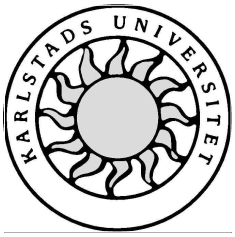


Datavetenskap

Patrik Jansson och Per Davidsson

Tidsåtgång för rundsändning i Internet-miljö

En jämförelse mellan ett skvallerprotokoll och ett protokoll för pålitligt rundsändning

---

D-uppsats

2003:05

Tidsåtgång för rundsändning i Internet-miljö

En jämförelse mellan ett skvallerprotokoll och ett protokoll för pålitlig rundsändning

Patrik Jansson och Per Davidsson

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en magisterexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Patrik Jansson

Per Davidsson

Godkänd, 2003-06-17

Handledare: Eivind Nordby

Examinator: Donald Ross

Sammanfattning

Vi har i tidskrifter och av personlig erfarenhet observerat en ökande trend för nätverksspel på Internet. Därför vill vi undersöka hur tiden för rundsändning i distribuerade protokoll uppför sig på Internet. Den egenskap av protokollen vi vill undersöka är den tid det tar för ett protokoll att distribuera ett meddelande från en process till alla andra processer, som ingår i ett nätverk av processer. Denna tid är en viktig aspekt i nätverksspel.

Vi valde två distribuerade protokoll, pålitlig rundsändning (Reliable Broadcast) samt ett skvallerprotokoll (Gossip), vilka båda efter litteraturstudier tycktes vara de mest lämpade. Genom att simulera en modell av Internet, som tar hänsyn till förluster och fördröjningar, utförde vi två olika tester. Det ena testet syftar till att undersöka om det finns någon klar skillnad mellan de två protokollen då förluster och fördröjningar var satta till medelvärden som observerats på Internet. Det andra testet syftar till att undersöka hur protokollen beter sig vid olika värden för förluster och fördröjningar som kan finnas på Internet. Under arbetets gång fann vi ett sätt att beräkna en graf, där avståndet mellan två godtyckliga noder kan hållas lågt, och på så vis minska tiden för rundsändning.

Båda testerna undersökte tre nätstorlekar. Under alla tre nätstorlekar var skvallerprotokollet snabbare än protokollet för pålitlig rundsändning med avseende på den tid vi ville mäta. Inga tydliga trender kunde observeras i testet av flera olika nivåer på förluster och fördröjningar.

Time consumption for broadcasting in an Internet environment

A Comparison between a Gossip protocol and a Reliable Broadcast protocol

Abstract

We have, in magazines and from personal experience, observed an increasing trend toward network games on the Internet. We would like to investigate how the time consumption for broadcasting using different protocols will behave on the Internet. The one property of the protocols we would like to investigate is the time it takes to send one message from one node to all the other nodes, since this time is of particular importance in network games.

We have selected two protocols, one out of the Gossip family and one from the Reliable Broadcast family of protocols. Through simulations of the characteristics of the Internet, based upon loss and latency, we conducted two tests. The first test was to establish if there exists any difference in the mean value for the time for broadcasting and the second test to investigate if loss and latency could influence the established difference. During the development of this thesis we have found a way to calculate a graph, where the distances between two arbitrary nodes are low.

The first test was based upon three network sizes, and in all of these three cases, time consumption was to the gossip protocols advantage, with respect to the amount of time elapsed for broadcast. No clear trends were observed in the second test. As a result, we imply that loss and latency may not influence the difference in time of broadcasting for the to protocols.

Innehållsförteckning

Figurförteckning

Tabellförteckning

Formelförteckning

1 Inledning

Denna uppsats är gjord på instutitionen för datavetenskap vid Karlstads universitet. Vi som skrev denna uppsats har själva valt ut och avgränsat område och omfång. Uppsatsen är uppdelad i fyra delar. Den första delen består av de två första kapitlen som ger en introduktion till uppsatsen och några viktiga termer och ordval som förekommer i den här uppsatsen. Vi vill studera tidsåtgången för rundsänding av meddelanden i ett logiskt nätverk, eftersom vi anser det vara en viktig studie i relation till nätverksspel. Den andra delen ger oss först en överblick över olika typer av protokoll som finns för rundsändning och sen beskrivs två på djupet i kapitel 3 och 4. Den tredje delen utgörs av två experiment. Ett huvudexperiment som undersöker om det finns någon skillnad i tidsåtgången för rundsändning mellan två olika protokoll, och ett biexperiment som undersöker om förluster och fördröjningar kan inverka på skillnaden i tidsåtgång. Den fjärde och sista delen av uppsatsen försöker dra slutsatser som relaterar till vår bakgrund.

Detta inledande kapitel börjar med en överblick över vår bakgrund i nätverksspel och varför vi intresserar oss för att genomföra experimenten i denna uppsats. Sen beskrivs uppsatsens syfte, som grundar sig på våra intressen för nätverksspel. Ur syftet tar vi sedan fram vilket mål vi har med uppsatsen. Det sista avsnittet i detta kapitel tar upp uppsatsens struktur.

1.1 Vår bakgrund

Författarna till denna uppsats är Patrik Jansson och Per Davidsson. Vi kommer här att ge en kort sammanfattning av våra intressen som lett fram till denna uppsats. Båda författarna har ett stort intresse för nätverksspel.

1.1.1 Nätverksspel

Spel har alltid varit en av våra hobbyer, och då speciellt bilspel och rollspel på dator eller på konsoler för tv-spel. Bilspel är spel då spelaren styr en bil på en dator och tävlar mot flera andra datorstyrda bilar. Rollspel är spel där karaktärer påverkar varandra i olika situationer. I båda sorterna spel styr datorn flera objekt, d v s andra bilar eller karaktärer. Ett problem som upplevts av oss båda, är att datorstyrda objekt lätt blir förutsägbara. Efter några spelomgångar

vet man ungefär vilka drag som det datorstyrda objektet skall göra och vilka situationer det datorstyrda objektet har svårt för. När ett spel blir förutsägbart kan vi som spelare lätt tappa intresset för att spela vidare. I nätverksspel har de datorstyrda objekten ersatts av andra mänskliga spelare, och vi människor är inte lika förutsägbara. Nätverksspel har blivit mycket populära och har en stor del av spelmarknaden [3]. Under 2003 lanserade de stora tillverkarna av konsoler för tv-spel sina nätverksplattformar, som är avsedda att fungera på Internet eller i ett nätverk. I spel till dessa plattformar är det människor som styr de flesta av objekten och detta ger en nästan oändlig variation i spelen.

Ett vanligt sätt att styra dessa nätverksspel, är att speltillverkarna tillhandahåller en centralt placerad server, och tillåter att alla spelarna anslutaer sig till den servern. Denna typ av lösning av nätets struktur kallas ofta klient/server-system. Om en server i ett sånt system faller bort, permanent eller tillfälligt, så kommer nätverksspelet att fungera sämre eller sluta fungera helt, med följd att många spelare kan tappa intresset att spela vidare. Ett annat problem är den begränsade kapacitet som varje server har. En av tillverkarna uppgav att runt 3000 spelare skulle kunna rymmas på en server samtidigt. En server kan troligen hålla flera omgångar av ett spel, och kanske även flera olika spel. Om det t ex finns 30 000 nätverksspelare vid ett tillfälle av hög belastning, kommer speltillverkarna behöva minst 10 servrar för att tillgodose spelarnas behov. Troligen kommer företagen att dimensionera antalet servrar för ett betydligt större antal spelare. Detta kan medföra en mycket stor kostnad. Om speltillverkarna istället skulle välja en serverlös nätverksarkitektur skulle spelarna bli fria från servrarnas vara eller inte vara.

Ett problem som uppstår i samband med alla nätverksspel är, när uppdateringar av spelarnas status behöver spridas. I ett spel där reaktionsförmågan är av vikt bör denna uppdatering ske snabbare än den tid en normal människa hinner reagera. Denna tid är ungefär en femtedels sekund [2]. I ett centraliserat nät behöver först alla spelare rapportera in sin egen status till den centralt placerade servern. Servern kan sedan sammanställa informationen och skicka ut all statusinformation till varje spelare. Då all information passerar servern kan statusinformation från olika spelare slås ihop till ett meddelande. Detta ger då ett lägre antal meddelanden än om varje statusinformation skall skickas till alla spelare. Enkelheten och snabbheten i denna lösning är en anledning till att den är så populär. När en uppdatering sker, i ett serverlöst nät, kommer väldigt mycket data att behöva skickas mellan deltagarna. Om det finns 100 spelare i ett spel måste varje spelare skicka sin statusinformation till alla andra, vilket betyder att $99^2 = 9801$ meddelanden behöver skickas. Det här kan ta mycket längre tid än den femtedels sekund vi människor tolererar. Dessutom behöver det gå snabbt för ett

meddelande att nå från en deltagare ut till alla andra. Om det går att finna en lösning där information kan spridas snabbt samt antalet meddelanden kan hållas lågt, så kan serverlösa nät bli ett bra alternativ till den serverberoende lösningen. Hänsyn behöver tas så att inte användare på en modemuppkoppling blir överbelastade med meddelanden.

1.2 Syfte med arbetet

1.2.1 Nätverksspel

Syftet med den här uppsatsen är att studera hur serverlösa nät fungerar för statusdistribution i nätverksspel. Genom att få en förståelse för hur snabbt olika protokoll för rundsändning kan sprida information, hoppas vi kunna ge ett alternativ till den klient/server-nätstruktur som det är vanligt att speltillverkarna använder idag. Protokollet skall klara av förutsättningar som existerar på Internet, exempelvis att enstaka statusuppdateringar eller spelare försvinner, utan att protokollens prestanda försämras avsevärt, men även att pålitligheten att ett meddelande når alla spelare inte försämras. Vi borde även försöka ta hänsyn till att spelare med till exempel modeluppkopplingar inte blir överbelastade. Den centrala frågan är ändå tidsåtgången för rundsändning, och hur denna kan påverkas av valet av protokoll för serverlösa nät.

1.3 Frågeställningar

Till en början undersökte vi vilka protokoll som fanns tillgängliga för att sprida ett meddelande i ett serverlöst nät. Tidsåtgången för att sprida ett meddelande med de protokoll som vi fann var den mest intressanta egenskapen att studera för vårt område. De protokollen vi vill undersöka behöver dock ta hänsyn till pålitlighet så att ett meddelande når alla medlemmar i en Internetmiljö. Protokollen skall inte generera meddelanden i onödan. För många meddelanden kan riskera att överbelasta en modemuppkoppling. Hur storleken på näten påverkade protokollen var en aspekt vi ville undersöka, då olika nätverksspel har olika antal spelare.

1.4 Mål

Vi vill ge en ytlig överblick över några av de protokoll vi träffat på under litteratursökningen och väljer ut ett par av dessa, med hänsyn till att efterlikna ett test som tidigare har utförts av Lin et al. [9] för att öka validiteten av vår testmiljö. De båda protokollen var också de som

efter litteraturstudierna verkade vara de protokoll som är snabbast på rundsändning av de protokoll vi fann. Dessa två heter Reliable Broadcast och Gossip. Vi har översatt dessa ord och kallat protokollen för ”pålitlig rundsändning” respektive ”skvallerprotokollet”. Vi vill optimera dessa protokoll så att tidsåtgången blir så liten som möjligt i en Internetmiljö och uppmäta denna tidsåtgång.

Internetmiljön skall bygga på förluster av data vid opålitliga sändningar och fördröjningar av paketen. Vi har två mål med vår undersökning. Vi vill jämföra de båda protokollens tidsåtgång för att avgöra vilket protokoll som sprider ett meddelande snabbast i en Internetmiljö. Internetmiljön skall ha värden för förluster och fördröjningar, som är medelvärdet för förluster och fördröjningar på Internet i skrivandet stund, vid olika storlekar på nätet. Vi vill också undersöka hur mycket olika mängder av förluster och olika långa fördröjningar inverkar på tidsåtgången. Alla dessa värden skall ligga inom ramen för rimliga värden som kan existera på Internet idag.

1.5 Referenstest

Genom att försöka upprepa ett tidigare test och få samma resultat kan validiteten för vår testmiljö öka. Vi ville därför upprepa ett test som gjorts av Lin et al [9]. Lin et al har undersökt pålitligheten för skvallerprotokollet och protokollet för pålitlig rundsändning i ett lokalt nätverk. Lin et al mätte tiden för rundsändning för respektive protokoll och nådde resultatet, att protokollet för pålitlig rundsändning var snabbare än skvallerprotokollet. Vi kommer i fortsättningen att kalla detta test för ”referenstestet”.

1.6 Resultat

I vår testmiljö fanns flera skillnader jämfört med den testmiljö som Lin et al. [9] använde. Vi var bland annat mer intresserade av att testa protokollen i en Internet-miljö. Då vi inte hade tid att utföra alla tester uteblev det test som skulle bekräfta Lin et als resultat. I våra tester kunde vi dra slutsatser om hur protokollen skulle bete sig i stora nät med 68 medlemmar eller fler. Det visade sig då att skvallerprotokollet var snabbare än protokollet för pålitlig rundsändning.

1.7 Uppsatsens struktur

Uppsatsen är uppdelad i fyra delar. De fyra delarna beskrivs i de fyra styckena i detta avsnitt. I kapitel 2 beskrivs de termer som kommer att användas i denna uppsats. En ordlista över termer återfinns i bilaga A. De avgränsningar som ledde fram till denna uppsats finns återgivna i bilaga B.

För att ge en känsla för vilka protokoll som är intressanta i nätverk, som Internet, där sändnings- och mottagningsförluster är vanliga, går vi, i kapitel 3, igenom ett antal protokoll som är designade för att klara av enkla fel. En övergripande beskrivning av protokollen ges där. I kapitel 4 väljer vi ut två protokoll av de beskrivna protokollen och motiverar varför de utvalda protokollen blev valda. Dessa två protokoll som vi översatt till ”protokollet för pålitlig rundsändning” (Reliable Broadcast) och ”skvallerprotokollet” (Gossip) beskrivs mer ingående. För att använda protokollet för ”pålitlig rundsändning” behövs en graf som beskriver vilka medlemmar en medlem känner till, och därmed kan kommunicera med. En algoritm ges som kan skapa en sådan graf. För att förbättra tiden för rundsändning beskrivs en graf som protokollet för ”pålitlig rundsändning” kan använda.

Utifrån de valda protokollen skapar vi, i kapitel 5, de frågeställningar som ligger till grund för testerna. Frågeställningarna bygger på den mer generella frågeställningen i avsnitt 1.3. För att besvara frågeställningarna bestämdes att två olika tester skulle utföras. Ett test skall undersöka trenderna inom ett givet intervall för antalet noder, storleken på paketförluster och fördröjningen i nätverket. Det andra testet skall vara mer djupgående för att få en stor säkerhet inom det testade området. Data som behövs för att svara på frågeställningarna beskrivs och sedan ingående de metoder som använts för att analysera resultatet. Kapitel 6 förklarar hur protokollen är konfigurerade i vårt test, då de kan ställas in olika för att ge olika hög pålitlighet. Vi presenterar de värden på antal medlemmar, fördröjningar och paketförluster vi valt att testa och motiverar valet av dem. I kapitel 7 beskrivs hur programmen som skall utföra testerna har designats och implementerats. Valet av programspråk motiveras. En djupare beskrivning av implementationen finns i bilaga C. Bilaga D ger en överblick över Internets protokollstack. I Kapitel 8 redovisas resultaten från testerna. Diagram används i de fall där det är praktiskt men i några fall presenteras resultatet i form av en tabell. Exempel av stickprov för testerna återfinns i bilaga G och E. I kapitel 9 analyseras resultatet för de olika testerna med de metoder som beskrivits i kapitel 1. Validiteten av experimenten diskuteras därefter.

Slutsatserna om användbarheten av protokollen i nätverksspel görs i kapitel 10. Först sammanfattas dock slutsatserna från analysen. En presentation av de intressanta aspekter som arbetet givit ges slutligen.

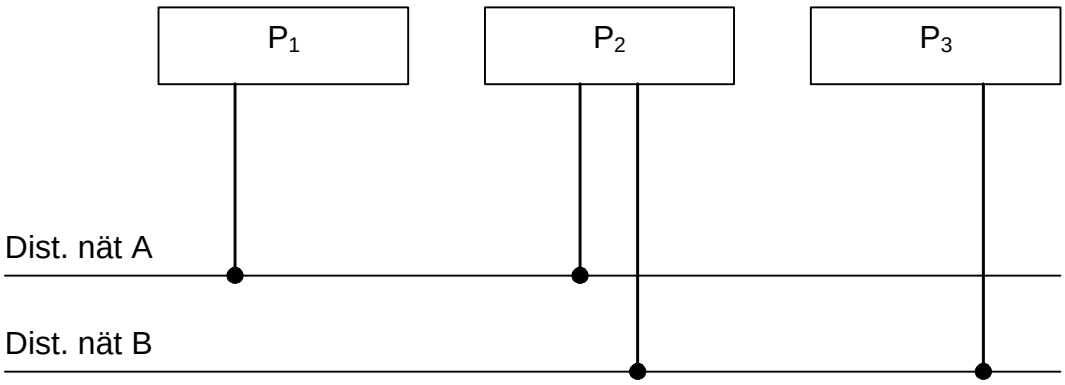
2 Termer, riktlinjer och ordval

Detta kapitel tar upp de termer som kommer att användas i kommande kapitel. De termer vi tar upp inkluderar medlemmar, serverbaserade och serverlösa nät, meddelanden, relationen mellan antalet hopp och tidsåtgången för rundsändning, pålitlighetens vikt för våra syften och några intressanta egenskaper hos Internet. Detta kapitel avslutar första delen av uppsatsen. I nästa del av uppsatsen kommer vi att beskriva några protokoll för rundsändning och sen välja ut de två intressantaste protokollen för en närmare studie i delen som följer.

