network. This makes it possible to treat the network as a programmable shared computer. Even today you can see noticeable trends that the networks become more active. Examples are firewalls and webproxies, which can be configured and controlled by fast loading of new program code. The disadvantage is increased security risks when using this form of mobile program code.

Keywords
active networks, computer security, mobile code

Further bibliographic information

Language Swedish

ISSN 1650-1942

Pages 24 p.

Price acc. to pricelist

Security classification

Säkerhet i aktiva nät

Mats Persson

1 februari 2002

Sammanfattning

Rapporten beskriver aktiva nät och vad de kan användas till. De forskningsresultat som hittills framkommit beskrivs översiktligt och i rapporten görs även en analys och värdering av säkerheten i några implementationer av aktiva nät. Med aktivt nät menas ett nät vars funktionalitet kan ändras genom att programkod, så kallad mobil kod, skickas mellan nätets olika noder. På detta sätt kan man behandla nätet som en "gemensam delad dator". Redan idag kan man se tydliga tendenser på att nätet blir alltmer aktivt. Exempel på detta är brandväggar och webproxies. Dessa kan konfigureras och styras med snabb laddning av ny programkod. En nackdel är att mobil kod leder till större säkerhetsrisker.

Innehåll

Summering

Vi blir allt mer beroende av ett fungerande datornätverk och kommer att ställa allt högre krav på dess funktion och även krav på att säkerheten skall vara tillfredställande. Den typ av kommunikationsprotokoll som är vanligast idag är både pålitligt, skalbart och har goda prestanda. Men det är inte lämpat för alla typer av informationsöverföring och har inte tillräcklig säkerhet, vilket man kan tydligt kan se vid alla de överbelastningsattacker som förekommer på Internet.

Aktiva nät

Våra datorer brukar vara sammankopplade i ett nätverk. Detta innehåller inte bara kommunikationslänkar utan även mellanliggande noder som dirigerar trafiken åt olika håll. Ett aktivt nät är ett nät med kraftigt utökad funktionalitet i de mellanliggande noderna genom att de är fullt programmerbara. Detta innebär att nya kommunikationsprotokoll enkelt kan programmeras in i nätet. Därmed slipper man tidigare problem med otillräcklig funktionalitet vid särskilda dataöverföringar. De mellanliggande noderna kan även förses med speciell programvara för att hantera övervakning och styrning av trafiken och förhoppningsvis stoppa många nätverksattacker.

Denna omprogrammering av mellannoderna möjliggörs genom att programkod skickas i datapaketen och denna kod programmeras sedan in i noden när den kommer fram. Detta är en form av mobil kod, vilket förstås ger en mängd nya säkerhetsproblem. Varje paket måste autenticeras, kontrolleras, ges rättigheter och sedan placeras i en säker exekveringsomgivning innan programkoden kan börja användas. Om man inte gör detta får man problem med elakartad programkod i form av virus.

Slutsatser

Ett dilemma uppkommer. Man vill ha högre funktionalitet och förbättrad säkerhet i nätet, men får istället problemen med mobil kod att ta hand om. Samtidigt gör de aktiva näten att nätet blir mer komplext, vilket gör att det blir svårare att förutsäga resultatet av små förändringar. En konsistent säkerhetspolicy blir svår att upprätthålla.

En annan slutsats man kan dra är att man redan idag kan se tydliga tendenser på att nätet blir alltmer aktivt. Exempel på detta är brandväggar och webproxies, som kan konfigureras och styras med snabb nerladdning av ny programkod. Framtida nät kommer förmodligen att innehålla någon typ av aktiv nod som fungerar som knutpunkt och har både brandväggsfunktionalitet, filtrering, mellanlagring, bandbreddsbegränsning och dynamisk uppladdning av nya protokoll.

1 Inledning

I projektet “Försvarsspecifik IT-säkerhet” har området mobil kod tidigare studerats. Mobil kod har visat sig förekomma i många sammanhang, allt från scripts i webbläsare till elaka datorvirus. Ett nytt intressant användningsområde för mobil kod är styrning av nät. Enkelt uttryckt innebär det att mobil kod skickas ut i nätverket och manipulerar eller programmerar om nätverksnoderna.

Denna rapport kommer att ge en översikt om detta nya forskningsområde och ge ett sammandrag av de resultat som hittills framkommit. Det finns för närvarande ett antal implementationer av aktiva nät och även dessa kommer att översiktligt beskrivas. Fördelar och nackdelar med användandet av aktiva nät kommer att utvärderas ur säkerhetssynpunkt, och även säkerhetsfunktionaliteten i själva arkitekturen kommer att belysas.

Syftet med denna rapport är att visa på en tänkbar utveckling av nätverken och vilka säkerhetsproblem detta kan innebära. Speciellt när mobil kod används för att öka funktionaliteten i nätverken eftersom detta kan innebära nya säkerhetsproblem. Motiveringen till att studera aktiva nät är att vi snart kommer att befinna oss i ett nätverkssamhälle och utnyttja många av dess tjänster, men samtidigt kommer vi att bli beroende av att nätet fungerar. Ett annat tydligt tecken på detta är att försvarets nya ledningssystem kommer att baseras på ett nätverksförsvar.

Mängden nätverk och kraven på dessa kommer också att öka. Den kommersiella sektorn satsar stora pengar på tjänster i datornäten. Ett exempel på detta är .NET-initiativet från Microsoft där man väljer en nätverkscentrerad lösning på införandet av nya tjänster. Ett annat exempel är Jini från Sun som är ett system för att koppla samman och snabbt konfigurera utrustning i nätet. Jini skickar mobila *service objects* och flyttar därmed abstraktionsnivån från nätverksprotokollnivån till objektnivån.

Rapporten kommer först att ge en enkel introduktion till vanliga nätverk och aktiva nät i kapitel 2. För att ytterligare motivera aktiva nät så tas ett antal användningsområden upp i kapitel 3. Några exempel på hur ett aktivt nät kan vara uppbyggt visas i kapitel 4 och en analys av säkerheten görs i kapitel 5.