2.1 Medlemmar, medlemslistor och rundsändning

Genomgående i den här uppsatsen kommer termen medlemmar att användas. En medlem ingår i ett logiskt nätverk. En medlem kan skicka ett meddelande till andra medlemmar som medlemmen känner till. Vi använder ordet medlem hellre än ordet process, eftersom en process kan vara medlem i flera nätverk samtidigt vilket inte en medlem kan. Detta visas i Figur 2-1, där punkterna på linjerna representerar att en del av en process är medlem i ett logiskt nät. En medlem kan endast skicka ett meddelande till medlemmar som medlemmen känner till. För att meddelandet skall nå alla medlemmar i nätet som medlemmen inte känner till och för att en hög pålitlighet skall uppnås, används ett protokoll för rundsändning. Rundsändning är en översättning av det engelska ordet broadcast.

En medlem använder sig av en medlemslista, som innehåller de medlemmar som medlemmen känner till. Denna lista behöver inte innehålla alla medlemmar som ingår i nätet, men samtliga medlemmars medlemslistor tillsammans kommer att innehålla nätets samtliga medlemmar. Om en medlem känner till alla andra medlemmar så har medlemmen global kunskap. Med partiell kunskap menar vi att en medlem endast känner till några andra medlemmar.



Figur 2-1: P_1 är medlem i nät A, P_2 i A och B samt P_3 är medlem i B

2.2 Serverbaserade, Klient/Server nät

I ett klient/server-nät finns en server, och många klienter. Klienterna kopplar upp sig mot servern. Servern kan på det sättet enkelt låta klienterna indirekt ha kontakt med varandra, och hålla reda på alla klienter. I detta fall kan klienterna också ses som medlemmar, eftersom de kan skicka ett meddelande till alla andra medlemmar i nätet, fast de inte har någon medlemslista. Medlemslistan i det här fallet ligger istället på servern. En server kan vidare slå samman information från flera klienter i en och samma sändning.

2.3 Serverlöst nät

I ett serverlöst nät skickar medlemmarna meddelandena mellan varandra och ej via en server. I vissa serverlösa nät kan en server ändå användas, men då för att sköta medlemshanteringen. Dock är det fortfarande så att själva skickandet av meddelanden sker utan hjälp av en server.

2.4 Meddelande, information, statusuppdateringar, resultat och meddelandelistor

Meddelanden skickas mellan medlemmarna, och innehåller någon form av information som behöver spridas. Denna information kan t ex vara en statusuppdatering från en spelare, resultat från en beräkning, eller något annat. En meddelandelista kan skickas med ett meddelande, och innehåller en lista över de medlemmar som meddelandet passerat. En sådan lista är användbar då den ger information om vilka medlemmar som har fått meddelandet.

2.5 Pålitlighet, antal meddelanden och tid för rundsändning

Pålitligheten hos ett protokoll för rundsändning är enligt Lin et al [9] sannolikheten att ett meddelande levereras av alla felfria medlemmar. En felfri medlem kan ta emot ett meddelande, vidarebefordra meddelandet och leverera meddelandet. Att leverera meddelandet innebär att medlemmen lämnar av meddelandet till processen som använder medlemmen.

Genom att skicka fler meddelanden kan ofta pålitligheten ökas. Om det är så att det är ungefär samma sannolikhet att ett meddelande ska försvinna mellan två godtyckliga medlemmar, betyder det att sannolikheten att ett meddelande försvinner ökar med antalet hopp det behöver göra. Det har beskrivits och bevisats av Lin et al [9]. Detta är ytterligare en anledning till att studera antalet hopp ett meddelande gör.

Det är viktigt att alla medlemmar får ett meddelande då medlemmarna är karaktärer i ett spel. Annars kan det bli konstiga effekter som att en karaktär återuppväcks från de döda då ett meddelande om förflyttning av karaktären ej kommit fram. Då vi är intresserade av att ett meddelande når alla medlemmar kommer därför pålitligheten för ett protokoll, i de fall den informationen finns tillgänglig, att rapporteras. Som vi tidigare har konstaterat är det av vikt att antalet meddelanden kan hållas nere. Vi kommer därför i de fall det går att rapportera antalet meddelanden ett protokoll genererar samt försöka att minimera antalet meddelanden. Tiden tills alla medlemmar har nåtts av ett meddelande är dock vårt huvudsakliga fokus.

2.6 Några intressanta Internet-egenskaper

Vi beskriver nedan ett antal av Internets egenskaper. Fördröjningar och förluster är de egenskaper av Internet som vi finner mest intressanta att studera.

2.6.1 Fördröjning

Ett meddelande kommer aldrig att nå fram till sin tänkta destination ögonblickligen, utan det finns en faktiskt mätbar fördröjning. Denna har uppmätts att vara från 40 ms till ungefär 200 ms, med ett medelvärde på ungefär 80 ms [7][12].

2.6.2 Förluster

Paket eller meddelanden som sänds på Internet kan gå förlorade. Sannolikheten för att detta ska inträffa är ganska låg, men den är mätbar. Flera försök har gjorts för att försöka mäta hur många paket som faktiskt går förlorade[7][12]. Dessa mätningar har visat på siffran 1,2 %. Hur förlusterna varierar har inte uppgetts. Se bilaga E för ytterligare information.

2.6.3 Statiskt och homogent Internet

Vi kommer senare att använda termen statiskt och homogent Internet. Med statiskt Internet menar vi ett Internet där sannolikheten till förluster och storleken på fördröjningar inte ändras med tiden. Med ett homogent Internet menar vi att ett nätverk där sannolikheten till förluster och storleken på fördröjningar mellan medlemmar är lika mellan alla medlemmar. I båda fallen är värdena på förluster och fördröjningar satta till värden som representerar medelvärden, som är uppmätta på Internet. Hur det statiska och det homogena beror av varandra har vi inte kunskap om.

2.6.4 Kort om fördröjningar på Internet

För att sända ett meddelande från en nod till en annan involveras följande förseningar och tider [6]:

- Bearbetningsfördröjning, då paketets huvuden ska undersöka, d_{proc}
- Köfördröjning, d_{queue}
- Transmissionsfördröjning, då ett paket skickas, d_{trans}
- Propageringsfördröjning, d_{prop}

Om vi slår samman dessa tider får vi alltså Formel 2-1 som ger oss den totala tiden för att skicka ett meddelande från en medlem till en annan.

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Formel 2-1: Tiden ett meddelande tar från en nod till en annan

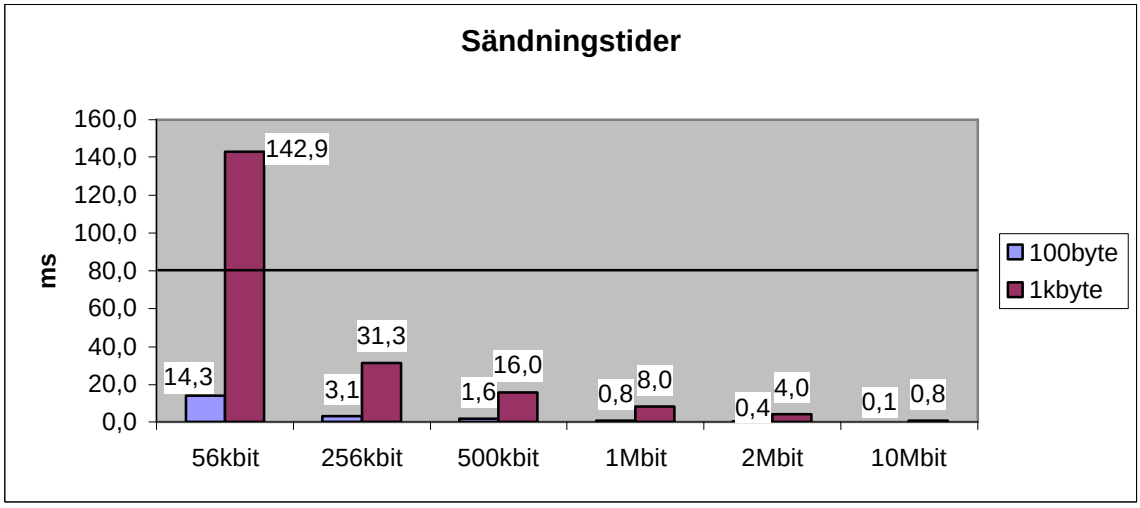
På ett icke stockat Internet, kommer d_{proc} och d_{queue} att vara mycket små i jämförelse med d_{trans} och d_{prop}, så små att de bara uppstår som brus i mätningarna enligt Kurose [6]. d_{prop} beror på avståndet mellan de medlemmar som kommunicerar. d_{trans} beror på meddelandets storlek samt bandbredden som medlemman förfogar över. De två intressanta fördröjningarna blir då sändningstiden samt utbredningstiden. Utbredningstiden är den tid som meddelandet återfinns i nätet.

Om vi studerar de olika fördröjningarna lite så ser vi att tre av fördröjningarna, d_{proc}, d_{queue} och d_{prop}, beror på ”Internet” och den fjärde fördröjningen, d_{trans}, beror på själva uppkopplingen till Internet. Låt oss förenkla modellen och säga att d_{nodal} består av $d_{Internet}$ och $d_{uppkoppling}$. I de fall då d_{trans} är mycket mindre än $d_{Internet}$ kommer resonemanget om relationen mellan avstånd och tid i 2.7 att gälla, eftersom alla meddelanden kommer iväg ungefär samtidigt. Däremot om d_{trans} blir i ungefär lika stor som eller större än $d_{Internet}$ så kommer inte alla meddelanden att gå iväg samtidigt, och parallelliteten försvinner, och resonemanget i 2.7

fungerar inte längre. Antalet hopp som ett meddelande gör blir beroende också på i vilken ordning som meddelandena skickas. För att förenkla experimentet valde vi att genomföra experiment där parallelliteten finns.

2.6.5 Sändningstid i relation till fördröjning

Från 2.6.1 vet vi att fördröjningen på Internet, d v s $d_{Internet}$ är ungefär 80 ms stor. Hur väl kommer resonemanget i 2.7 om parallellitet att fungera med de olika tekniker som finns tillgängliga idag för uppkoppling till Internet? Vi studerar 56kbit modem, ADSL vid 2mbit nerströms och 256kbit uppströms och 10mbit Ethernet-koppling. Vidare studerar vi hur snabbt ett meddelande på ungefär 100 byte och ett på ungefär 1000 byte totalt kan skickas till Internet. I Tabell 2-1 är alla sändningstider samlade för de olika paketstorlekarna och bandbredderna, och linjen markerar den ungefärliga fördröjningen på Internet, d v s $d_{internet}$.

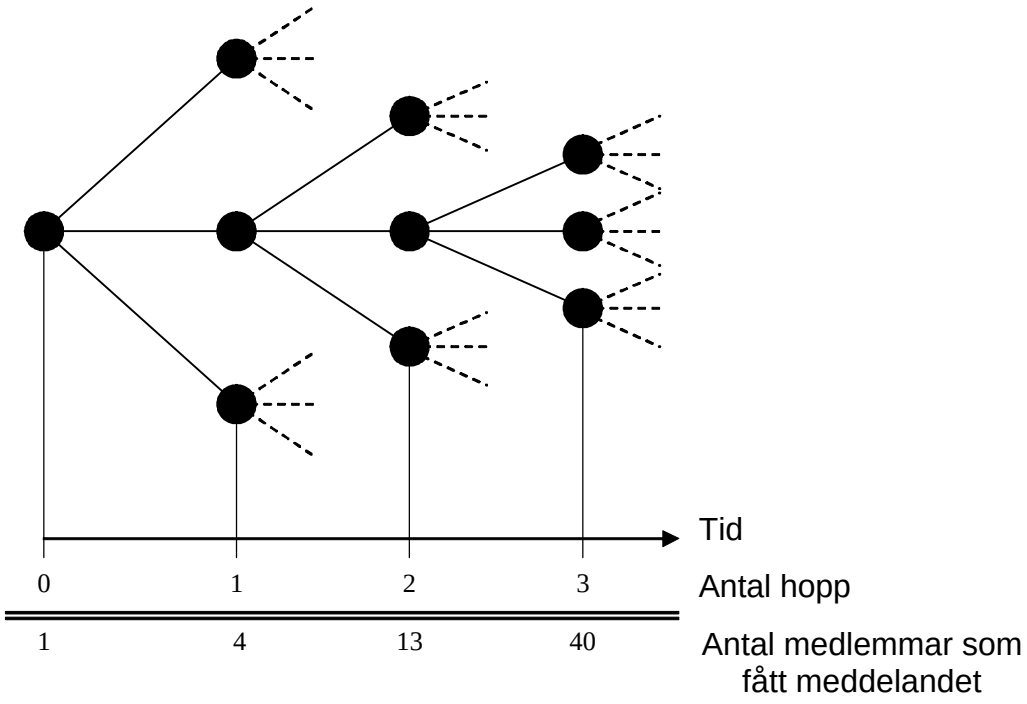


Tabell 2-1: Sändningstider i relation till fördröjningen på Internet

Om vi studerar Tabell 2-1 ser vi att om vi endast skickar små meddelanden så kommer paralleliteten att bevaras, ganska väl. På ett 56kbit modem hinner vi skicka ungefär 5 meddelanden på 100byte (inklusive alla huvuden) innan det första meddelandet hinner komma fram till sin destination. Troligen betyder det här att parallelliteten skulle fungera vid små meddelanden även på en så långsam enhet som ett modem, men med stora meddelanden behöver vi troligen minst en 500kbit:s koppling, för att sändningen ska uppfattas som parallell.

2.7 Relationen avstånd mätt i antalet hopp och i tid

Då ett meddelande skickats från en medlem och nästa medlem mottagit meddelandet säger vi att meddelandet gjort ett hopp. Ett meddelande kan göra flera hopp för att nå en medlem, dvs meddelandet passerar flera medlemmar på vägen. Om ett meddelande gjort flera hopp på vägen till en medlem blir tiden tills medlemmen mottagit meddelandet minst summan av tiden för varje hopp. Enligt Kurose [6] är tiden som ett meddelande passerar mellan medlemmar den faktor som tar mest tid. Därför anser vi att det finns anledning att studera antalet hopp ett meddelande gör på sin väg till en medlem samt att försöka hålla antalet hopp så lågt som möjligt. Figur 2-2 illustrerar antalet hopp som gjorts innan ett visst antal medlemmar fått ett meddelande, då varje medlem skickar ett meddelande vidare till tre andra medlemmar.



Figur 2-2: Vad vi menar med antal hopp, i samband med tidsåtgång

2.7.1 Antal hopp

Vid ett statiskt och homogent Internet där sändningstiderna är försumbara uppför sig ett rundsändningsprotokoll som om meddelanden som skickas från en medlem direkt efter varandra skickas samtidigt. Eftersom nätet är statiskt och homogent kommer meddelandena att skickas i vågor, om det är tillräckligt hög bandbredd på uppkopplingen till nätet. Den tid det tar för ett meddelande att nå någon medlem kommer då att bli antalet hopp som meddelandet gör gånger tiden det tar för ett meddelande att färdas mellan två medlemmar.

2.8 Fel som kan uppstå under en sändning

Det finns många olika sorters fel som kan uppstå under en sändning. Med termen fel menar vi i denna uppsats då medlemmar eller då enskilda meddelanden faller bort under en rundsändning. Vi tar alltså inte hänsyn till så kallade bysantinska fel, det vill säga fel som gör att meddelanden kan komma fram, men att meddelandet också kan vara förändrat.

3 Allmänt om distribuerade protokoll

Den första delen av uppsatsen har presenterat inriktning och avgränsningar för uppsatsen, samt beskrivit de termer som kommer att användas i denna uppsats. Den här delen av uppsatsen inkluderar även nästa kapitel, och kommer att ge en genomgång av flera protokoll för rundsändning, samt en närmare studie av två av protokollen. I nästa del av uppsatsen, som består av kapitel 5 till kapitel 9 beskrivs frågeställningarna samt de experiment som utförts för att besvara frågeställningarna.