2 Bakgrund

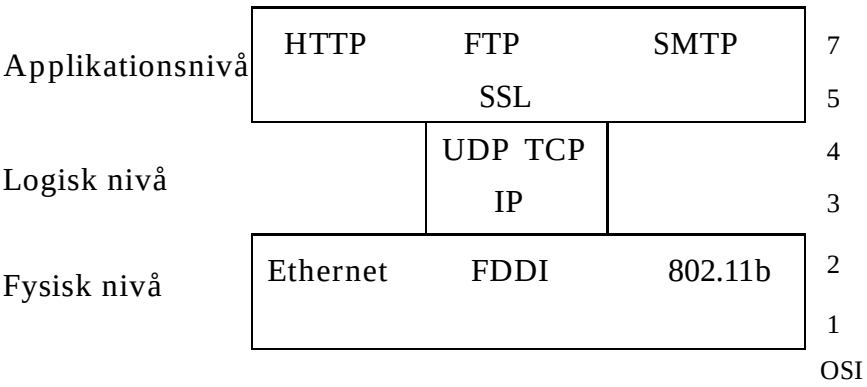
För att kunna förklara vad ett aktivt nätverk är och hur man kan gruppera nätverksnivåerna, görs här en kort introduktion till nätverk. För en mer utförlig beskrivning av datornätverk rekommenderas boken “Computer Networks” [1].

De flesta datorer är anslutna till någon form av nätverk och detta består rent fysiskt av kablar och ihopkopplingspunkter. I dessa punkter kan det sitta ett nav som enbart skyfflar datapaket eller en router som dirigerar datatrafiken.

För att datorerna skall kunna kommunicera med varandra räcker det inte med enbart en kabel, utan det behövs även ett nätverksprotokoll. Det numera vanligaste protokollet är IP. I detta protokoll finns regler för hur ett datapaket som skickas i det fysiska nätet skall se ut och hur man adresserar datorerna så att paketen hittar rätt.

Dessutom behövs protokoll för användartjänster, som till exempel epost och webb. Dessa protokoll reglerar handskakningsprocedurer och format på meddelanden.

Utifrån ovanstående blir det grovt sett tre nivåer: den fysiska med kablar och kopplingspunkter, den logiska med pakethantering, och applikationsnivån med tjänster. En annorlunda uppdelning kan göras med hjälp av OSI-modellen med sju olika nivåer men detta gör inte förklaringen enklare (rentav försämrande som förklaras längre fram). De tre nivåerna skall ses som staplade ovanpå varandra. Den logiska nivån utnyttjar den fysiska och applikationsnivån utnyttjar den logiska nivån. Se även figur 1 där även namnen på ett antal olika protokoll finns med. Notera att IP är förhärskande på den logiska nivån.



Figur 1: Nivåer av nät

2.1 Passiva nätverk

De nätverk som finns idag är egentligen inte särskilt passiva och ordet “passiv” används här för att skilja från aktiva. Passiva nät har en viss dynamik i protokollen och routingalgoritmerna. Men i jämförelse med aktiva nät är de ändå rigida i sitt beteende.

Det mest använda protokollet på Internet är IP. Det är en fördel att använda ett sådant standardiserat och enkelt protokoll eftersom det leder till mindre problem i implementationer och vid sammankoppling av olika nät. Innan TCP/IP blev den gemensamma standarden fanns det ett stort antal företagsägda protokoll. Varje datortillverkare hade sin egen standard och detta ledde till stora problem om två olika datorer skulle fås att kommunicera med varandra. Internet växte snabbt upp när TCP/IP etablerades på allvar.

Det finns även standardiserade protokoll för de andra nivåerna, som till exempel ICMP för styrpaket och HTTP för webbtjänster. För den logiska nivån finns det nästan enbart IP. Detta på grund av de interoperabilitetsskäl som nämnts tidigare. Att det bara finns ett protokoll har förstås även sina nackdelar.

Under den tid TCP/IP använts har man funnit många begränsningar och problem [2]:

- Begränsad mängd IP-adresser. Just nu används 60% av den tillgängliga adressrymden och resten räcker inte hur länge som helst.
- Multicast fungerar inte särskilt väl. Multicast innebär att sända till många mottagare samtidigt.
- Problem med nätverksattacker som till exempel *distributed denial of service*, vilket betyder distribuerad överbelastningsattack.
- Möjligheter till spoofning av IP-adresser vilket ger svårigheter med spårning av attacker. Spoofning innebär att man luras att tro att man kommunicerar med rätt dator. I själva verket har en annan dator tagit dess adress eller IP-nummer.
- Dålig cachning av webtjänster i TCP. Om många laddar ner samma stora fil mellanlagras den inte, utan skickas flera gånger. Å andra sidan är inte den logiska nivån lämpad för denna typ av funktionalitet.
- Mindre bra support för mobil IP.
- Svårigheter med att upprätthålla kvalitén på förbindelserna (Quality of Service).
- Mindre lämpat för radioförbindelser med höga paketförluster eller satellitförbindelser med långa fördröjningar. Detta gäller i första hand TCP.
- Redundanta operationer på grund av för många nivåer av protokoll kan ge dåliga prestanda. (Detta är en anledning till den förenklade uppdelningen i tre nivåer i figur 1).

Man måste konstatera att TCP/IP konstruerades för enklare tillämpningar, och att vissa applikationer har högre krav. De kräver mer specialiserade protokoll.

2.2 Aktiva nät

På samma sätt som man kan programmera ändnoderna, som vanligen är datorer, borde man kunna programmera nätet mellan dem. Som det är nu, är nätet enbart konfigurerbart genom att man med ett enkelt kommando

kan aktivera ett antal avancerade funktioner i noden. Med ett programmerbart nät har man istället många primitiva funktioner som kan kombineras till en oändlig mängd avancerade funktioner. Motsvarande införande av programmerbarhet har redan skett i skrivare där programspråket PostScript har införts.

Visserligen är nätet redan till viss del “programmerbart”. Det kan vara i form av avancerade konfigurerbara brandväggar som har programmerbara filter eller webproxies som genom särskilda algoritmer kan lagra websidorna på de platser där de oftast används eller läses. Man kan även påstå att nätet redan är “aktivt” vid till exempel dynamisk tilldelning av IP-adresser. Detta går till så att när en dator kopplas in på nätet så skickar den ut en förfrågan om vilken IP-adress den skall använda. När den får svaret sätter den sin IP-adress till det värde den blev tilldelad.

En annan fördel med aktiva nät är att man har möjlighet att lösa framtida problem i nätverken på ett smidigare sätt. Näten kan rentav bli självupprätthållande eller självläkande vid nätverksattacker. Skydd mot nätverksattacker tas upp längre fram i den här rapporten.

Ett aktivt nätverk använder sig av mobil kod för att dynamiskt kunna lägga in nya tjänster och protokoll i nätet. När det kommer en ny typ av data som behöver ett nytt specialiserat protokoll, kan det enkelt läggas in i de aktiva noderna i nätet. En aktiv nod kan vara en router eller en dator med speciell programvara för att ta hand om de speciella datapaket som skickas mellan noderna.

Aktiva nät kan finnas i alla tre nätverksnivåerna, från det fysiska lagret och uppåt, beroende på abstraktionsnivå för protokollet eller tjänsten. Aktiv paketfiltrering finns på den fysiska nivån. Nya kommunikationsprotokoll läggs oftast in på den logiska nivån. På applikationsnivån finns dynamiska och aktiva tjänster, men dessa kan indirekt lägga in nya protokoll på den logiska nivån.

Det finns två allvarliga problem med aktiva nät. Det första är prestandaförluster, eftersom det finns extra information i paketen. Denna extra information kan vara typ-information eller rentav exekverbar kod. Det tar en del extra tid att undersöka alla paketen och detta kan ge märkbara tidsfördröjningar när paketen passerar många aktiva noder.

Det andra problemet är säkerheten. Det är förstås alltid en säkerhetsrisk att exekvera mobil kod, vilket har konstaterats tidigare [3]. Säkerheten kommer att behandlas i kapitel 5.

3 Användningsområden för aktiva nät

Det finns många potentiella användningsområden för aktiva nät. Generellt sett kan de vara användbara när man behöver införa nya protokoll snabbt utan att behöva förlita sig på att all annan utrustning och programvara känner till det nya protokollet. Resten av detta kapitel tar upp flera

tänkbara användningsområden. Allt från sådana som finns redan idag, till de mer spekulativa som enbart finns på tankestadiet.

3.1 Övervakning av nätverk

Nätverk behöver kontrolleras och övervakas. Det är bäst om nätverken till stor del själva kan ta hand om rutinhändelser i nätet och ställa om sig själva vid behov. Detta gör många routingprotokoll redan. I ett aktivt nät kunde noderna själva injicera övervakningskod i närliggande noder och sedan filtrera bort ointressant information. Detta kan även fungera som avancerad intrångsdetektering som tidigt kan förvarna resten av noderna eller övervakarna om framtida problem. Nätet kan även upptäcka felaktiga eller förstörda noder, varna för dessa och skicka paketen runt dem. Om två nät kopplas samman kan de båda aktiva näten upptäcka detta och om möjligt kartlägga varandras nät.

3.2 Adaptiv routing

Redan idag är routers i viss mån aktiva. De har ganska avancerade operativsystem och använder komplexa routingprotokoll som kan hantera dåliga förbindelser, borttappade noder eller mobila noder. En naturlig fortsättning vore att göra dem ännu mer aktiva och programmerbara.

3.3 Virtuella privata ad hoc nätverk

En annan möjlighet med aktiva nät är att sätta upp ett virtuellt nät bestående av distribuerade noder. Detta nät kan till exempel innehålla ett distribuerat filsystem eller databas. På samma sätt kan man sätta upp peer-to-peer system via detta virtuella nät.

När allt fler delar av Internet skyddas av brandväggar eller när nät läggs bakom en gateway, skulle man vilja upprätta egna privata kanaler genom dessa. Detta kan man lösa genom att sätta upp en aktiv nod mellan näten och inuti brandväggen. Noden autenticerar varje paket och tillåter endast ett begränsat antal operationer. Denna nod kan då skapa en privat kanal och bli en del av det virtuella nätet.

3.4 Upprätthålla QoS och hindra DDoS

I aktiva nät finns det ofta möjligheter att begränsa exekveringstid eller räkna antalet paket från en viss adress. I och med att paketen redan är autenticerade vet man vem de tillhör. Detta gör att man kan fördela bandbredden för olika användare och sannolikt även garantera viss överföringshastighet, så kallad *Quality of Service*.

Det finns en välkänd nätverksattack som genom att ett större antal datorer utspridda i nätet exakt samtidigt skickar paket till samma adress kan få datorn som har den IP-adressen att bli överbelastad. Denna attack kallas

på engelska *Distributed Denial of Service*, vilket kan översättas till distribuerad överbelastningsattack på svenska. Som Internet är konstruerat idag finns inga riktigt bra motmedel mot dessa attacker. Men om nätet istället var aktivt, och därmed kunde reducera paketmängden eller filtrera bort lämplig mängd paket, så kan denna attack bli ett betydligt mindre problem.

Brandväggar har redan filtrering av paket, och de mer avancerade varianterna på brandväggar har programmerbara filter. Om brandväggarna sinsemellan kunde utbyta filtreringsinformation så kunde blockering av attacken tryckas ut närmare anfallarens alla distribuerade datorer.

3.5 Informationsbehandling

Två andra områden där aktiva nät kan göra nytta är informationsfusion och webproxies. Informationsfusion innebär att en stor mängd information från många olika sensorer behandlas och enbart den relevanta informationen behålls. En webproxy agerar som mellanlagringsplats till webservrar och kan även generera de dynamiska sidor som normalt genereras på webservern. I båda fallen handlar det om att mellanlagra information i de aktiva noderna för minska mängden nätverkstrafik, filtrera och mellanbehandla informationen och ge snabb respons på förfrågningar. Möjligtvis kan man även med hjälp av algoritmer automatiskt balansera placeringen av webproxies.

3.6 Autenticeringsmekanismer

Dagens nät har i de flesta fallen autenticeringsfunktionerna i ändpunkterna. Det är alltså i klienten och servern som autenticeringen sker. Nätet som ligger mellan dessa är oftast ovetande om vilken applikation paketen tillhör men kan förstås se avsändaradressen. Adressen kan lätt förfalskas. Detta faktum är förmodligen anledningen till att all autenticering pressats ut i ändpunkterna. Dessutom har man blivit beroende av autenticering i varje protokollager eftersom varje applikation har sina särskilda behov.