I detta kapitel redovisar vi några av de protokoll vi hittat under vår litteraturstudie. Vi börjar kapitlet med en översikt över dessa protokoll och visar på några av deras egenskaper. Efter det ger vi en lite djupare diskussion om var och ett av dessa protokoll. I nästa kapitel väljer vi ut två av protokollen, som vi där beskriver i detalj. De protokoll som väljs är protokollet för pålitlig rundsändning och skvallerprotokollet. En djup förståelse av protokollen är nödvändig för att kunna förstå vad som ligger till grund för protokollens skillnader i en jämförelse mellan protokollen. Alla protokoll i det här kapitlet är ämnade att användas i ett serverlöst nät.

3.1 Överblick över olika kategorier protokoll

För att ge en snabb överblick över de protokoll som vi fann under våra inledande litteraturstudier presenteras dessa protokoll samt deras huvudsakliga egenskaper. Vilka protokoll som har vilka egenskaper sammanfattas i en tabell. Det finns två huvudtyper av protokoll som vi har studerat, deterministiska och slumpmässiga. Det som skiljer de båda typerna åt är sättet en medlem väljer ut till vilka andra medlemmar den ska skicka vidare ett meddelande. Bland de deterministiska protokollen redovisar vi ett protokoll kallat protokollet för pålitlig rundsändning (Reliable Boardcast), som Mullander skrivit om [10]. Bland de slumpmässiga protokollen har vi undersökt en grupp av protokoll som kallas för ”epidemiprotokoll”. Ordet epidemiprotokoll används då de bygger på läran om epidemier, alltså hur en sjukdom sprids. Av dessa protokoll beskriver vi ”icke-oordningsprotokollet” (Anti Entropy) [1], ”ryktesspridningsprotokollet” (Rumor mongering) [1] ,

”skvallerprotokollet” (Gossip) [9], ”riktat skvaller” (Directional Gossip) [8] och till sist ett ”resurssnålt sannolikhetsbaserat protokoll” (Lightweight Probablistic Broadcast) [11].

De medlemmar som en medlem väljer att skicka ett meddelande till väljs på ett slumpmässigt sätt om ett epidemiskt protokoll används. En medlem som använder protokollet för pålitlig rundsändning har däremot några förutbestämda grannar som ett meddelande skall skickas till.

Det som skiljer dessa sex protokoll åt är bland annat hur många av alla medlemmarna i ett nät varje medlem känner till, när ett meddelande skickas vidare och vad som bestämmer hur många gånger detta sker. Tabell 3-1 sammanfattar egenskaperna av de studerade protokollen. Dessa egenskaper beskrivs här även i text. De två protokoll som valts ut för testerna är markerade med en asterix.

Det resurssnåla sannolikhetsbaserade protokollet och protokollet för pålitlig rundsändning har bara kunskap om ett förutbestämt antal andra medlemmar, och riktat skvaller lär känna medlemmar efterhand som meddelanden inkommer. I de tre andra protokollen känner alla medlemmar till alla andra medlemmar.

Icke-oordningsprotokollet och det resurssnåla sannolikhetsbaserade protokollet försöker skicka sin information vidare vid regelbundna tillfällen även om ingen ny information skapats, medan de fyra andra protokollen sprider ett meddelande vidare när de precis har fått det.

Ryktesspridningsprotokollet använder en sannolikhetsfaktor för att bestämma hur många gånger ett meddelande skickas vidare och både skvallerprotokollet, riktat skvaller samt det resurssnåla sannolikhetsbaserade protokollet skickar vidare ett meddelande ett bestämt antal gånger. Icke-oordningsprotokollet sprider sin information i oändlighet. Protokollet för pålitlig rundsändning skickar ett meddelande vidare endast en gång.

	Val av grannar	Kännedom av andra medlemmar	Tidpunkt för vidarebefodran	Antal vidarebefodran
Pålitlig rundsändning*	Deterministiskt	Partiell	Direkt	En gång
Icke-oordning	Slumpmässigt	Total	Regelbundet	I oändlighet
Ryktesspridning	Slumpmässigt	Total	Direkt	Sannolikhetsbaserat
Skvaller*	Slumpmässigt	Total	Direkt	Förutbestämt
Riktat skvaller	Slumpmässigt	Varierande	Direkt	Förutbestämt
Resurssnålt sannolikhets-baserat	Slumpmässigt	Partiell	Regelbundet	Förutbestämt

* Protokoll vi kommer att jämföra mot varandra

Tabell 3-1: Översikt över några egenskaper på redovisade protokoll

3.2 Protokollet för pålitlig rundsändning (Reliable Broadcast)

Protokollet för pålitlig rundsändning [1] är enkelt att implementera och kan konfigureras så att den ger en hundraprocentig pålitlighet då färre än ett förutbestämt antal meddelanden eller medlemmar faller bort under en sändning. Första gången då en medlem tar emot ett meddelande skickas det vidare till de andra medlemmar som den känner till, förutom den medlem som meddelandet kom ifrån. Efterföljande gånger då medlemmen tar emot samma meddelande, skickar den inte meddelandet vidare. Antalet medlemmar eller meddelanden som kan falla bort under en sändning utan att pålitligheten skall minska bestäms av antalet medlemmar man låter varje medlem känna till. Genom att låta medlemmarna ha total kunskap, blir pålitligheten hög, men nätverksbelastningen blir också väldigt högt. Ju högre nätverksbelastningen är, desto fler meddelanden kommer att gå förlorade, och risken att inte alla medlemmar kommer att få ett meddelande vid en sändning blir då större.

Ett sätt att dra ned på antalet meddelanden för att minska nätverksbelastningen är att minska antalet andra medlemmar som varje medlem känner till. Vilka andra medlemmar som en medlem känner till går inte att välja hur som helst eftersom detta i värsta fall kan ge ett partitionerat nät. I ett partitionerat nät kan inte en sändning nå ut till alla medlemmar. Det behövs därför ett bra sätt att välja vilka andra medlemmar varje medlem känner till. Frank Harary [9] skapade en algoritm, som kan användas för att göra just detta. Denna algoritm har beskrivits av Lin et al [9]. Genom att använda denna algoritm, och att antalet medlemmar och meddelanden som försvinner under en sändning är mindre än antalet medlemmar varje medlem känner till, kan den höga pålitligheten uppnås.

3.3 Epidemiprotokoll

En tidig teknik som användes för att hålla duplicerade databaser synkroniserade var så kallad direktpost. Direktpost innebär att ny information skickas direkt till alla andra databaser, och att inga omsändningar görs. Om den nya informationen på väg till någon av databaserna försvann uppstod problemet att databaserna inte längre var konsistenta. För att lösa det här problemet testades två protokoll [1] som bygger på studier om epidemier. De två epidemiprotokoll som testades var icke-oordningsprotokollet och ryktesspridningsprotokollet.

Epidemiprotokollens funktion bygger på matematiska modeller om hur en sjukdom sprids vid en epidemi [1]. Vid epidemier kan dessa modeller användas för att begränsa spridningen av en sjukdom, men i epidemiprotokoll försöker de användas för att maximera spridningen av ett meddelande. Dessa epidemiprotokoll användes först för att uppdatera duplicerade

databaser [1], men det allmänna intresset för dessa algoritmer har ökat, bland annat genom den ökande populariteten för serverlösa lösningar.

Senare studier har föreslagit variationer och utbyggnader av de epidemiska protokoll som beskrivits av Demers [1]. Demers protokoll samt de variationer och utbyggnader av epidemiska protokoll som vi funnit i våra litteraturstudier beskrivs nedan.

3.3.1 Icke-oordning (Anti entropy)

Vid användning av icke-oordningsprotokollet väljer varje medlem, vid regelbundna tillfällen, slumpmässigt ut en annan medlem att synkronisera sig med. Efter synkroniseringen har båda medlemmarna i bästa fall samma information, oordningen mellan databaserna är åtminstone något mindre. Det finns tre sätt en uppdatering i icke-oordningsprotokollet kan ske. Första sättet går ut på att först skickas en resumé av ena databasen till den andra, och om den andra databasen har nyare uppgifter än den första databasen så skickas dessa uppgifter tillbaka. Det andra sättet bygger på att den första databasen begär en resumé av den andra databasen, och sen skickar den första databasen alla nyare uppgifter tillbaka till den andra databasen. Det tredje och sista sättet är en kombination av dessa två sätt. Först skickas en resumé från den ena databasen till den andra, den andra svarar med nya uppgifter och sin egen resumé, och till slut skickar den första databasen alla sina nya uppgifter tillbaka till den andra databasen. Dessa tre sätt kallas för att hämta (pull), skicka (push) och hämta/skicka (pull/push).

Epidemiska teorier visar att icke-oordningsprotokollet kommer att sprida ett meddelande till alla nätets medlemmar. Dock fungerar de tre olika sätten olika bra då ett fåtal noder återstår som inte fått en ny uppgift. Enligt Demers et al [1] är sättet att skicka och hämta/skicka ny information snabbare än sättet med att bara hämta ny information.

3.3.2 Ryktesspridning (Rumor mongering)

När ryktesspridningsprotokollet tagit emot ett meddelande, så skickas meddelandet till ett förutbestämt antal medlemmar. Medlemmarna som meddelandet skickas vidare till väljs slumpmässigt. Då ett meddelande kommer till en medlem som tidigare mottagit samma meddelande är det en viss sannolikhet att meddelandet inte sprids vidare. Om meddelandet inte sprids vidare av en medlem kommer meddelandet fortsättningsvis ej att spridas vidare av medlemmen.

Tyvärr är inte det här protokollet 100 % pålitligt. Det finns en liten risk att alla noder ej får meddelandet då varje medlem väljer vem som skall få ett meddelande slumpmässigt. Den risken är dock väldigt liten om alla medlemmar känner till alla andra medlemmar.

För att medlemmarna i ett nät skall ha större chans att välja en medlem som ännu inte fått meddelandet kan en lista över de medlemmar som meddelandet redan besökt skickas med meddelandet. En sådan lista kallar vi för en meddelandelista. På så sätt, kan en medlem skapa sig en bättre bild över vilka andra medlemmar som inte fått ett meddelande. Om medlemmen dessutom minns vilka medlemmar som fått ett meddelande då ett likadant meddelande tas emot, blir bilden av vilka medlemmar som fått meddelandet ännu bättre och risken att inte alla får ett meddelande bli ännu mindre.

3.3.3 Skvallerprotokollet (Gossip)

Skvallerprotokollet fungerar likt ryktesspridningsprotokollet. Skillnaden är, att istället för en sannolikhet som bestämmer om ett meddelande skall skickas vidare så skickas ett meddelande vidare ett förutbestämt antal gånger. Även här skickas meddelandet bara vidare till ett fast antal, slumpvis utvalda medlemmar, ur den totala mängden medlemmar. Fördelen med att skicka vidare ett meddelande ett förutbestämt antal gånger är att tiden tills alla medlemmar har fått meddelandet minskar. Skvallerprotokollet har samma problem som ryktesspridning, d v s att det är inte 100% pålitligt. Pålitligheten har studerats av Lin et al [9], under förutsättningen att medlemmar kraschar. En meddelandelista kan användas av skvallerprotokollet på samma sätt som av ryktesspridningsprotokollet, för att få en ökad pålitlighet.

3.3.4 Riktat skvaller (Directional Gossip)

För att öka pålitligheten, har ett protokoll som kallas riktat skvaller utvecklats av Lin och Marzullo [8]. I riktat skvaller har varje medlem en lista över alla de andra medlemmarna i nätverket. Varje medlem i listan tilldelas en vikt som anger hur många vägar det finns till just den medlemmen. Då en nod tar emot ett meddelande kommer meddelandet att skickas vidare till alla noder med vikt lägre än ett förutbestämt värde eftersom relativt få medlemmar känner till den medlemmen. Sedan skickas meddelandet även vidare till ett bestämt antal andra slumpvis utvalda medlemmar som har en vikt som är större än det förutbestämda värdet. Även här kan en meddelandelista användas som innehåller vilka medlemmar som meddelandet besökt.

3.3.5 Resurssnålt sannolikhetsbaserat protokoll (Lightweight Probabilistic Broadcast)

Det här protokollet är väldigt likt skvallerprotokollet, och där igenom även ryktesspridningsprotokollet, men det har två skillnader. Den ena skillnaden är att medlemmar som kör detta protokoll bara har partiell kunskap om vilka andra medlemmar som ingår i

nätverket. Den andra skillnaden är att regelbundna sändningar genomförs. Den partiella kunskapen om andra medlemmar förändras hela tiden, och här följer en kort redovisning på hur Eugester et al [11] vill att detta skall fungera.

Varje medlem har en lista med referenser till ett fast antal andra medlemmar i nätverket. Detta kallas en medlemslista. Medlemslistan har samma längd hos alla medlemmar. För att protokollet skall fungera som bäst skall medlemslistorna vara enhetliga. Enhetliga medlemslistor har tre egenskaper. Den första egenskapen är att om samtliga medlemmars medlemslistor slås ihop skall en lista som innehåller referenser till samtliga medlemmar skapas. Den andra egenskapen är att varje nod skall finnas i lika många andra noders medlemslistor som längden på en medlemslista. Den tredje egenskapen är att medlemslistorna skall skapa en graf som inte är i närheten av att partitionera. För att hålla enhetliga medlemslistor, skickas en mängd referenser till andra medlemmar med varje meddelande. Får någon nod in en referens till en medlem som den inte har i sin medlemslista, läggs denna referens till i medlemslistan. Om medlemslistan blir större än en viss storlek tas slumpmässigt utvalda referenser bort ur listan tills storleken ej är större än den bestämda storleken. Medlemshanteringen och meddelandespridningen sköts alltså med samma meddelande. Om en medlem ej vill vara en medlem längre anmäler medlemmen detta till någon annan medlem. En lista över avhoppade medlemmar sparas hos varje nod, och den sköts på samma sätt som listan över andra medlemmar, d v s en lista över avhoppade medlemmar skickas med i varje meddelande. Varje avhopp sparas endast en begränsad tidsperiod för att inte ett avhopp skall sparas i systemet för alltid, då detta skulle kunna bli ett resursproblem. Medlemmar som kraschat och därmed inte hunnit avanmäla sig, försvinner gradvis från systemet, eftersom medlemmar plockas bort då medlemslistor vid en nod blir för långa.

Det finns en risk att nätverket blir partitionerat, även om sannolikheten för detta är väldigt liten. Genom att öka antalet medlemmar i nätverket eller genom att öka storleken på medlemslistorna vid varje medlem minskar sannolikheten för att nätverket ska partitionera. Ett sätt att förhindra att nätet delas är att bestämma att några medlemmar skall vara kända av alla andra medlemmar i systemet.

4 Valda protokoll och teorier

Det här kapitlet avslutar teoridelen i denna uppsats. Nästa del av uppsatsen kommer att beskriva de experiment vi genomfört på de två protokoll vi väljer ut i det här kapitlet. Det viktiga i det här kapitlet är motiveringen till varför vi väljer just skvallerprotokollet och protokollet för pålitlig rundsändning. Det är även intressant att läsa om vårt eget bidrag till beräkningen av en viss typ av grafer.

Beskrivningen av protokollen är mer detaljerad i detta kapitel, för att ge en inblick i hur protokollen fungerar. Först beskrivs varför protokollen blev valda och varför experiment med de andra protokollen uteblev. De två utvalda protokollen är skvallerprotokollet samt protokollet för pålitlig rundsändning. Dessa två protokoll beskrivs ingående. Efterföljande kapitel kommer att undersöka protokollen med hjälp av experiment.

För att skapa en graf som protokollet för pålitlig rundsändning kan sprida sina meddelanden över beskrivs algoritmer för att skapa Harary-grafer. Pålitlig rundsändning får de egenskaperna som grafen som meddelandena sprids över har. Därför beskrivs grafens egenskaper ingående. Genom att ändra i grafens struktur visar vi hur en Harary-graf kan skapas för att ge ett mindre avstånd mellan två godtyckliga noder med protokollet för pålitlig rundsändning. Algoritmen för protokollet för pålitlig rundsändning ges också. I nästa kapitel beskriver vi de frågeställningar som vi ställt oss för att jämföra protokollens sändningstid.

4.1 Motivering till val av protokoll

De två protokoll vi valde att testa mot varandra var protokollet för pålitlig rundsändning och skvallerprotokollet. Den största anledningen till att dessa protokoll valdes var att det var precis vad Lin et al [9] hade gjort i sitt arbete och det arbetet är vad som vi kallar för referenstestet. Skälet till att vi valde samma som referenstestet är att vi behövde starta någonstans och då det i referenstestet gjorts liknande jämförelser skulle vi ha något att jämföra vårt resultat med för att ge vår testmiljö en ökad validitet. De andra protokollen skulle dock ha undersökts om mer tid funnits tillgänglig.

Det finns fler anledningar till varför en jämförelse mellan dessa två protokoll är intressant. En jämförelse mellan ett deterministiskt och ett slumpmässigt urval av medlemmar som ett

meddelande skickas till, är mycket intressant för att studera hur deterministiska och slumpmässiga protokoll skiljer sig åt. Eftersom vi bara studerat ett enda deterministiskt protokoll, så faller valet på att studera protokollet för pålitlig rundsändning. Det andra protokollet som valts är skvallerprotokollet. Förutom motiveringen ovan ges här några fler anledningar.

Vi kan konstatera att både icke-oordningsprotokollet och det resurssnåla sannolikhetsbaserade protokollet sänder meddelanden även när ingen ny information tillkommit. Vi önskar bland annat att hålla nätverkstrafiken nere och båda dessa protokoll motverkar detta syfte. Dock hade den lilla trafik som resurssnålt sannolikhetsbaserat protokoll genererar kunnat vara överkomlig, men studie av detta protokoll lämnas på framtiden.

Enligt Demers [1] är skvallerprotokollet snabbare på rundsändning än ryktesspridningsprotokollet. Det är därför mer intressant att studera skvallerprotokollet.

Resurssnålt sannolikhetsbaserat protokoll är mycket likt skvallerprotokollet. Dock är resurssnålt sannolikhetsbaserat protokoll snabbast på att sprida ett meddelande då alla medlemmar känner alla andra medlemmar. Detta gör att protokollet uppför sig som skvallerprotokollet. Därför är det mer intressant att studera skvallerprotokollet.

I de två följande avsnitten beskrivs först skvallerprotokollet och sedan protokollet för pålitlig rundsändning.

4.2 Skvallerprotokollet

I detta avsnitt beskrivs det skvallerprotokoll som valts ut att testas i större omfattning och dess funktion presenteras även med ett mer formaliserat språk. Skvallerprotokollets övergripande funktion beskrevs i avsnitt 3.3.

4.2.1 Analys av protokollet

Då beräkningar av algoritmer som använder sig av slumpen kan bli komplicerade, föreslår Lin et al [9] att skvalleralgoritmen skall undersökas i simuleringar. Simuleringar och experiment gjorda av Lin et al har visat att då ett meddelande vidarebefordras fler gånger och när antalet andra medlemmar som det skickas vidare till minskas så ökar pålitligheten och nätverksbelastningen minskar. Detta gäller under förutsättningen att produkten av antalet gånger meddelandet vidarebefordras och antalet medlemmar som meddelandet skickas vidare till förblir den samma. Orsaken till dessa effekter är att ju senare ett meddelande skickas från en medlem desto större vetskap kan den ha om vilka andra medlemmar som redan fått meddelandet. Genom att öka antalet medlemmar som en medlem skickar ett meddelande till

och/eller genom att öka antalet gånger en medlem får skicka ett meddelande vidare ökar pålitligheten. Det maximala antalet skickade meddelanden ges av produkten av antalet fungerande medlemmar vid sändningens början, antalet gånger som en medlem skickar ett meddelande vidare och hur många medlemmar en medlem sänder ett meddelande till.

Vid nätverksstorlekar runt 32 medlemmar och kollisioner som påverkar sändningstiden har Lin et al har visat att protokollet för pålitlig rundsändning är snabbare än skvallerprotokollet på att sprida ett meddelande till alla medlemmar. Någon information om tiden för spridning av ett meddelande i en Internetmiljö har inte påträffats under litteraturstudierna.

4.2.2 Algoritm för skvallerprotokollet

Algoritmen som beskrevs i föregående avsnitt ges i Figur 4-1. Algoritmen är tagen från Lin et al

```
börja skicka meddelande m:
    skicka m till B_initial andra slumpvis utvalda medlemmar

när ( p mottar ett meddelande m från q )
    om ( p har mottagit m <= F antal gånger )
        lägg till sin referens i m
        skicka m till B slumpmässigt utvalda medlemmar som p vet
        att de inte redan fått m
```

m	meddelande
p, q	medlemmar
F	antal gånger ett meddelande skickas vidare
B	antal medlemmar som ett meddelande skall vidarebefordras till
$B_{initial}$	antal medlemmar meddelandet först ska skickas till (där det genererades)

Figur 4-1: Algoritm för skvallerprotokollet

4.3 Protokollet för pålitlig rundsändning

I kapitel 3.2 beskrevs den övergripande funktionen för protokollet för pålitlig rundsändning. I avsnitten nedan ges en mer ingående beskrivning av protokollet och dess egenskaper Protokollet för pålitlig rundsändning analyseras först i avsnitt 4.3.1. Då protokollet för pålitlig rundsändning får egenskaperna som den graf som meddelandet sprids över har, presenteras hur Harary-grafer byggs upp och vilka deras egenskaper är. Efter det beskrivs algoritmerna för att generera Hararys nätstrukturer och till sist presenteras algoritmen för protokollet för pålitlig rundsändning.

4.3.1 Analys av protokollet

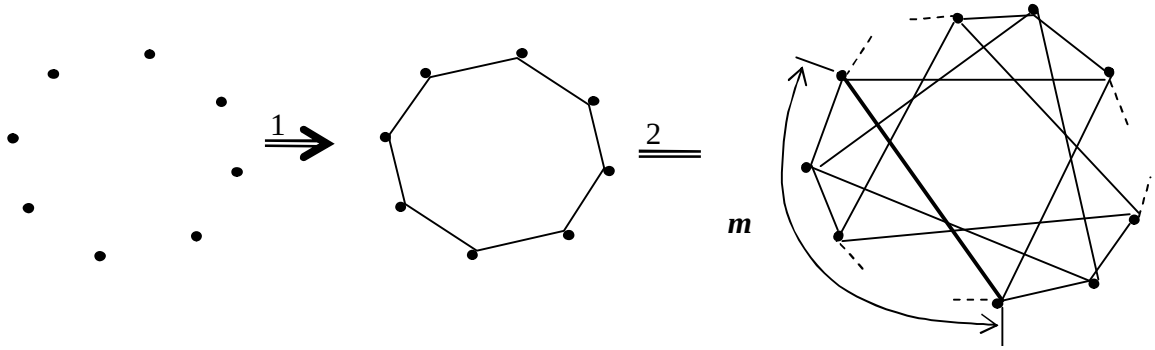
Protokollet för pålitlig rundsändning skickar ett meddelande, om meddelandet mottagits för första gången, till alla medlemmar som medlemmen känner till. Dock skickas meddelandet inte till den medlem som meddelandet kom ifrån (då denna medlem med säkerhet fått meddelandet).

Medlemmarnas kännedom av varandra kan beskrivas som den graf som skapas då kanter dras mellan de medlemmar som känner till varandra. För att studera egenskaperna hos protokollet för pålitlig rundsändning kan man således analysera egenskaperna för den graf som byggs upp. En sådan analys görs i avsnitten nedan. Valet att använda en Harary-graf gjordes då detta var den enda enkla algoritm för att skapa en graf åt protokollet för pålitlig rundsändning som påträffats under våra litteraturstudier. Grafen har också ett minimum antal länkar för en given felmodell, vilket genererar ett litet antal meddelanden vid användning av protokollet för pålitlig rundsändning. Först beskrivs hur en sådan graf kan skapas, sedan diskuteras pålitligheten och antalet meddelanden som genereras och till sist studeras tiden för rundsändning av ett meddelande baserat på antalet kanter ett meddelande passerar innan meddelandet nått alla medlemmar.

4.3.2 Hararys nätstrukturers uppbyggnad

I detta avsnitt studeras fallet då varje medlem endast känner till fyra andra medlemmar. Nätet kommer att klara av att upp till sammanlagt tre medlemmar eller meddelanden försvinner under en sändning. Det första som sker i uppbyggnaden av nätstrukturen är att medlemmar lär känna andra medlemmar i en enda stor ring. Detta steg illustreras som steg ett i Figur 4-2. Varje medlem känner nu till två andra medlemmar. Det återstår då att välja två medlemmar till. Dessa medlemmar väljs så att antalet hopp i ringen till en vald medlem blir två. Detta avstånd betecknas med m i steg två i Figur 4-2. För att göra grafen symmetrisk görs samma

sak åt båda håll i ringen. Det går även att välja ett avstånd på *m* som är större än två, vilket ändrar grafens egenskaper. Hur egenskaperna förändras beskrivs i avsnitten nedan.



Figur 4-2 Steg för att skapa av en Harary-graf

4.3.3 Hararys nätstrukturers egenskaper

Om protokollet för pålitlig rundsändning används då medlemmars kännedom om andra medlemmar valts med hjälp av Hararys nätstruktur, så kommer protokollet för pålitlig rundsändning under rätta förutsättningar att leverera ett meddelande till alla medlemmar. Förutsättningarna är att summan av antalet medlemmar och meddelanden som fallit bort under en sändning är under antalet medlemmar varje medlem har kännedom om, d v s tre eller färre i våra experiment.

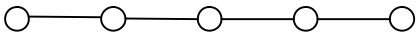
Formaliserar vi detta, på det sätt som beskrivits av Lin et al [9] , kallar vi ett nät med *n* medlemmar där varje medlem känner *t* andra medlemmar för *H(n, t)*. I Figur 4-3 nedan kallas medlem för nod, Hararys nätstruktur för en Harary-graf och kännedomen hos en medlem om en annan medlem kallas för en länk. En Harary graf har egenskaperna som beskrivs i den tidigare nämnda figuren. Samma jargong som i figuren används i resten av detta kapitel.

1. Den är **t-nod kopplad**. Om *t-1* noder tas bort kommer inte grafen att partitionera. Det finns dock sätt att ta bort *t* noder så att grafen blir partitionerad.
2. Den är **t-länk kopplad**. Om *t-1* länkar, tas bort kommer inte grafen att partitionera. Det finns dock sätt att ta bort *t* länkar så att grafen partitionerar.
3. Den har ett **minimum av länkar**. Tas någon länk bort är grafen inte längre **t-länk** eller **t-nod** kopplad.

Figur 4-3: Egenskaper hos en Harary-graf

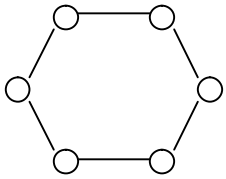
En Harary-graf skapas på olika sätt beroende på hur många meddelanden och medlemmar som kan tänkas försvinna. Vi kallar detta för antalet fel som tolereras under en sändning. Här följer några exempel på Harary-grafer.

Figur 4-4 illustrerar en *H(5, 1)* Harary-graf. Den har fem noder och tolererar inte något fel. Tidigare har vi nämnt att om ett nät delas, kommer en sändning inte att nå ut till alla medlemmar i ett nät. Det syns tydligt att under en rundsändning i en *H(5, 1)*-graf så kommer ett enda fel att dela grafen i två delar.



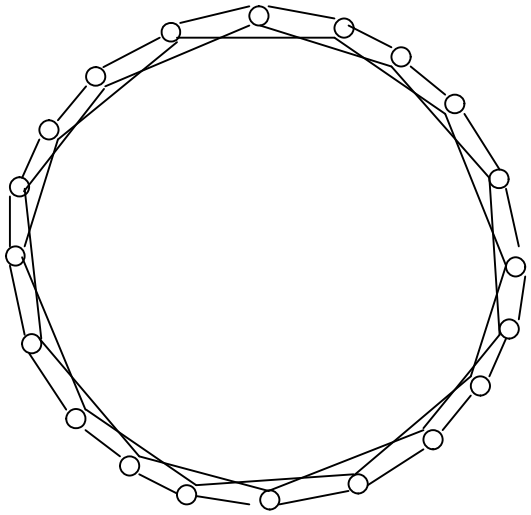
Figur 4-4: H(5, 1) Harary-graf med fem noder som inte klarar något fel

Figur 4-5 visar en *H(6, 2)* Harary-graf. Den tolererar att en länk eller nod försvinner, utan att dela nätet. Det förstår vi genom att det finns två sätt ett meddelande kan flöda runt om i grafen, och det därför måste inträffa två fel för att inte alla noder eller medlemmar ska få meddelandet.



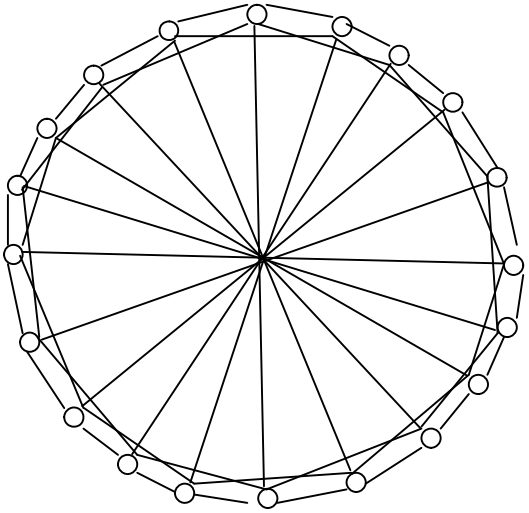
Figur 4-5: H(6,1) Harary-graf med sex noder som tolererar ett fel

En H(20, 4) graf illustreras i Figur 4-6. I den här grafen måste 3 fel inträffa för att grafen ska delas.



Figur 4-6: En H(20, 4) Harary-graf som klarar tre enkla fel

En H(20, 5) graf illustreras i Figur 4-7. För att dela den här grafen måste 4 fel inträffa.



Figur 4-7: En H(20, 5) Harary-graf som klarar fyra enkla fel

Om antalet noder, *n*, och antalet fel som ska tolereras, *t*, båda är jämna antal och det inte uppstår något fel under en sändning kommer antalet meddelanden som skickats kunna beräknas med Formel 4-1, hämtad från Lin et al [9] .

$$n(t-1)+1$$

Formel 4-1: Antal meddelanden som skickas med protokollet för pålitlig rundsändning

Om antalet noder är ojämnt och antalet fel som skall tolereras är jämnt kan ett extra meddelande skickas då grafen inte är symmetrisk. Så länge nätet inte är delat kommer protokollet för pålitlig rundsändning alltid att leverera ett meddelande till alla medlemmar, under tidigare nämnda villkor.

4.3.4 Algoritmer

Både algoritmen för protokollet för pålitlig rundsändning i Figur 4-8 och algoritmen för att skapa en *H(n, t)* Harary-graf i Figur 4-9, har hämtats ur Lin et al[1].

p skickar *m* till alla kända medlemmar

om (*p* mottar ett meddelande *m* från *q* som ej mottagits förut)
  p skickar *m* till alla sina kända medlemmar utom *q*
  p levererar *m* till sig själv

p, q	Medlemmar
m	Meddelande

Figur 4-8: Algoritm för protokollet för pålitlig rundsändning

H(n, 1): Trädet med länkar $\forall i : 0 \le i < n-1 : (i, i+1)$

H(n, 2): Ringen med länkar $\forall i : 0 \le i < n-1 : (i, i+1 \bmod n)$

H(n, t) om *t* > 2 och *t* är ett jämnt tal: Konstruera först *H(n, 2)*.
  Sedan, för varje värde av $m : 2 \le m \le t/2$ lägg till länkar *(i, j)*
  där $|i - j| = m \bmod n$

H(n, t) om *t* > 2 och *t* är ett udda tal: Konstruera först *H(n, t-1)*.
  Sedan, koppla alla par *(i, j)* av noder så att $j - 1 = \lfloor n/2 \rfloor$

i,j	Medlemmar
n	Antalet medlemmar
t	Antalet kanter eller antalet medlemmar varje medlem känner till
$\lfloor x \rfloor$	Avrunda *x* neråt till närmaste heltal

Figur 4-9: Algoritm för att skapa en Harary-graf

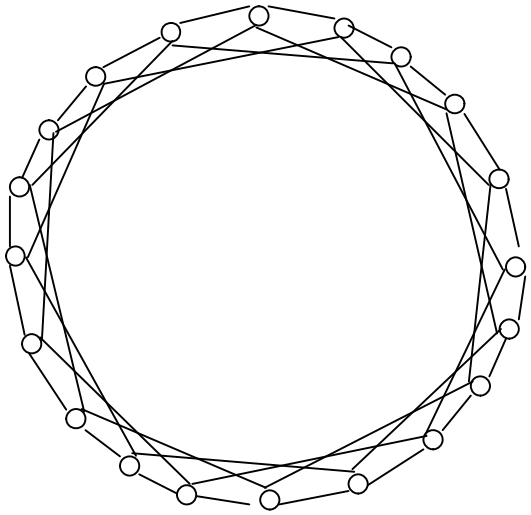
4.4 Förbättrade Harary-grafer med avseende på pålitlighet och avstånd

I avsnitten nedan beskrivs några modifierade Harary-grafer. I dessa grafer har noderna ett mindre avstånd till den nod som ligger längst bort, jämfört med graferna i 4.3.3. Först beskrivs en förändrad Harary-graf som Lin et al beskrivit. Deras graf skapades för att uppnå

en högre pålitlighet. I vårt fall vill vi minska hopplängden för att tiden för rundsändning skall minskas. För att få en så liten total hopplängd som möjligt beskriver vi i 4.4.2 ett sätt att skapa Harary-grafer med ett litet antal hopp mellan alla noder.