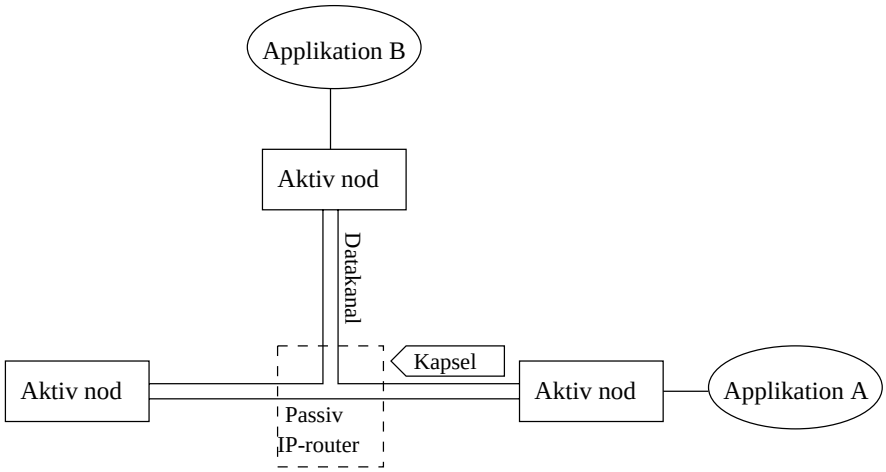
I ett aktivt nät kan säkerhets- och autenticeringsmekanismerna istället ligga i varje aktiv nod. Man kan då programmera in särskilda säkerhetspolicies i nätet för varje separat användare, applikation eller organisation. Därigenom kan man låta nätet ta hand om en del av autenticeringen.

4 Aktiva näts uppbyggnad

Detta avsnitt ger en allmän beskrivning av hur ett aktivt nät kan vara uppbyggt. Olika implementationer av aktiva nät kan variera i sin arkitektur, men det finns några generella principer som kommer att beskrivas här.

En förutsättning för att ett aktiv nät skall fungera är att det finns aktiva noder i nätet. Dessa aktiva noder är förpreparerade med speciella program, som tar hand om inkommande paket. Den mobila kod som används

i ett aktivt nät finns inuti datapaketen och paketen brukar då kallas kodkapslar (*code capsules* [4]), eller *switchlets* [5] om det rör sig om paketfiltrering på fysisk nivå. Dessa kodkapslar skickas ut i nätet och fångas upp av de aktiva noderna som sedan behandlar dessa kapslar. Den kod som finns i kapslarna är ofta nya protokoll som applikationer vill upprätta, men kan även innehålla kod för övervakning eller filtrering. Figur nummer 2 visar en principiell skiss på hur ett enkelt aktivt nät med tre aktiva noder kan se ut.



Figur 2: Aktivt nät med aktiva noder

Applikationer utnyttjar det aktiva nätet för att kommunicera med sina egna nätprotokoll. I figur 2 skickas en kodkapsel mellan två av applikationerna. Applikation A kontaktar sin aktiva nod och begär att den skall skicka data via ett särskilt protokoll. Den aktiva noden börjar då med att skicka en kodkapsel via datakanalen till den andra noden. Denna tar hand om kapseln och upprättar protokollet mellan applikation A och B. Sedan börjar dessa kommunicera via det egna nya protokollet.

I nätet finns även en passiv IP-router som inte behandlar kapslarna alls, utan agerar som en enkel paketförmedlare eller brygga. Vanligtvis används IP, men de prototyper på aktiva nät som finns implementerade utnyttjar dock inte alltid IP. Detta kan dock bli nödvändigt när tester mellan fysiskt distanserade datorer skall utföras.

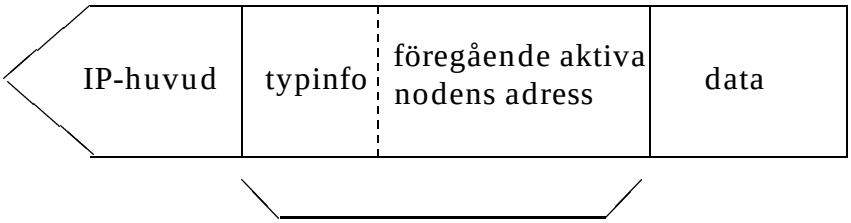
Det går att generellt dela in de aktiva näten i tre typer [6]:

- **Diskreta**. Skickar koden i separata paket via särskilda kanaler. Denna typ av nät behandlas enbart översiktligt i denna rapport.
- **Kod på begäran**. Datapaketen innehåller speciell typinformation och den mobila koden skickas på begäran. Denna typ behandlas mer ingående längre fram i rapporten.
- **Integrerade**. Alla datapaket kan innehålla mobil kod och exekveras i den aktiva noden om det behövs. För den mest integrerade formen finns det ingen kod i den aktiva noden för att behandla datapaketen. Istället har varje paket tillräckligt med kod för att hantera sig själv.

Det finns alltså två viktiga delar i ett aktivt nät, kodkapslar och aktiva noder. Dessa kommer att beskrivas i efterföljande avsnitt. Hur de aktiva noderna kopplas samman i ett nät har inte någon nämnvärd betydelse eftersom de aktiva noderna själva kommer att bygga upp topologin utifrån applikationernas behov. Detta är också upp till varje typ av implementation av aktiva nät att avgöra.

4.1 Kodkapslar

De datapaket som innehåller mobil kod brukar kallas kodkapslar eller *switchlets*. Rent principiellt skiljer sig inte kapslar och switchlets särskilt mycket, utan skillnaden ligger mer på hur den aktiva noden ser ut och i vilken ordning paketen skickas. En poäng med aktiva nät är att både kod och data skall kunna skickas i samma kanal. Därför behöver man markera vilka paket som är kod och vilka som är ren data. En kodkapsel kan se ut som i figur 3 nedan



Figur 3: Kodkapsel

Första delen av en kodkapsel består av ett vanligt IP-huvud som innehåller IP-adress och den andra vanliga informationen. Den andra delen av kapseln är ett extra huvud som innehåller typinformation som den aktiva noden sedan kan använda för att avgöra vilket protokoll som denna kapsel tillhör. Denna del innehåller även adressen från vilken aktiv nod denna kapsel kommer ifrån. Adressen används för att fråga föregående nod hur paketet skall behandlas. Sista delen av kapseln innehållen vanliga data som eventuellt kan vara exekverbar kod.