4.4.1 Pålitligare Harary-grafer

Lin et al [9] har modifierat algoritmen för att skapa en Harary-graf så att de får ut en graf med högre pålitlighet. Genom att öka avståndet mellan grannarna, det vill säga *m* i både Figur 4-2 och Figur 4-10, kan en Harary-graf med ökad pålitlighet skapas. Det vill säga, även om det inträffar ett fel mer än det antal fel som en Harary-graf garanterat klarar av, så kommer grafen ändå med stor sannolikhet inte att delas. Lin et al har visat varför det blir så, men vi tar inte upp det i detalj här. Den modifierade algoritmen finns i Figur 4-11. Figur 4-10 visar en graf som skapats med hjälp av den modifierade algoritmen, med ökat avstånd mellan grannarna. Grafen har 20 noder men klarar med stor sannolikhet fler än tre fel utan att nätet delas jämfört med en Harary-graf skapad med de ursprungliga algoritmerna, till exempel grafen som visas i Figur 4-6.



Figur 4-10: H(20, 4) Harary-graf med en ökad pålitlighet

Orsaken till att pålitligheten ökar är att det finns färre sätt att dela den modifierade grafen på än det finns att dela den omodifierade grafen på. Det har även visats sig att det går att skapa Harary-grafer som med ännu högre pålitlighet, alltså Harary-grafer som har ännu färre sätt att bli delade på, men ännu inga algoritmer för att skapa dessa grafer. Hopplängden i grafen som visas i Figur 4-10 har även en lägre hopplängd jämfört med grafen i Figur 4-6. Den minskade hopplängden gör att tiden för rundsändning kan minska. Pålitligheten vid förluster av meddelanden blir också högre när hopplängden minskas, då det finns färre chanser att förlora ett meddelande om det skickas färre gånger.

H(n, t) Konstruera först *H(n, 2)*

Sedan, för varje värde av *m* där $2 \le m \le \frac{t}{2}$

lägg till länkar *(i, j)* där | *i-j* | = *(m*+1*)* mod *n*

Figur 4-11: Algoritm för en modifierad Harary-graf

4.4.2 Minimera hopplängden

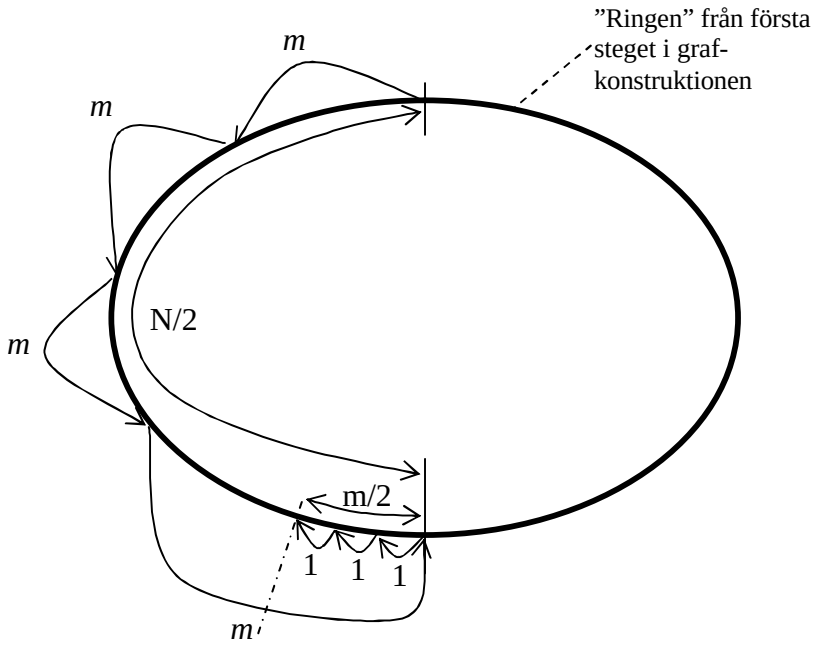
För att minska tiden för rundsändning vill vi minska antalet hopp ett meddelande gör för att nå alla medlemmar. Då den litteratur som skrivits av Frank Harary om Harary-grafer ej gick att få tag på, har vi har själva modifierat algoritmen för att skapa en Harary-graf för att försöka minimera hopplängden mellan medlemmar som har störst avstånd till varandra. Antagligen finns denna modifikation redan i den ursprungliga skriften, men vi har inte kunnat verifiera detta.

Genom hela uppsatsen har vi har studerat Harary-grafer där varje medlem känner till fyra andra medlemmar, det värde som benämns som *t*. Vi försöker minimera hopplängden genom att försöka beräkna hur många hopp (se Figur 2-2 för en definition på hopp) ett meddelande som mest gör, med avseende på nätstorleken och hopplängden *m*, som kan ses i Figur 4-2 och Figur 4-10. Formel 4-2 beräknar en approximation av detta antal hopp.

$$h = \frac{N}{2m} + \frac{m}{2}$$

Formel 4-2: Antalet hopp som krävs för en Harary-graf med fyra grannar

Vi har kommit fram till formeln genom observationer av Harary-grafen. För att beskriva vår härledning, behöver vi först notera att en *H(n, 4)* Harary-graf har symmetriskt fördelade grannar. Vidare noterar vi att de graferna ser ut som ringar. Ett meddelande måste minst utföra $\frac{N}{2m}$ hopp längs kanterna på denna ring innan det möter sig själv på ”andra sidan”. När meddelandet mött sig själv på ”andra sidan” kommer det ha hoppat över ett antal noder, och ytterligare $\frac{m}{2}$ hopp behövs för att täcka in dessa noder. Detta illustreras i Figur 4-12.



Figur 4-12: Antalet hopp ett meddelande gör

Vi antar att Formel 4-2 är åtminstone approximativt korrekt och använder den till att beräkna ett minsta *h* med avseende på *m*. Genom att derivera formeln med avseende på *m* får vi fram:

$$h' = -\frac{N}{2m^2} + \frac{1}{2}$$

Sätter vi nu *h'=0* kommer vi att få fram de extrempunkter som formeln har. Resultatet är att extrempunkterna ges av:

$$m = \sqrt{N}$$

En examination av den positiva extrempunkten ger att den är en minimipunkt. Men eftersom hopp bara kan göras i hela steg, och inte delar av ett steg, blir den formel som vi använder oss av för att beräkna hopplängden *m*:

$$m = \lfloor \sqrt{N} \rfloor$$

Formel 4-3: Formel med vilken vi bestämmer hoppstorleken i programmet

4.4.3 En utvecklad observation av Harary-grafer

Inledande tester visade att även nätstorleken inverkar på det verkliga antalet hopp ett meddelande gör relativt det beräknade. Vid närverksstorlekar som kan beräknas som m^2 så kommer antalet hopp ett meddelande som mest gör kunna beräknas med Formel 4-2. Vid

andra nätverksstorlekar kommer det verkliga antalet hopp vara något mindre. Vid nätverksstorlekar där $N=m^2+m/2$ uppträdde en förbättring på $m/4$ hopp. Om vi korrigerar Formel 4-2, bli formeln så som Formel 4-4 visar.

$$h = \frac{N}{2m} + \frac{m}{2} - \frac{m}{4}$$

Formel 4-4: Utvecklad beräkning av antalet hopp

Denna förbättring kommer av en överlappning av vilka som får meddelandet på bortre sidan av ringen. Dock är det inte säkert att förbättringen gäller från alla medlemmar, utan det är troligt att denna förbättring endast gäller sett från medlemmen ”högst upp”. Vidare om vi använde Formel 4-4 skulle en annan extrempunkt finnas än $m = \sqrt{N}$. Då vi inte kunde finna en generell beräkning av antalet hopp valde vi beräkningen beskriven av Formel 4-2 då vårt resonemang gällde för flera av de Harary-grafer vi observerat.

5 Frågeställningar och försöksplanering

Det här kapitlet markerar början på den tredje delen i uppsatsen. I den här delen kommer vi att genomföra två experiment, ett huvudexperiment med syftet att avgöra om det finns någon skillnad i tiden för rundsändning för de två protokollen vi valde ut i förgående del och ett biexperiment för att undersöka om fördröjningar och förluster kan påverka skillnaden i tid.

I detta kapitel återger vi frågor kring dessa protokoll och formulerar hypoteser, med inriktning mot vilket protokoll som har den kortaste tiden för allmänsändning. I nästa kapitel väljer vi ut värden till alla parametrar, både till vårt simulerade Internet samt till protokollen. Det är dessa värden som används i de experiment som utförts för att besvara frågeställningarna.

Detta kapitel börjar med att vi sätter upp ett par hypoteser kring tiden för rundsändning, för att besvara en av frågeställningarna i 1.3 om den tiden beror av valet av protokoll. Sedan presenteras hypoteser för att försöka besvara frågan om förluster och fördröjningar kan påverka resultatet från den tidigare frågeställningen. Även de metoder som skall användas för att besvara frågeställningarna beskrivs kort.

5.1 Hypoteser

Den huvudsakliga frågeställningen i den här uppsatsen är om det finns skillnad i tidsåtgången för rundsändning mellan de två utvalda protokollen. Under resans gång upptäckte vi att det kan vara intressant att studera om denna skillnad i tidsåtgång för rundsändning kan påverkas av olika nivåer av förluster och fördröjningar, då förluster och fördröjningar varierar på Internet. Vi börjar här med att fokusera på huvudfrågeställningen och ställer upp hypoteser som beskriver den. Sen beskriver vi bifrågeställningen. Huvudfrågeställningen kommer att analyseras statistiskt, genom ett så kallat konfidensintervall, men bifrågeställningen kommer inte att analyseras med någon statistisk metod.

5.1.1 Huvudfrågeställningen

Hypoteser i Figur 5-1 beskriver huvudfrågeställningen mer formellt. Frågorna är ställda med avseende på att protokollen körs i en Internet-liknande miljö.

$H_{konfidens\text{-}0}$:	Rundsändning genom protokollet för pålitlig rundsändning tar samma tid som genom skvallerprotokollet.
$H_{konfidens\text{-}1}$:	Rundsändning genom det ena protokollet tar inte samma tid som genom det andra.

Figur 5-1: Hypoteser för huvudfrågeställningen

5.1.2 Bifrågeställning

För att besvara den frågeställningen om hur skillnaden i tidsåtgång mellan de båda protokollen förändras när fördröjningar och förluster förändras, har vi i Figur 5-2 ställt upp två hypoteser. Genom dem vi kan se att:

$H_{trend\text{-}0}$:	Relativa skillnaden i tid för rundsändning mellan skvallerprotokollet och protokollet för pålitlig rundsändning påverkas inte av olika nivåer av förluster och fördröjningar.
$H_{trend\text{-}1}$:	Relativa skillnaden i tid för rundsändning mellan protokollen förändras vid olika nivåer på förluster och fördröjningar.

Figur 5-2: Hypoteser för bifrågeställningen

5.2 Försöksplanering

Testerna som beskrivs i de två avsnitten nedan sätter vi upp för att försöka nå en slutsats baserad på hypoteserna. För huvudfrågeställningen väljer vi ett konfidensintervalltest vilket är ett test som jämför två medelvärden med varandra. Vi kallar detta test för djuptestet. Bifrågeställningen skall besvaras genom att räkna ut skillnaden i tidsåtgång för de bägge protokollen vid olika värden på förluster och fördröjningar. Detta test kommer i fortsättningen att kallas för trendtestet.

5.2.1 Tester för att besvara huvudfrågeställningen: Djuptest

Vi behöver ett sätt att avgöra om de olika tiderna skiljer sig åt. Då enskilda tester kan ge olika resultat, behöver vi ett test som inte beror på osäkerheten i enskilda test. Vi har valt att använda en konfidensintervallskattning. Konfidensintervallskattning används för att uppskatta

skillnaden mellan två stokastiska variablers medelvärde utifrån observerade mätdata. I vårt fall är dessa två stokastiska variabler de tider som krävs för rundsändning, och vi har endast mätdata att tillgå. Våra variabler är stokastiska eftersom de inte håller ett enda värde, utan deras värde utgörs av en sannolikhetsfunktion, en funktion som är för oss okänd. Dock går det att approximera funktionen. Information om hur konfidensintervallet skall beräknas är hämtad ur en lärobok i statistik av Vännman [14]. Ur mätdatan, vilket är alla möjliga resultat som kan fås, tas ett antal stickprover, vilka är de vi uppmäter. För att de statistiska verktyg vi valt ska fungera behöver dessa stickprover vara oberoende från varandra, och helt slumpmässigt utvalda.

Till en konfidensintervallskattning behöver vi medelvärdet till de båda stokastiska variablerna, d v s vi behöver veta medelvärdet på de båda tiderna. Dessutom behöver vi veta variansen. I vissa fall är den känd, eller kan beräknas på annat sätt, men till det här testet måste variansen till de båda variablerna uppskattas utifrån våra stickprov. Skattningen till både medelvärdet och variansen görs på vanligt statistiskt vis.

Vidare behöver vi veta hur hög konfidensgrad skattningen ska ha. Ju högre konfidensgrad, ju säkrare är vi på att den faktiska skillnaden i medelvärdet finns med i intervallet Genom en hög konfidensgrad, blir lätt skattningsintervallet brett. För att göra intervallet smalare så behöver vi enligt Vännman genomföra minst 30 tester på vardera protokollet.

För att skatta konfidensintervallet, använder vi Formel 5-1 som är hämtad ur Vännmans bok. S är beräknad standardavvikelse för t-fördelningen.

$$\bar{x} - \bar{y} \pm Z\sqrt{\frac{S_x^2}{n_x} + \frac{S_y^2}{n_y}}$$

Formel 5-1: Konfidensintervallskattning mellan två mängder av observerade tal

Vanligt är att vid sådana här tester välja en konfidensgrad på 99 %. Därför har vi också valt att göra testet med denna konfidensgrad. Värdet till normalfördelningen hämtades ur en vanlig statistiktabell. Det Z-värde som ger ett konfidensintervall på 99 % är 2,58, enligt tabellen.

5.2.2 Tester för att besvara bifrågeställningen: Trendtest

För att undersöka om det finns påverkningar på den relativa skillnaden i tidsåtgången för rundsändning som beror på förluster och fördröjningar, väljer vi en grafisk metod. För varje kombination av fördröjningar och förluster beräknas ett medelvärde för tiden. Medelvärdet på tiden för skvallerprotokollet dividerades med medelvärdet på tiden för protokollet för pålitlig

rundsändning. Även för antalet hopp divideras medelvärdet för skvallerprotokollet med medelvärdet för pålitlig rundsändning. Vi inkluderar även en analys av skillnaden i hopp, eftersom vi tror att det finns ett samband mellan antalet hopp och tidsåtgången för rundsändning. Genom att åskådliggöra divisionerna i ett diagram, med fördröjningar i ena axeln, förluster i andra och den proportionella skillnaden i den tredje, hoppas vi kunna se om funktionsytan lutar. Om funktionsytan lutar kraftigt och endast åt något ett håll, kan vi säga att proportionella skillnaden i tid faktiskt ändras beroende av nivåerna på förluster och fördröjning. Om funktionsytan är böjd, eller inte påminner om ett plan, kan vi inte dra några sådana slutsatser. Studien genomförs på varje nätstorlek separat, och vi skiljer på resultaten från tidsanalysen och hoppanalysen.

5.3 Mätdata

Vi är intresserade av att främst mäta tidsåtgången för rundsändning. Därför behöver vi genomföra tester där vi kan erhålla stickprov för denna tidsåtgång. För att kunna analysera vad som är anledningen till att en viss tid uppmätts redovisar vi även antalet hopp som behövts innan ett meddelande nått den medlem som mottog meddelandet sist. För att få en mer komplett bild av hur ett meddelande spreds under ett test, och därmed vad som orsakade den uppmätta tiden, skulle vi behöva redogöra för vid vilka tidpunkter ett meddelande mottogs för första gången av en medlem. Vi skulle också behöva skapa en graf som redovisar hur alla meddelanden skickades mellan medlemmarna. Att analysera tidpunkterna då ett meddelande mottogs för första gången. Att analysera de spridningsgrafer som kunde skapas skulle ta väldigt lång tid. Då vi inte hade tid att analysera alla resultat valde vi att samla in data om tidsåtgång för rundsändning och antalet hopp, då tiden beräknades räcka för att analysera de data som då skulle genereras.

Vid en screening visade det sig att vår testmiljö inte kunde mäta tiden för rundsändning på ett rättvist sätt. Tabellen i appendix F.1.1 visar tiden för rundsändning med protokollet för pålitlig rundsändning. Tabellen i appendix F.2.2 visar tiden för rundsändning med skvallerprotokollet. I båda fallen gjorde meddelandet lika många hopp för att leverera meddelandet till alla medlemmar. Tiden borde då vara i stort sett densamma för bägge protokollen. Det visade sig dock att skvallerprotokollet fick en mindre tid än protokollet för pålitlig rundsändning. Tiderna som uppmättes var också betydligt större än antalet hopp multiplicerat med fördröjningen. Mer information om testmiljöns oförmåga att mäta en korrekt tid finns i appendix C.3. För att mäta tiden bestämde vi oss då för att använda antalet

hopp för att mäta tiden för rundsändning. I avsnitt 2.7 beskrevs att detta kan ge ett korrekt resultat vid ett homogent och statiskt Internet där sändningstiderna är försumbara.

5.4 Utrustning

Testerna kommer att genomföras på likvärdiga datorer från datortillverkaren Dell. Processorerna är av typen Pentium4 på 1,8 GHz. Primärminnet är 512 MB. Genom att beräkna storleken på de objekt vi skulle använda och räkna ut hur mycket plats 150 medlemmar skulle ta kunde vi konstatera att alla medlemmar skulle få plats i primärminnet och inte riskera att bli swappade. Operativsystemet vi kommer använda är Red Hat Linux, v 6.2.

Då vi testade att köra alla medlemmar på en dator upptäckte vi att processorn endast utnyttjades till några få procent även vid flera hundra medlemmar. Då antog vi att det fanns tillräckligt med processorkraft för att tiden för rundsändning inte skulle påverkas nämnvärt av att ha alla medlemmar på en dator. Att ha alla medlemmar på en dator skulle underlätta tidtagningen då olika datorer ej behövde synkronisera sina klockor. Vi valde då låta alla medlemmar rymmas på samma dator.

5.5 Förändrad tidmätningsmetod

Under preliminära försök visade det sig att tidmätningen inte fungerade fullt ut i vår testmiljö. Tiden för ett meddelande att göra ett visst antal hopp för ena protokollet blev inte densamma tiden som när samma antal hopp gjordes med det andra protokollet. Detta gör att vi förändrar metoden för tidmätning från att mäta verklig tid, till att mäta antalet hopp och sen använda sambandet mellan tid och antalet hopp för att beräkna den tid som experimentet skulle ha tagit. Se bilaga C.3 för mer information.

6 Val av parametrar till experimenten

I det här kapitlet väljs alla parametrar till experimenten. I nästa kapitel kommer dessa parametrar att användas i en verklig testmiljö, där vi utför våra tester och hämtar stickproven från vår mätdata.

För att besvara de hypoteser som framlades i förra kapitlet, försöker vi här välja ut de parametrar som ger oss svaren. Vi väljer ut nätverksstorlekar, vilka fördröjningar och förluster vi vill använda, beskriver vårt simulerade Internet, samt tillsätter protokollspecifika värden. Motivering av valen för alla värden ges. Sen redovisar vi några skillnader mellan våra parametrar och parametrarna i referenstestet. Sist i det här kapitlet finns en sammanfattning av alla dessa val och värden. I nästa kapitel kommer vi att beskriva den testmiljö som testerna genomfördes i.

6.1 Val av nätverksstorlekar

Vi vill testa viktiga antal medlemmar i både protokollet för pålitlig rundsändning och skvallerprotokollet. I referenstestet har 32 medlemmar använts för att jämföra protokollet för pålitlig rundsändning och skvallerprotokollet. De nådde bland annat resultatet att protokollet för pålitlig rundsändning var snabbare än skvallerprotokollet. Genom att genomföra deras experiment och få samma resultat som referenstestet skulle vår testmiljö få en hög validitet. Därför valde vi att göra ett konfidensintervallskattningstest med 32 medlemmars nätverksstorlek.

Gupta et al [4] och Eugster et al [11] , har indikerat att skvallerprotokollet är effektivast vid stora nätstorlekar. Vi anser att stora nätverksstorlekar är nät med över 100 medlemmar. Därför har vi valt att testa både konfidensintervalltest och trendtest på 105 och 150 medlemmar. För att bättre se eventuell utveckling av den relativa skillnaden mellan tidsåtgången för rundsändning med de två protokollen, valde vi även att testa med nätverksstorleken 68 medlemmar. Dessa storlekar valdes också med hänsyn utifrån resonemanget, som återfinns i avsnitt 4.4.3, för att nå en bieffekt där vissa nätverksstorlekar har bättre spridningsförmåga än andra nätverksstorlekar

6.2 Val av fördröjnings- och förlustnivåer

Då alla simuleringar skulle ske internt i datorn, så behövde vi ett sätt att simulera olika aspekter av Internet. De två parametrarna vi valde var förluster och fördröjningar. De värden på fördröjningar som uppgetts av Matrix och Internet Traffic Report var medelvärden. Hur fördröjningarna varierar runt medelvärdet uppgavs ej. Vi resonerade då att variationer av fördröjningar skulle ge ett medelvärde då ett meddelande färdas över många länkar. då detta skulle leda till samma resultat som att testa ett statiskt Internet valde vi ett statiskt Internet. Detta resonemang tas upp i avsnitt 9.3 om validitet.

6.2.1 Huvudtest

Enligt olika källor på Internet [7] [12], är medeltalen för fördröjningar och förluster 80 ms respektive 1,2 %. Vi valde att testa på en fördröjning på 80 ms och förluster på ungefär 1,2 %. Dessa värden skulle då ge oss ett simulerat medel-Internet.

6.2.2 Trendtest

För dessa tester valde vi att testa fördröjningar i intervallet 0 till 160 ms, och förluster i intervallet från 0 till 4,5 %. Intervallen på förluster och fördröjningar är tagna från undre och övre gränserna som uppmäts på Internet av Matrix [7] och Opnix [12]. Storleken på paketen som använts för att mäta fördröjningar och förluster har inte uppgetts av varken Matrix eller Opnix. Försök till kontakt med organisationerna som utfört testerna har misslyckats. Dock uppges att ping-tjänsten använts för mätningarna. Denna tjänst kan ha olika storlek på paketen, men grundinställningen är på 32 byte. Sen tillkommer några extra byte för overhead som kommer av protokollhuvuden på olika lager. Det skall noteras att det händer att routrar är inställda på att bara tillåta en viss ping-trafik. Om denna ping-trafik blir för stor kastas meddelanden. Det är därför möjligt de uppmätta värdena ej är representativa för meddelanden skickade med UDP. De båda intervallen på förluster och fördröjningar delades in i fem respektive sex testnivåer. Alla kombinationer av förluster och fördröjningar skall testas för att ge en fullständig överblick inom de valda intervallen. Anledningen till att så få testnivåer valdes var att antalet tester inte skulle bli allt för många då tid inte fanns för ett större test. Tabell 6-1 visar de valda värdena på förluster och fördröjningar.

fördröjning(ms)	förluster(%)
0	0,0
40	0,9
80	1,8
120	2,7
160	3,6
	4,5

Tabell 6-1: Parametrar till trendtestet

6.3 Protokollspecifika värden

Protokollen har några parametrar som måste tilldelas värden. I det här avsnittet kommer vi att tillsätta dessa parametrar värden. Parametrarnas värden kommer inte att förändras i dessa tester, men det vore intressant att genomföra sådana tester.

6.3.1 Protokollet för pålitlig rundsändning

Protokollet för pålitlig rundsändning behöver ha ett fast antal medlemmar i listan över vilka medlemmar som den känner till. Antalet medlemmar i listan, betecknas vanligen med *m*. Antalet medlemmar sattes till fyra för att efterlikna det test som utfördes av Lin et al [9] . Vidare är fyra just det antal som vi valt att studera noggrannare i kapitel 4.4, och vi har ännu inte studerat det generella fallet.

6.3.2 Skvallerprotokollet

Skvallerprotokollet har två parametrar som skall tillsättas värden. Dessa parametrar är antalet medlemmar en medlem skall skicka ett meddelande till samt hur många gånger ett meddelande skall vidarebefordras. Varje medlem i pålitlig rundsändning kommer att skicka ett meddelande vidare till ytterligare tre medlemmar som beskrivits i avsnittet ovan. Undantaget är den första medlemmen som skickar meddelandet till fyra medlemmar. Därför valde vi att skvallerprotokollet ska skicka vidare ett meddelande till tre slumpmässiga utvalda medlemmar för att vara jämförbart med protokollet för pålitlig rundsändning.

För att välja hur många gånger ett meddelande skall skickas vidare utförde vi en screening. Det visade sig då att pålitligheten blev låg då ett meddelande tilläts skickas vidare två gånger då paketförlusterna var 1,2%. Om vi istället valde att en medlem fick skicka ett meddelande vidare tre gånger så var fanns pålitligheten vara hög. Vi sätter alltså antalet gånger en medlem som använder skvallerprotokollet får sända ett meddelande vidare till tre.

6.4 Antal tester

På något sätt behöver vi bestämma antalet tester vi vill genomföra. Om vi vill ha en hög signifikansnivå på konfidensintervalltestet måste vi använda ett stort antal tester, speciellt då vi inte vet speciellt mycket om den fördelning som tiden för rundsändning kommer att få. Därför måste vi utnyttja att det är möjligt att använda en skattning av variationen om tillräckligt många tester utförs.

6.4.1 Huvudtest

Vi väljer att göra 200 tester, som kommer att tillåta oss att använda även ett snävt konfidensintervall, på kanske 95 % eller snävare [14]. Det finns ingen anledning att göra färre tester, då det ändå går så fort som tre till fyra sekunder att genomföra ett test. Att göra fler tester skulle göra att vi får knappt om tid för att analysera resultaten.

6.4.2 Trendtest

Vi valde att göra 10 tester per konfiguration i detta experiment. Om vi hade valt att göra fler tester, så skulle det totala antalet tester för att genomföra detta experiment ha blivit otroligt mycket större, och experimentfasen skulle ha tagit åtskilligt fler antal dagar. På grund av att experimentmiljön blev klar så sent så kunde vi inte genomföra fler tester.

6.5 Skillnader i testmiljön jämfört med referenstest

När vi valt alla de parametrar som behövdes för att utföra testerna upptäckte vi att de parametrar vi valt och var intresserade av skilde sig från referenstestet på flera viktiga punkter. De punkter som skilde de båda uppsättningarna åt var följande:

Referenstestet hade ett nät där kollisioner inträffade. Kollisionernas påverkan på sändningstiden skulle därmed kunna bidra till att experimenten ej blir jämförbara. Fördröjningen på ett meddelande var vald för att likna ett lokalt nätverk, medan vi var mer intresserade av en Internet-miljö. Förlusterna i en Internet-miljö skilde sig också då referenstestet använde ett lokalt nätverk med kollisioner som följd.

Den Harary-graf vi ville använda, då avståndet mellan medlemmar skulle minska, var ej den Harary-graf som använts i referenstestet. Detta skulle också kunna bidra till att experimenten ej kan bli jämförbara.

Den sista aspekten som skiljer de båda experimenten åt är att i referenstestet skickade skvallerprotokollet vidare ett meddelande två gånger, för att antalet meddelanden skulle bli

ungefär lika för skvallerprotokollet och protokollet för pålitlig allmänsändning. Vi har däremot valt att jämföra protokollen då skvallerprotokollet tillåts skicka ett meddelande vidare tre gånger, då en screening visade att pålitligheten vid 1,2% paketförluster var god.

Då vi inte hade tid att både upprepa referenstestet och att testa de aspekter som vi tyckte var intressanta bestämde vi oss för att ej upprepa referenstestet. Vi ansåg att de parametrar vi valt var viktigare för vårt syfte än att bekräfta referenstestet. Vi valde därför att testa med våra valda parametrar varför en jämförelse mellan vårt resultat och referenstestets resultat skjuts på framtiden.

6.6 Sammanfattning av testvärdena

6.6.1 Huvudtest

200 tester genomfördes, per konfiguration och protokoll för konfidensintervallskattning. Tabell 6-2 ger en lista av de konfigurationer vi kommer att testa.

Nätstorlek	förluster	fördröjning
68	1,20%	80 ms
105	1,20%	80 ms
150	1,20%	80 ms

Tabell 6-2 Konfigurationer för konfidensintervallskattning

6.6.2 Trendtest

För trendtestet genomfördes 10 tester, på 3 olika nätstorlekar (68, 105 och 150), på 5 olika fördröjningar (0, 40, 80, 120 och 160 ms) samt 6 olika förlustnivåer (0 %, 0,9 %, 1,8 %, 2,7 %, 3,6 %, 4,5 %). Detta blir totalt 10 * 3 * 5 * 6 = 900 tester per protokoll, d v s 1800 tester för att utföra detta experiment.

7 Experimentbeskrivning

Nu har vi tillräckligt med information om protokollen för att klara av att designa och sätta upp en experimentmiljö. I nästa kapitel redovisar vi resultaten från testerna i detta kapitel, och i kapitlet 9 har vi analyserat resultaten genom att använda metoderna som beskrevs i avsnitt 5.2.

I det här kapitlet kommer vi att ge en övergripande beskrivning av vår testmiljö. Vi börjar med att beskriva hur en testomgång sätts upp, hur vi samlar ihop den relevanta statistiken och beskriver varför vi väljer C++ som programspråk. Sen följer en översikt över hur kommunikationen mellan medlemmarna sker i vår testmiljö, d v s hur vi genererar fördröjningar och förluster.

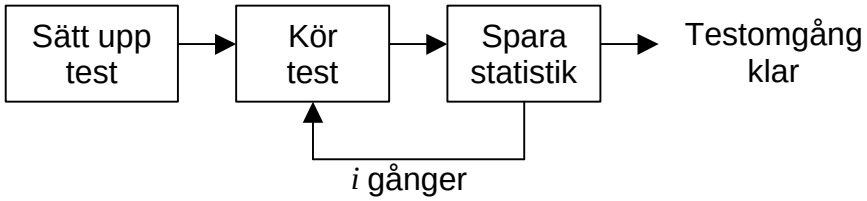
7.1 Upplägg av testmiljön

Testmiljön består av två delar. Den ena delen sköter uppsättning av testet, start av testet samt rapportering av testdata. Denna del beskrivs i 7.1.1. Den andra delen är den del som utför testerna, alltså flera medlemmar som deltar i en rundsändning. En beskrivning av den andra delen återfinns i de återstående avsnitten. Det är möjligt att låta medlemmar befinna sig på olika datorer men i våra tester finns alla medlemmar på samma dator.

7.1.1 Uppsättning av experimentet och rapportering av tider

Uppsättningen och rapporteringen av ett test sker med hjälp av en experimentövervakare och flera medlemmar. Experimentövervakaren är ansvarig för att tala om för medlemmarna vilket protokoll som ska testas, utse vilka medlemmar som känner andra medlemmar, starta testerna och samla in statistik från testet. Experimentövervakaren sammanställer och sparar också statistiken. Medlemmarna är ansvariga för att utföra testerna. Då medlemmarna utför testerna skickar de meddelanden mellan sig och blir då sådana medlemmar som beskrivits tidigare. En beskrivning av hur experimentet sätts upp och sammanställs beskrivs nedan. För en mer ingående beskrivning se bilaga A.

I Figur 7-1 visas hur *i* stycken tester genomförs.



Figur 7-1: Översikt av en testomgång

Experimentövervakaren, börjar med att vänta in att tillräckligt många medlemmar skall anmäla sig till den. När alla medlemmar har gett sig till känna, utför experimentövervakaren några beräkningar om bland annat vilka medlemmar som ska känna till vilka andra. Sen skickar experimentövervakaren ut all information till medlemmarna, som då alla utom en startar det valda protokollet. Experimentövervakaren sätter då igång testet genom att meddela den sista medlemman att testet kan starta. Under tiden som testet utförs ”sover” experimentövervakaren för att inte påverka testet.

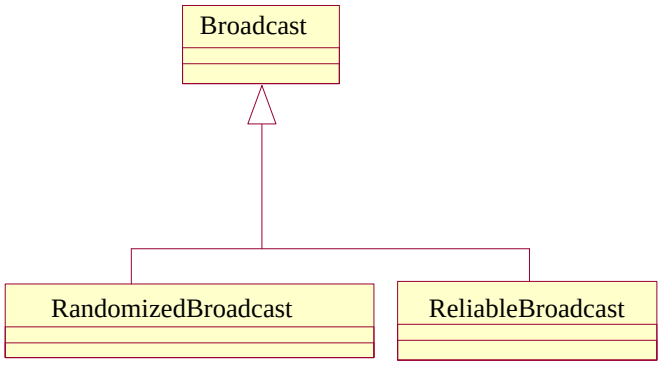
När experimentövervakaren ”vaknat” samlas statistiken för testet in. Denna statistik består av de tidpunkter medlemmarna mottog meddelandet första gången, antal meddelanden som skickats och tagits emot, samt uppgifter om antalet hopp som meddelandet gjorde för att nå just den medlemmen. Tidsåtgången för rundsändningen beräknas som skillnaden mellan värdet på den senaste tidpunkten då ett meddelande togs emot för första gången och tidpunkten då den första medlemmen startade spridningen av meddelandet. Som vi noterade tidigare, så är denna tid ej korrekt, utan har påverkats av någon för oss okänd faktor. Den tid vi kommer använda att göra analysen är maximalt antalet hopp gånger fördröjningen i testet. Statistiken skrivs ut till en fil och ett nytt test startas. Efter ett förutbestämt antalet gånger kommer experimentövervakaren och alla medlemmar att stängas av.

7.1.2 Val av programspråk och översikt av experimentprogrammet

Detta avsnitt beskriver valet av programspråk samt börjar beskriva den del då medlemmarna kör en rundsändning.