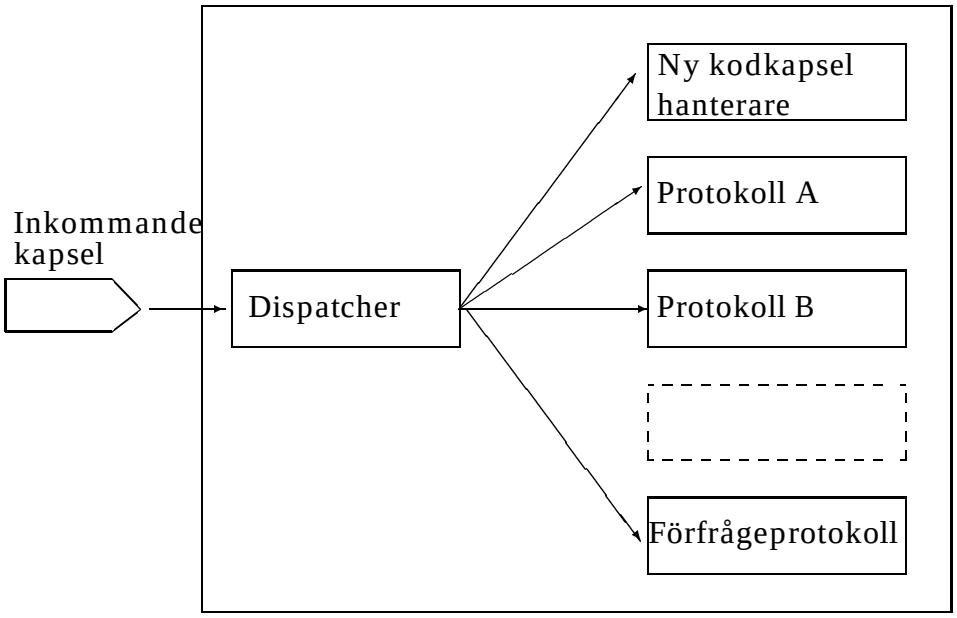
I en switchlet kan paketet sakna det vanliga IP-huvudet och datadelen enbart innehålla kod för att uppgradera den aktiva noden. Dock kan switchleten innehålla typinformation.

4.2 Aktiva noder

En aktiv nod innehåller speciell programvara för att ta hand om inkommande kodkapslar. Noden innehåller även ett gränssnitt som applikationer kan utnyttja för att skicka iväg kodkapslar innehållande nya protokoll. När en kodkapsel kommer in till den aktiva noden kontrolleras typen och kapseln skickas vidare till den modul som tar hand om just den typen

av kapslar. Om det inte finns någon sådan modul så finns det ett särskilt sätt att hantera detta. Man låter den aktiva noden begära kod för modulen från föregående nod som då skickar en kodkapsel innehållande kod för modulen ifråga. Se figur 4 som visar en förenklad bild på hur en sådan kapselhanterare kan se ut.

Andra typer av aktiva nät löser hanteringen av kod för protokoll på ett annat sätt. Detta sker genom att först distribuera ut lämplig programkod för modulen och sedan skicka data via det nya protokollet. Om en okänd typ av kapsel dyker upp slängs den.

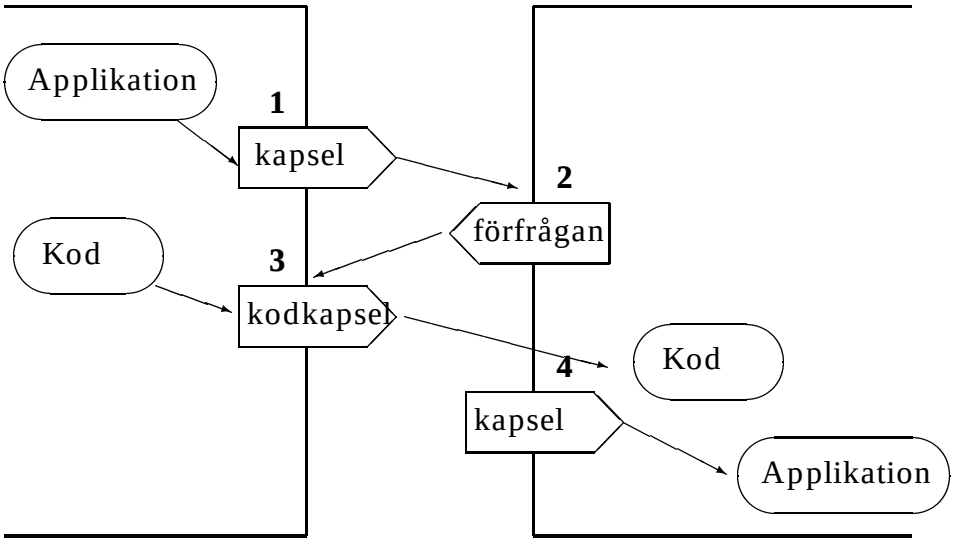


Figur 4: Kapselhanterare

Figur 4 visar den del av den aktiva noden som tar emot en kapsel. *Dispatchern* tittar på typen av kapsel och skickar den till rätt modul. Om det är ett känt protokoll så skickas den till modulen för det protokollet. Om det är en ny okänd typ av kapsel skickas den till *förfrågeprotokollet* som skickar en ny förfrågekapsel åt det håll varifrån den okända kapseln kom. Den nya kapseln begär in kod för den nya okända typen, och denna nya kod tas om hand av modulen *Ny kodkapselhanterare*. En bild på hur kapslarna skickas finns i figur 5.

4.3 Olika implementationer av aktiva nät

Det finns ett antal implementationer av aktiva nät. Många av dem är forskningsprototyper. De som har fått mest uppmärksamhet redovisas här nedan.



Figur 5: Nerladdning av kod på begäran

ANTS

Detta är ett aktivt nät som utvecklats på MIT [7]. Det är baserat på Java och använder sig av dynamisk laddning av protokoll vid behov. Detta system behandlas mer ingående i kapitel 4.4.

Netscript

De som utvecklar Netscript [8] på Columbia University har inriktat sig på att göra ett programspråk för programmering av nätverk. Netscript innehåller ett antal grundläggande funktioner (eller komponenter) för att styra nätverket som helhet, och med dessa funktioner kan man sedan sätta ihop mer komplexa protokoll i nätet. Netscript lägger sig på en högre nivå än att lägga in kapslar i aktiva noder. Detta språk kan lämpa sig väl för övervakning och filtrering av paket, och kanske rentav utföra brandväggsfunktioner på andra noder än ens egna.

SwitchWare

Denna prototyp har utvecklats på UPenn [5]. Den är tänkt att fungera som en avancerad router eller switch som kan programmeras om med mobila *switchlets*. Dessa innehåller kod för nya protokoll. SwitchWare bygger på att det skall finnas en fristående enhet med ett eget inbyggt operativsystem som tar emot nya komponenter skrivna i Caml (som är ett programspråk som är gjort för att lätt kunna verifieras formellt). Dessa komponenter är sedan tidigare signerade och typkontrollerade innan de skickas ut i nätet. SwitchWare har prioriterat säkerheten framför prestanda och användbarhet. Ett besläktat projekt finns på Bellcore där de istället fokuserar

på prestanda i sina höghastighetsnät. Notera att SwitchWare är av den integrerade typen av aktiva nät som nämns i kapitel 4.

CANEs

CANEs [9] [10] har utvecklats på Georgia Tech University och är en typ av aktivt nät där applikationerna kan styra nätet genom att begära att olika algoritmer används i noderna. Dessa algoritmer kan vara kompression eller omkodning av data. Nya tjänster eller algoritmer i nätet kan sättas ihop av tidigare existerande komponenter. Dessa kan rentav vara gjorda i ANTS eller Netscript som nämnts ovan.

SmartPackets

Dessa "smarta paket" har utvecklats på Kansas University [11] [12]. Paketen innehåller Java-kod och denna kod utnyttjar de funktioner för resurshantering som finns i varje nod. Det har gjorts implementationer för HTTP och SMTP protokollen som visar vissa prestandaförbättringar. De som utvecklade SmartPackets observerade att det normalt blir väldigt många HTTP uppkopplingar när man laddar ner en websida. Detta resulterar i onödigt många ACK/NACK paket i TCP/IP förbindelserna. SmartPackets har löst detta genom att lägga upp en särskild kanal för hela nerladdningen och slipper därmed alla överflödiga paket.

SOFTNET

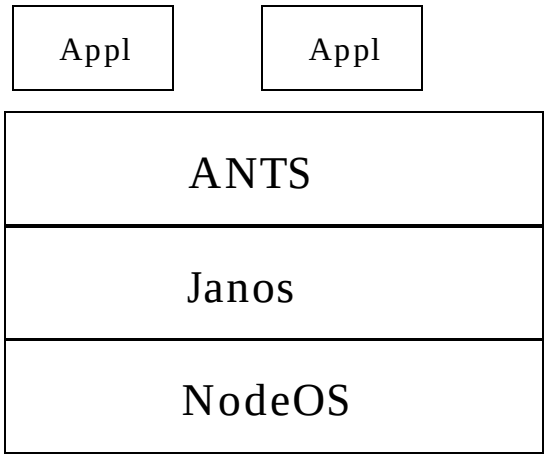
Redan 1983 konstruerades ett aktivt nät av Jens Zander och Robert Forchheimer på Linköpings universitet. Det kunde skicka paket innehållande FORTH-kod till noderna. SOFTNET blev inte så vida spritt, kanske på grund av säkerhetsproblem och den låga prestandan i näten på den tiden. Däremot inspirerades dagens forskning om aktiva nät av deras ideér.

Dessutom finns ett antal närbesläktade projekt som mest berör telekommunikation och trådlösa nät. Andra implementationer handlar mer om programmerbara switchar och då utan mobil kod i traditionell mening. Ingen av dessa kommer att tas upp i denna rapport.

4.4 Närmare titt på ANTS och Janos

Vanligtvis är systemen för aktiva nät uppdelade i flera nivåer. Dessa består nerifrån och upp av drivrutiner för att ta hand om hårdvaran, ett operativsystem för att hantera kommunikation och andra systemrutiner, och en nivå för den aktiva noden. Ovanpå allt detta körs applikationerna som kan beskådas i figur 6.

Janos är ett operativsystem som körs i en aktiv nod och ANTS är det system som tar hand om paket och protokoll i det aktiva nätet. ANTS kör ovanpå Janos och utnyttjar dess systemrutiner. Både Janos och ANTS är



Figur 6: Arkitekturen för ANTS

skrivna i Java. I princip fungerar ANTS som tidigare beskrivits i kapitel 4 och har alltså dynamisk nerladdning av java-kod på begäran.

ANTS

ANTS har utvecklats på MIT och förkortningen står för "Active Node Transfer System". Systemet är som tidigare nämnts Javabaserat, och använder sig i den senaste versionen även av Javas säkerhetmodell med säkerhetspolicy och referensmonitor [13]. Arkitekturen i ANTS består huvudsakligen av tre delar. I nedanstående uppräkning nämns också det som specifikt skiljer sig från den tidigare modellen med kod på begäran:

- **Kodkapslar**. Dessa ser ut som beskrivits i kapitel 4.1 men de har även ett fält med en MD5 hashsumma. Denna hash används dels för att identifiera det protokoll som kapseln tillhör, dels för enkel verifiering av kapselhuvudet och därmed undvika problem vid injektion av felaktiga kapslar i nätet. Varje kapsel har dessutom ett *time-to-live*-fält som anger hur mycket resurser denna kapsel får använda. Värdet räknas ner varje gång en resurs används och när värdet blir noll kastas kapseln bort. En resurs kan vara CPU-tid eller bandbredd.

- **Exekveringsomgivning**. Den omgivning som kodkapslarna kommer att exekvera i när de anländer till en aktiv nod innehåller ett antal primitiver. Dessa primitiver är av typerna: *tillgång till omgivningen* för att få status på nätverkslänkar, routertabeller eller lokal tid; *manipulering av kapslar* för att läsa innehållen i huvud och datafälten i andra kapslar; *styroperationer* för att skapa nya kapslar, kopiera eller skicka vidare kapslar; *lagringsprimitiver* för att mellanlagra de temporära objekt som skickas av applikationerna.