Som vanligt är det av vikt vilket programspråk vi använder. Vi valde mellan Java, C och C++. Java för dess enkelhet i att skriva objekt och våra egna erfarenheter av att skriva nya kommunikationsprotokoll i det programspråket, C för dess snabbhet och C++ för att det har en kombination av de två tidigare nämnda programspråkens egenskaper. Bäst hade varit om vi kunde skriva allt i Java, men vi var tvingade att välja bort Java, då det är svårt att mäta just tid i Java [5], och att mäta tid är ju ett centralt moment i denna uppsats. Vi skapade ett testprogram i C++ som mätte tiden tio gånger i följd. Tidsupplösningen visade sig vara i

storleksordningen microsekunder. Odisciplinerad C-programmering hade lätt gett för plottrig kod, och dessutom ville vi ha möjligheten att skriva en klient som kunde starta olika protokoll. Valet blev då C++, och vi drog nytta av att vi kunde skriva en abstrakt superklass åt protokollen, som innehöll den största funktionaliteten. Superklassen hade två underklasser. Dessa två underklasser innehåller rutinen som väljer vilka medlemmar i medlemslistan ett meddelande ska spridas vidare till. I RandomizedBroadcast-klassen, som agerar som skvallerprotokollet som beskrivits i avsnitt 4.2, sker detta på ett slumpmässigt sätt, och i ReliableBroadcast-klassen, som fungerar som protokollet för pålitlig rundsändning som beskrivits i avsnitt 4.3, sker detta på ett deterministiskt sätt. Klassernas hierarki visas i Figur 7-2.



Figur 7-2 Schematisk bild över klasshierarkin

7.2 Översikt av kommunikationen mellan processer

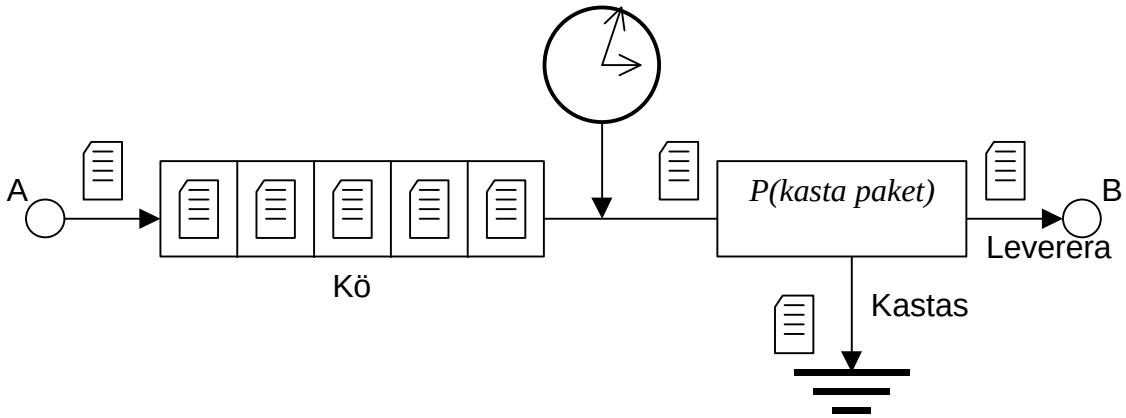
I detta avsnitt beskriver vi hur kommunikationen mellan processer fungerar, var förseningarna skapas och hur förlusterna genereras. Först beskriver vi hur ett meddelande skickas från en medlem till en annan. Sedan tittar vi närmare på mekanismerna som gör att protokollen fungerar, och till sist diskuterar vi olika fall som kan inträffa beroende på om datorn inte är snabb nog, genom att studera de olika fördröjningarna som ett paket kommer att uppleva.

7.2.1 Simulering av förluster och fördröjning

Från början hade vi tänkt testa protokollen på ett lokalt nätverk på Karlstads universitet. Det var i ett sent skede, då vår testmiljö redan var utvecklad, som vi hittade information om storleken på förluster och fördröjningar på Internet, och därför fann behov av att simulera dessa aspekter. Då det var nätverksspel på Internet som intresserade oss bestämde vi oss för att simulera dessa aspekter av Internet. Vi hade då inte tid att leta efter redan utvecklade

nätverkssimulatorer utan valde att implementera förluster och fördröjningar i vår utvecklade testmiljö.

När ett protokoll har tagit emot en information från användaren, kommer protokollet att skapa ett meddelande bestående av bl.a. en unik identifierare för meddelandet. Protokollet kommer sen att skapa en lista av andra medlemmar som den ska skicka sitt meddelande vidare till, och sen skicka meddelandet dit. När meddelandet sen tas emot hos mottagaren levereras inte meddelandet direkt, utan meddelandet tidstämplas och läggs på kö, där det får ligga och vänta tills fördröjningen, som getts genom fördröjningsparametern, har förlöpt. Vid det här laget slänger datorn paketet med den sannolikhet som är den som förlustparametern anger. I Figur 7-3 illustreras detta.



Figur 7-3: Simulering av fördröjning och förluster

7.2.2 Tiden från en medlem till en annan

I detta avsnitt beskrivs först de två moment som kan ge fördröjningar. Sedan diskuteras vad som kan hända när vissa av dessa är för stora. De två saker som kan ge en fördröjning är:

- Otillgänglighet av processorn
- Fördröjning då ett paket ligger i kö

Otillgänglighet av processorn är en fördröjning som vi inte vill ha i vår testmiljö då en stor fördröjning kan leda till att processerna ej kan arbeta ”parallellt” och att den uppmätta tiden kan påverkas av en sådan fördröjning. Om fördröjningen på grund av otillgänglighet av processorn är försumbar jämfört med fördröjningen då ett paket ligger i en kö kommer fördröjningen på grund av otillgänglighet till processorn ej att ge någon märkbar effekt på tiden för rundsändning. Inledande tester visade en processoranvändning på några få procent vilket indikerar att fördröjning på grund av otillgänglighet till processorn inte skulle bli något problem. Detta resonemang tas upp igen i avsnitt 9.3.1 om validitet för testmiljön samt i C.3.

7.2.3 Antal hopp ersätter tid

Som vi tidigare nämnt så visade tidiga experiment att själva tidmätningen lider av dessa fördröjningar som vi inte kan styra över. Därför kommer vi att simulera fördröjningarna, men vi kommer att dra slutsatserna utifrån antalet hopp och resonemang om antal hopp och tid istället för den uppmätta tiden.

8 Resultatredovisning

I föregående kapitel beskrevs den testmiljö som skapats för att utföra experimenten. Detta kapitel redovisar de viktigaste resultaten från experimenten. Nästa kapitel innehåller den analys av datan som behövs för att i sista delen av uppsatsen dra några slutsatser om betydelsen av våra experiment. Ett urval av de testresultat som resultaten bygger på finns i bilaga E. Datan i bilagan har bara till uppgift att exemplifiera resultaten, och är ej komplett eller håller någon statistisk riktighet. Om resultaten från testerna önskas i digital form kan författarna till denna uppsats leverera resultaten.

8.1 Djuptest

I detta avsnitt redovisas de statistiska värden som är av intresse vid senare statistisk beräkning. Exempel på rådata till dessa diagram och tabeller finns i bilaga E. Tabellerna visar avrundade värden. Tiderna visas i millisekunder.

8.1.1 Tabeller

I det här avsnittet redovisar vi medelvärdet för antal hopp ett meddelande gjorde under rundsändning med de båda protokollen och variansen för respektive nätverksstorlek och protokoll. Detta görs i Tabell 8-1och Tabell 8-2.

	Medel (antal hopp)	
Antal Noder	**Pålitlig Rundsändning**	**Skvaller-protokollet**
68	6,16	5,44
105	8,01	6,02
150	9,22	6,30

Tabell 8-1: Uppmätt medelvärde på antal hopp

	Varians (S^2) (antal hopp)	
Antal Noder	**Pålitlig Rundsändning**	**Skvaller-protokollet**
68	0,14	0,30
105	0,01	0,11
150	0,17	0,21

Tabell 8-2: Uppmätt varians på antal hopp

8.2 Trendtest

Trendtestet genererade 180 konfigurationer. Då varje konfiguration testades 10 gånger genererades 1800 resultat. Alla dessa resultat kan omöjligen redovisas i en uppsats. Därför har vi sammanställt medelvärdena för tidsåtgång och antalet hopp i flera tabeller. Det finns två tabeller för varje kombination av storleken på antalet medlemmar och de två protokoll som testats. Detta ger sammanlagt tolv tabeller. Den ena tabellen anger tiden för rundsändning vid olika konfigurationer på förluster och fördröjningar medan den andra tabellen visar medelvärdet för antalet hopp. En tabell innehåller exempelvis den genomsnittliga tiden för rundsändning med skvallerprotokollet och 68 medlemmar, och även tiden vid de olika värden på fördröjningar och förluster som testats. Då antalet tabeller blev många har vi valt att redovisa tabellerna i bilaga G. I avsnitt 9.2 redovisas en jämförelse av tabellerna med samma antal medlemmar i tredimensionella diagram. Dessa diagram är betydligt lättare att studera än resultatet i tabellform vilket är en anledning till att tabellerna är presenterade som diagram.

9 Analys av resultat

I föregående kapitel presenterades resultaten av testerna. I nästa kapitel drar vi slutsatser om hur lämpliga protokollen kan tänkas vara i spelnätverk. En diskussion om vad som kan vara intressant att undersöka vidare återfinns också i nästa kapitel.

I det här kapitlet analyserar vi djuptestet och trendtestet, och drar slutsatser från analyserna. Vi kommer sedan att föra en diskussion om validiteten av resultaten, experimenten och slutsatserna.

9.1 Djuptest

I detta avsnitt beskrivs de beräkningar som gjorts för att kunna analysera djuptestet. Analyser av resultaten som genereras av beräkningarna ges sedan.

9.1.1 Resultat av beräkning av konfidensintervall

Vi använde Formel 5-1 för att beräkna ett konfidensintervall för medelvärdena för antal hopp som ett meddelande som mest behövde göra för att fullborda rundsändningen. Beräkningarna med Formel 5-1 redovisas i Tabell 9-1. I Tabell 9-1 redovisar vi övre (+) och undre (-) gränser vi erhöll vid konfidensintervalltestet. De positiva gränserna visar hur mycket färre hopp skvallerprotokollet behövde för rundsändningen än vad protokollet för pålitlig rundsändning behövde. Det verkliga medelvärdet som anger hur mycket snabbare det ena protokollet är än det andra ligger mellan de båda gränserna med 99 % sannolikhet.

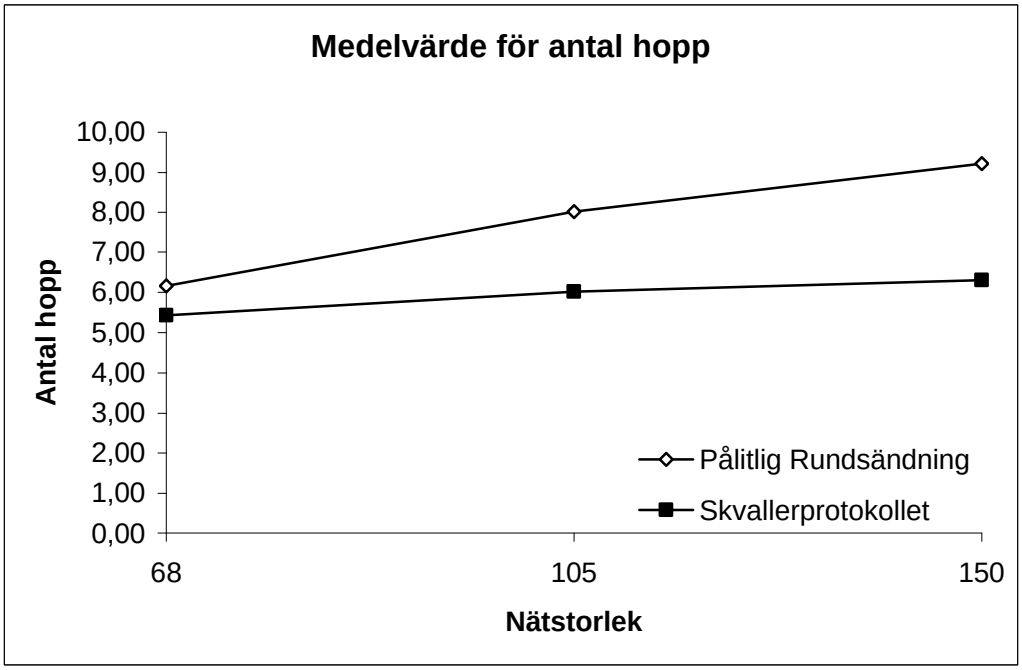
	99% konfidens-intervall	
antal noder	**-**	**+**
68	0,61	0,84
105	1,93	2,06
150	2,81	3,03

Tabell 9-1: Konfidensintervall för skillnaden i antal hopp

9.1.2 Analys av djuptest

Figur 9-1 illustrerar hur medelvärdet för antal hopp vid allmänsändning varierar med antalet noder. Med hjälp av en linjär regression av minus-kolumnen i Tabell 9-1, finner vi att

skillnaden i tid mellan de två medelvärdena för de uppmätta antal hoppen kommer att konvergera vid runt fem noders nätverksstorlek. Att extrapolera information på det här viset är vanskligt, och vi har ingen aning om hur protokollen beter sig vid en mindre nätstorlek. Det är alltså inte säkert att graferna konvergerar vid fem noder eller att skillnaden i tid fortsätter att öka vid fler antal medlemmar. Detta är dock en fråga som borde undersökas i framtiden.



Figur 9-1: Diagrammet visar medelvärdesutvecklingen vid ökat antal noder

I Tabell 9-1 kan man se att skvallerprotokollet var snabbare än protokollet för pålitlig rundsändning oberoende av om man valde 68, 105 eller 150 medlemmar. I Figur 9-2 är de slutsatser för djuptestet som kan dras av Tabell 9-1 sammanställda.

Vid djuptestet visade det sig att pålitligheten för skvallerprotokollet inte var så hög som screeningen visat. I en procent av fallen kom inte meddelandet fram till alla noder. Protokollet för pålitlig rundsändning levererade dock meddelandet i alla tester. Det tyder på att protokollet för pålitlig rundsändning är pålitligare vid 1,2 % paketförluster. Fler tester behöver dock utföras för att säkerställa detta.

Nätverksstorlek	Slutsats
68	Skvallerprotokollet behöver minst 0,61 färre hopp än protokollet för pålitlig rundsändning
105	Skvallerprotokollet behöver minst 1,9 färre hopp än protokollet för pålitlig rundsändning
150	Skvallerprotokollet behöver minst 2,81 färre hopp än protokollet för pålitlig rundsändning

Figur 9-2: Slutsatser vid olika nätverksstorlekar

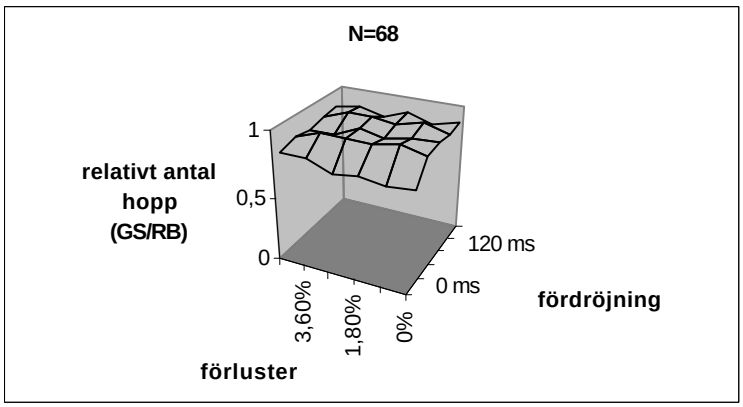
9.2 Trendtest

I detta avsnitt beskrivs de beräkningar som gjorts för att kunna analysera trendtestet. Analyser av resultatet som genereras av beräkningarna ges sedan.

9.2.1 Analys av trendtest

Genom att använda metoden som beskrivits i avsnitt 5.2.2 erhölls tre tredimensionella diagram. Diagrammen kan ses nedan. Notera att axlarna på diagrammen är vända åt olika håll i olika diagram. Det fanns inte ett sätt att vända alla, så att alla diagrammens karakteristik åskådliggjordes. De olika parametrarna vi har testat i detta experiment är från minimivärden till maximivärden som observerats på Internet. Antalet hopp är medelvärdet för det maximala antalet hopp som behövdes för att en nod skulle få meddelandet för första gången.