- **Koddistribuering**. Kod skickas på samma sätt som beskrivs i kapitel 4.2. Om en nod saknar Java-kod för ett visst protokoll frågar den

föregående nod och begär kod. I ANTS har man delat upp koden i kodgrupper som är moduler som är beroende av varandra, och som därför skall skickas samtidigt.

Ett möjligt problem som ANTS inte hanterar är när ett paket tappas bort vid nerladdning av ny kod. Detta fel måste högre nivåer ta hand om, vilket kan bli svårt om ANTS inte indikerar felet på ett bra sätt. Överhuvudtaget är ANTS lite känsligt för störningar, eftersom det finns för många steg i protokollen. Ett annat problem, som kanske är enklare att åtgärda, är hur och när kod som inte används skall avallokeras i varje nod.

Janos

Janos är ett speciellt anpassat operativsystem som körs i den aktiva noden. Det hanterar resurser som CPU-tid och minne, tar hand om paketbuffrar, skapar separerade exekveringsomgivningar för varje applikation, begränsar bandbredden, och har en sandlåda för opålitlig inkommande programkod. Janos i sin tur kör ovanpå ett NodeOS som enkelt uttryckt är drivrutiner för hårdvaran. Se figur 6.

I ett senare skede och som en fortsättning på denna rapport kommer en demonstrator för aktiva nät att byggas med hjälp av ANTS och Janos.

5 Säkerheten

Det finns två sätt att diskutera säkerheten för aktiva nät. Det ena sättet är att visa hur säkerheten i datornäten kan förbättras med hjälp av aktiva nät. Exempel är att förhindra DDoS som nämns i kapitel 3.4, förstärkning av skydden i nätverk nämnt i kapitel 3.6 eller bättre övervakning som tas upp i kapitel 3.1. Då fokuserar man inte på de aktiva noderna och kodkapslarna som separata logiska komponenter, utan ser istället på aktiva nät som ett system för skydd av kommunikationskanalerna.

Det andra sättet att diskutera säkerheten är att titta på de aktiva näten själva och visa på vilka risker som följer av att nätet exekverar mobil kod. Resten av detta kapitel kommer att handla om denna aspekt.

5.1 Säkerheten i implementationerna

De implementationer som nämnts tidigare i denna rapport har i de flesta fall ägnat en del uppmärksamhet åt säkerheten. Två av dem har studerats lite noggrannare, ANTS / Janos och SwitchWare. Båda har ett underliggande “operativsystem” som hanterar det mesta av säkerheten.

Janos är som tidigare nämnts baserat på Java. Säkerheten består till stor del av säkerheten i Java’s inbyggda typsäkerhet, men även av autentisering av inkommande kapslar. Tyvärr verkar implementationen köra på Linux och dess fria implementation av JavaVM som kallas Kaffe [14]. Denna

har för närvarande ingen typsäkerhetskontroll och därmed faller ganska mycket av säkerheten i Janos.

SwitchWare [15] genomför autenticering av inkommande paket, kontrollerar vem som skrivit paketen och ger dem därefter lämpliga rättigheter. Systemet har en säker uppstartsekvens där varje systemlager som startas är kontrollerat med digitala signaturer. Detta skall ge systemet ett garanterat säkert tillstånd när uppstartsekvensen är klar. På samma sätt kontrolleras inkommande kod med hjälp av digitala signaturer. De som utvecklat systemet säger sig använda en enkel PKI-struktur för de digitala certifikaten men går tyvärr inte in på hur nycklarna skall distribueras. Ej heller nämner systemutvecklarna något om att ett så dynamiskt system som en aktiv nod kan hamna i ett oförutsett tillstånd där inga garantier för säkerheten finns kvar. Vad händer om till exempel en nod blir överlastad med paket och får slut på resurser? Ingen sådan analys finns. Istället fokuserar man på den säkra uppstarten.

Man är alltså medveten om säkerhetsproblematiken men den får ofta stå i skymundan bakom kraven på höga prestanda.

5.2 Analys med sandlådekriterierna

I en tidigare FOA-rapport [3] studerades säkerhetsproblemen som uppstår om man låter främmande mobil programkod skickas över ett datornätverk för att köras på andras eller egna datorer. Den säkerhetsmodell som många system använder är den så kallade sandlådemodellen. Den innebär att man kapslar in koden i en begränsad omgivning där den inte kan ställa till med så mycket skada. Koden kommer inte åt variabler och minne utanför sandlådan, den kan ha begränsningar på tid och minnesutrymme som den får exekvera i, och den får begränsad tillgång till filsystem och nätverk via ett mer eller mindre avancerat behörighetssystem.

I denna tidigare FOA-rapport finns även ett antal kriterier som måste vara uppfyllda för att en sandlåda eller exekveringsomgivning skall anses som tillräckligt säker. Dessa kriterier kan användas för aktiva nät i allmänhet. Detta har gjorts i tabell 1, där en enkel bedömning gjorts för hur väl kriterierna uppfyllts.

	Säkerhetskrav	Aktiva nät
1	Verifikation	Verifikation av kod måste ske precis innan exekvering vilket inte något aktivt nät gör. Om JavaVM används så verifieras all kod, men tyvärr saknas ofta detta.
2	Spärrade funktioner	Ingen tillgång till andra I/O-enheter brukar finnas eller behövas, förutom till nätverket förstås.
3	Spärrad omgivning	Tillgång till statistik och systemspecifikation brukar finnas eftersom detta behövs för att upprätthålla QoS eller för övervakning.
4	Minneskydd	Eftersom mycket data passerar, har ofta tillräckligt med begränsningar i minneshanteringen lagts in.
5	Autenticering	Finns ofta i designspecifikationen men är inte alltid implementerat.
6	Exekvera program	Inget aktivt nät tillåter start av yttre program.
7	Auktorisering	Behörighet för kanaler och bandbredd finns ibland.
8	Begränsa CPU	Finns oftast.
9	Kryptering	Nej, detta är upp till varje protokoll. Åtgång av CPU-tid kan dock sätta käppar i hjulen.
10	Loggning	Nej.
11	Samlad säkerhet	*ej kontrollerat*
12	Enkelt system	*ej kontrollerat*
13	Ej maskinkod	Något av systemen tillåter förkompilerad och kontrollerad maskinkod.

Tabell 1: Kontroll på hur väl aktiva nät uppfyller säkerhetskraven för sandlådor

Av denna enkla analys kan man se att många förbättringar kan göras, främst på den delen som sker precis innan en kapsel exekveras, dvs punkterna 1, 5 och 7. Däremot verkar skyddet tillräckligt när kapseln exekveras. Det som kanske saknas i dessa kriterier är hur väl systemet uppför sig vid överbelastning eller oväntade situationer, eller hur ofta systemet städar bort gammal kod och data. Detta är förmodligen en viktigare aspekt i aktiva nät jämfört med andra system för mobil kod.

Autenticering

Autenticering i nätverk är ett komplicerat problem och detta syns tydligast i aktiva nät. På grund av sin distribuerade natur blir detta mycket svårare i nät jämfört med autenticering i ett operativsystem eftersom där

finns all nödvändig information lagrad i kärnan. Ett annat problem är att autenticering kan vara tidskrävande och denna extra fördröjning vill man undvika i nätverk.

Auktorisering

Det finns tillfällen då olika användare och applikationer använder samma protokoll och skickar med likadan kod i sina paket. Resultatet blir att två versioner av samma kod finns i den aktiva noden. Med hjälp av auktorisering skulle de kunna få utnyttja samma kod och slippa onödiga extrapaket med kod.

6 Diskussion och slutsatser

Det finns två allvarliga problem med aktiva nät. Prestandaproblemet är störst längst nere på den fysiska protokollnivån, och på den nivån kommer aktiva nät förmodligen inte att vara en realistisk möjlighet. Det andra problemet är säkerheten. Problemet blir alltid påtagligt när man inför mobil kod. Med extra säkerhetsfunktioner blir prestandan lidande och man får ett motsatsförhållande mellan säkerhet och prestanda.

Å andra sidan kan aktiva nät öka säkerheten som helhet i och med att man i högre grad kan kontrollera och övervaka nätet.

En annan sak som man bör konstatera är att det behövs en gemensam och allmän modell för nätverksprogrammering om aktiva nät skall bli globalt användbara. Detta innebär att man måste skapa standardiserade primitiver och protokoll, på samma sätt som IP är standardiserat.

Det tidiga Internet var tillståndslöst. I princip innebär detta att ingen information om kommunikationsparametrar mellanlagras i noder. Detta gjorde nätet mycket skalbart samtidigt som det blev pålitligt och fick hög prestanda. Men moderna applikationer kräver i allt högre grad tillståndsfulla system. Genom att göra nätet aktivt flyttar man ut tillstånden till nätet och låter detta hantera en del av de tillstånd som applikationerna behöver.

Med aktiva nät ökar även komplexiteten i hela det system som bildas i nätet, vilket äventyrar säkerheten. I varje enskild nod kan det vara enkelt att styra behörigheter och skydda sig mot påverkan, men ett sammankopplat nät med aktiva noder med var sin egen policy blir snabbt svårhanterligt och det blir svårt att förutsäga konsekvenser av små förändringar.

Slutsatsen man kan dra är att man kan se tydliga tendenser till att nätet redan idag blir alltmer aktivt med särskilda programmerbara noder. Exempel på dessa är brandväggar och webproxies, som kan konfigureras och styras med snabb laddning av ny programkod. Framtida nät kommer förmodligen att innehålla någon typ av aktiv nod som fungerar som knutpunkt och har både brandväggsfunktionalitet, filtrering, mellanlagring, bandbreddsbegränsning och dynamisk laddning av nya protokoll.

Referenser

[1] A. S. Tanenbaum, *Computer Networks.* Prentice Hall, 3 ed., 1996. ISBN 0-13-349945-6.

[2] D. Wetherall, U. Legedza, and J. Guttag, "Introducing new internet services: Why and how," *IEEE Network Magazine*, pp. 12–19, July 1998.

[3] M. Persson, "Mobile code and safe execution," User Report FOA-R–98-00807-503–SE, FOA, April 1998.

[4] D. Wetherall, "Active network vision and reality: lessions form a capsule-based system," in *Symposium on Operating Systems Principles*, pp. 64–79, 1999.

[5] D. S. Alexander, M. Shaw, S. Nettles, and J. M. Smith, "Active bridging," in *SIGCOMM*, pp. 101–111, 1997.

[6] D. Alexander, M. Hicks, A. Keromytis, J. Moore, S. Nettles, and J. Smith, "A taxonomy of active code," 1999. IWAN.

[7] D. Wetherall, J. Guttag, and D. Tennenhouse, "Ants: A toolkit for building and dynamically deploying network protocols," in *IEEE OPENARCH'98*, April 1998.

[8] Y. Yemini and S. daSilva, "Towards programmable networks," 1996.

[9] "CANEs, composable active network elements." http://www.cc.gatech.edu/projects/canes/.

[10] S. Merugu, S. Bhattacharjee, Y. Chae, M. Sanders, K. Calvert, and E. Zegura, "Bowman and canes: Implementation of an active network." 37th annual Allerton Conference on Communication, Control and Computing, Monticello, Illinois, September 1999.

[11] "Smartpackets." http://www.ittc.ukans.edu/ kulkarn/ActiveNetworks.html.

[12] B. Schwartz, A. W. Jackson, W. T. Strayer, W. Zhou, R. D. Rockwell, and C. Patridge, "Smart packets for active networks," in *IEEE OPENARCH'99*, pp. 90–97, March 1999.

[13] D. Lindahl, "Mobil kod och datasäkerhet: Säkerhetsmekanismer i javas virtuella maskin," Technical Report FOA-R–99-01166-503–SE, FOA, 1999.

[14] "Kaffe - open source implementation of java virtual machine." http://www.kaffe.org.

[15] D. S. Alexander, W. A. Arbaugh, A. D. Keromytis, and J. M. Smith, "A secure active network environment architecture," *IEEE Network Magazine, special issue on Active and Controllable Networks*, vol. 12, no. 3, pp. 37–45, 1997.