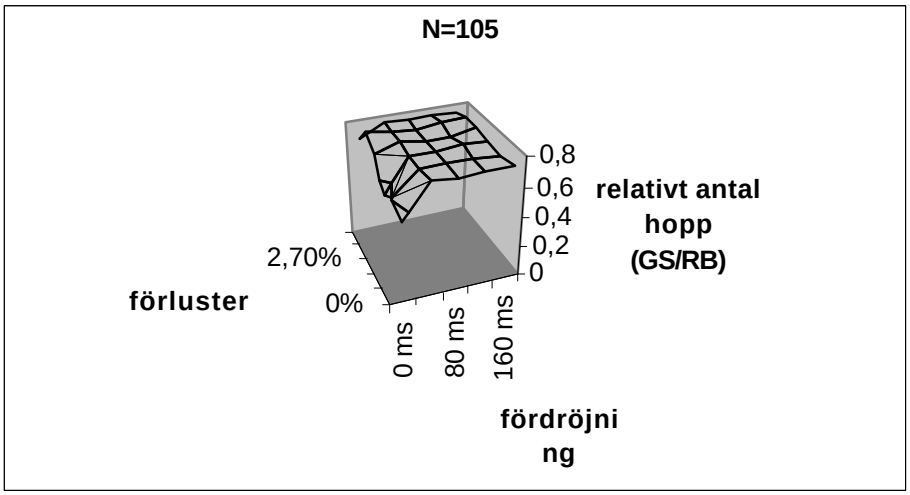
I Figur 9-3: till Figur 9-5, visas det relativa antalet hopp för olika nätverksstorlekar.



Figur 9-3: Relativ skillnad i antalet hopp vid 68 medlemmar

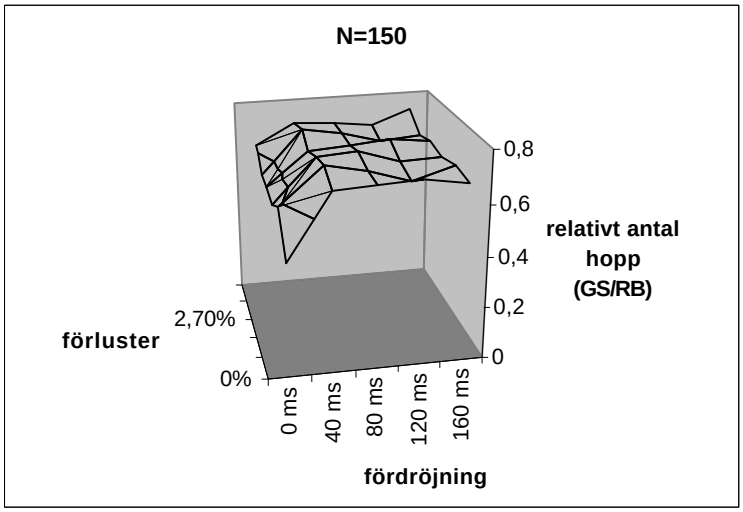
Figur 9-3: visar skillnaden i hopp mellan protokollen då antalet noder är 68. I Figur 9-3: ser vi att antalet hopp är mindre för skvallerprotokollet än för protokollet för pålitlig rundsändning för alla konfigurationer av förluster och fördröjningar. När fördröjningen var noll millisekunder fick vi ett märkligt resultat vilket antagligen berodde på att processorn fick

för mycket att göra och processerna inte fick arbeta ”parallellt”. Protokollet för pålitlig rundsändning fick då av någon anledning ett sämre resultat i antalet hopp räknat jämfört med tester som hade en högre fördröjning. Förutom då fördröjningen är noll millisekunder verkar inte en förändring av fördröjning och förluster påverka skillnaden mellan de båda protokollen.



Figur 9-4: Relativ skillnad i antalet hopp vid 105 medlemmar

Diagram i Figur 9-4 visar resultatet för tester på 105 noder. Liksom i testerna då antalet noder var 68 är resultatet märkligt, då fördröjningen är noll millisekunder. Då fördröjningen ej är noll är skvallerprotokollet snabbare och gör färre antal hopp för alla konfigurationer av fördröjning och paketförluster. Skillnaden verkar vara densamma mellan protokollen oberoende av vilka värden som då väljs.



Figur 9-5: Relativ skillnad i antalet hopp vid 150 medlemmar

Diagram i Figur 9-5 visar resultatet för tester på 150 noder. Liksom i de tidigare testerna då antalet noder var 68 och 105 är resultatet märkligt, då fördröjningen är noll millisekunder. Då

fördröjningen ej är noll är skvallerprotokollet snabbare och gör färre antal hopp för alla konfigurationer av fördröjning och paketförluster. Skillnaden verkar vara densamma mellan protokollen oberoende av vilka värden som då väljs.

Alla diagram liknar varandra. Ingen tydlig lutning finns, och i många fall är ytan väldigt ”bucklig”. Därför kan vi inte dra slutsatsen att skillnaden i tidsåtgång för rundsändning mellan de två protokollen påverkas av förluster och fördröjningar. Dock visar alla diagram att i nästan alla testfall, förutom då fördröjningen är 0 ms på vissa diagram, att skvallerprotokollet tar mindre hopp på sig att nå den sista medlemmen. Vi bortser ifrån att testerna vid 0 ms fördröjning skjuter i höjden och faller till pålitlig allmänsändnings fördel, detta eftersom testmiljön i detta läge troligen uppförde sig på ett felaktigt sätt. En diskussion om orsakerna till detta ges nedan samt i avsnitt 9.3.1 som tar upp testmiljöns validitet.

Som slutsats kan vi säga att detta trendtest indikerar att ett djuptest skulle visa att skvallerprotokollet är snabbare på rundsändning än protokollet för pålitlig rundsändning oberoende av storleken på konstant fördröjning och ett konstant medelvärde på förluster. Detta gäller förutsatt att dessa värden på fördröjning och förluster ligger inom områdena 40 till 160 ms respektive 0 till 4,5 % förluster.

9.3 Validitet

Validiteten för testerna är uppdelad i tre avsnitt. Dessa avsnitt är intern validitet, konstruktionsvaliditet och extern validitet. I intern validitet diskuteras hur rättvis jämförelsen mellan protokollen var, och om den fysiska och den implementerade miljön fungerade som det var tänkt. I konstruktionsvaliditet diskuteras om protokollen som jämförts används i stor utsträckning och om implementationen av protokollen är korrekt. I extern validitet diskuteras valet av statistisk metod och om testmiljön representerar Internet. De tre validitetsavsnitten kan ses nedan.

9.3.1 Intern validitet

Protokollen som testades skickade ett meddelande till lika många grannar. Antalet gånger som skvallerprotokollet skickade ett meddelande vidare valdes efter en screening för att uppnå en hög säkerhet men ändå hålla nere antalet meddelanden. Antalet meddelanden som respektive protokoll genererade var dock lägre för protokollet för pålitlig rundsändning. Om antalet meddelanden som de båda protokollen genererade var mer lika skulle resultaten från testerna kunna bli ett annat. Nedan förklaras varför. Genom att öka antalet medlemmar en medlem känner till med ett, för pålitlig allmänsändning, skulle antalet meddelanden bli mer lika

mellan de båda protokollen. Detta skulle också leda till att varje medlem skulle känna till den medlem som var längst bort i den ursprungliga ring som skapas i en Harary-graf. Detta skulle i sin tur leda till att antalet hopp till den nod som var längst bort från någon nod skulle minska och tiden för att sprida ett meddelande till alla andra noder skulle således, i vår miljö, sannolikt minska och pålitligheten skulle öka. Att lägga till en granne i protokollet för pålitlig rundsändning skulle då kunna leda till att protokollet för pålitlig rundsändning skulle vara snabbare och pålitligare än vad vi uppmätt för ett givet antal meddelanden i vår miljö.

För att testerna skall vara jämförbara krävs att den fysiska miljön var lika vid alla tester. Vid trendtesterna gjordes testerna på tre olika datorer. De tester som gav värden som sedan jämfördes i trendtesterna utfördes på samma dator. Skillnaden mellan protokollen påverkas alltså inte av att resultaten erhölls från olika datorer. Tiderna i de olika trendtesterna skulle däremot kunna påverkas av valet av dator. Dessa datorer hade dock testats i en screening för att säkerställa att de gav samma resultat om samma protokoll kördes med samma inställningar för protokollet på olika datorer. Datorerna körde samma operativsystem och konfiguration och hade samma prestanda. Vid konfidenstestet utfördes testerna på samma dator.

För att tiderna som uppmättes inte skulle påverkas av att det fanns flera medlemmar på en dator krävs det att tiden som ett meddelande väntade i en kö var stor nog. Det visade sig vid testerna att den uppmätta tiden var märkbart större än antalet hopp multiplicerat med fördröjningen. Tiderna som uppmätts skulle därför kunna vara större än de tider som skulle ha producerats om medlemmarna haft tillgång till varsin processor. När antalet hopp jämförs mellan protokollen ser man att antalet hopp är betydligt lägre för skvallerprotokollet än för protokollet för pålitlig rundsändning. Skillnaden mellan protokollen skulle därför sannolikt kvarstå även om varje medlem förfogade över en varsin processor. Om vi studerar diagrammen för trendtestet kan vi se att i de diagram som behandlar tiden finns det en avvikelse då fördröjningen på nätverket sätts till 0 ms. Detta är sannolikt ett resultat av att experimentmiljön inte fungerar väl då det får köra för fullt. Rundsändningen kommer inte att ske parallellt på ett sådant sätt som den borde. Protokollet för pålitlig rundsändning har en högre hopplängd till den sista noden men får ändå en lägre tid. Testmiljöns beteende behöver studeras innan slutsatser om orsakerna till de märkliga tiderna kan dras. Vad som orsakade den förhöjda tiden har vi inte hunnit fördjupa oss i. Det är dock av största vikt att kontrollera detta om testmiljön skall användas fortsättningsvis.

Nätverkstrafiken externt från datorerna har sannolikt inte påverkat testerna, eftersom inga andra program kördes som vi vet belastar nätverket under testerna. Automatiska uppdateringar sker inte under dagtid, vilken var tidpunkten för testerna.

9.3.2 Konstruktionsvaliditet

Protokollen har i litteraturen som sammanfattats i kapitel 3 beskrivits som vanliga och standardiserade protokoll. Protokollen har implementerats på det sätt som de beskrivits i teorin. Dock upptäcktes ett fel i implementationen efter att testerna utförts. En medlem som använder protokollet för pålitlig rundsändning skulle inte skicka ett meddelande till den medlem som meddelandet kom ifrån. Detta hade av misstag förbisetts vid implementationen. Tiden för rundsändning påverkades troligen inte av denna miss då meddelandet redan varit hos den medlem som fick ett extra meddelande. Den processorkraft som går åt för att skicka ett extra meddelande kan dock ha påverkat tiden något. Vidare studier av hur detta påverkat testet behövs för att ta reda på hur de extra meddelandena påverkade testutfallet. Inte heller pålitligheten påverkades av samma skäl. Den Harary-graf vi har använt är dock inte exakt den som beskrivits av Lin et al. Den Harary-graf vi har använt har ett mindre största avstånd. Hur pålitlig den Harary-graf vi använt är har vi ej undersökt, utan en sådan undersökning lämnas åt framtiden. I de tester vi utfört var dock pålitligheten väldigt hög. Det inträffade aldrig att någon medlem ej fick ett meddelande.

Den slump vi har använt bygger på funktioner rekommenderade i Linux. Det är därför sannolikt att denna slump skulle bli den slump som används då ett skvallerprotokoll skapats med Linux som operativsystem.

9.3.3 Extern validitet

Vid en miljö som beter sig som testmiljön är resultaten testade med en konfidensintervallskattning. Tillräckligt många tester utfördes för att variansen skulle kunna uträknas från testdatan. De båda testerna var oberoende av varandra. Den statistiska metod som använts skulle således gälla.

Då vi simulerat Internet istället för att utföra testerna på Internet behövs en diskussion om hur väl vår simulering representerar Internet. Det är osannolikt att varje medlem har samma tidsavstånd till varje annan medlem i stora nät och att detta tidsavstånd ej varierar med tiden. En variation i tiden för sändning av ett meddelande mellan två medlemmar skulle sannolikt leda till, då det finns flera vägar till varje medlem, att tiden till den sista medlemmen skulle bli något bättre än genomsnittet av vägarna till medlemmen. Den bästa vägen, då det finns flera vägar till varje medlem, skulle bli den tid som mäts och tiden för att sprida ett meddelande till alla andra medlemmar skulle sannolikt bli bättre vid variationer av tidsavstånd mellan medlemmar då vissa vägar skulle kunna få flera minskningar i tiden jämfört med en medeltid. Då vi mäter den bästa tiden till den sista medlemmen skulle detta

påverka våra resultat. Samma resonemang skulle då gälla om tiden mellan olika medlemmar var olika istället för ett genomsnitt. Hur mycket de båda variationerna skulle påverka testutfallet beror på hur stor variationen är. Vi har ej kunskap om hur fördröjningar varierar på Internet. Vidare studier om hur sändningstider varierar på Internet behövs för att kunna svara på om en variation i tiden skulle påverka testutfallet.

10 Slutsatser och vidare forskning

I föregående kapitel analyserade vi våra data. En sammanfattning av dessa analyser presenteras först i detta kapitel. Analyserna leder fram till slutsatser om de testade protokollens användning i nätverksspel. Dessa slutsatser presenteras efter sammanfattningen av analyserna. Det sista avsnittet i detta kapitel tar upp de frågeställningar som väckts under arbetets gång, men som inte kunnat studeras närmare. Det avsnittet har vi kallat vidare forskning.

10.1 Sammanfattning av analys

I analysen i föregående kapitel kunde vi dra slutsatsen att skvallerprotokollet var snabbare än protokollet för pålitlig rundsändning både i djuptestet och i trendtestet. Enda avvikelsen från detta resultat var då fördröjningen satts till 0 ms. Då testmiljön verkar ha betett sig på ett märkligt sätt bortser vi från detta resultat. Den anledning vi kunnat se till att skvallerprotokollet är snabbare än protokollet för pålitlig rundsändning i våra tester är att det meddelande som nått fram till den medlem som fått meddelandet sist tagit fler hopp vid användning av protokollet för pålitlig rundsändning. Då det verkar ha varit någon mer fördröjning, än fördröjningen då ett meddelande låg i en kö i väntan på att levereras till en medlem, som påverkat tiden, behövs ytterligare studier av vår testmiljö för att kunna förklara varför tidsåtgången blev som den blev.

10.2 Resonemang om användning av protokollen i spelnätverk

Tiden som uppmättes för rundsändning med de bägge protokollen var över 0,6 sekunder i medel för alla tester i djuptestet. Reaktionstiden för en människa är ungefär 0,2 sekunder. Tiden det tar för ett meddelande att spridas till alla noder skulle då sannolikt vara för hög för

vissa typer av spel. Om spelet bygger på att spelaren skall reagera snabbt på olika händelser skulle de tider vi uppmätt sannolikt vara för höga och spelet skulle kunna uppföra sig hackigt. Det skulle kunna leda till att spelkaraktärer som dör, återuppväcks, då ett meddelande om förflyttning inte kom fram i tid. Om man ökar antalet medlemmar varje medlem känner till i pålitlig rundsändning och ökar antalet medlemmar skvallerprotokollet skickar ett meddelande till, minskas det största avståndet mellan medlemmar. Då skulle sannolikt tiden för rundsändning minskas då ett meddelande kan spridas till fler medlemmar vid ett givet antal hopp. Det skulle därför vara intressant att studera hur andra värden på parametrarna i de båda protokollen skulle påverka sändningstiden för att se om protokollen då går att använda i spel där reaktionstiden hos en människa är en gräns för den maximala tiden för att sprida ett meddelande.

Vissa spel är mer turbaserade, d.v.s. att spelarna agerar i tur och ordning, och andra spel bygger inte så mycket på reaktion. I sådana spel som en liten fördröjning kan accepteras, skulle protokollen kunna användas.

Då skvallerprotokollet användes hände det att ett meddelande inte nådde alla medlemmar. Om ett meddelande inte når alla medlemmar skulle det kunna ge konstiga effekter som att en karaktär är död hos en medlem men levande hos en annan, eller att resultatet i en fotbollsmatch är olika vid olika medlemmar. Om meddelandena innehåller information som bygger på att tidigare information har setts kan förluster bli ett stort problem. Om informationen istället innehåller en absolut uppgift, t ex den exakta positionen en karaktär befinner sig på så skulle ett senare meddelande kunna användas trots att ett tidigare meddelande ej nått fram. Det skulle dock behövas något sätt att maskera att ett meddelande förlorats. Detta är dock ett uppslag för vidare forskning.

10.3 Vidare forskning

I avsnitten nedan kommer de uppslag till vidare forskning, som framkommit under vårt arbete, att presenteras. Pålitlighet, antal meddelanden och tid för rundsändning är alla egenskaper som vi vill undersöka vidare. Vi skulle även vilja ta fram en mer verklighetstrogen bild av Internet samt undersöka vår testmiljö närmare. Alla dessa uppslag beskrivs mer ingående i avsnitten nedan.

10.3.1 Pålitlighet

Om antalet medlemmar ökas så kommer antalet meddelanden att öka. Då vi använt en fast sannolikhet för paketförluster kommer också antalet tappade meddelanden att öka. Det skulle

vara intressant att undersöka om pålitligheten minskar då antalet medlemmar ökas då protokollens parametrar förblir de samma. Om pålitligheten minskar skulle parametrarna för protokollen behöva ökas för att återfå önskad pålitlighet. Ökning av parametrarna leder till ökad nätverkstrafik. Om en ökning av parametrarna behövdes skulle protokollen vara oanvändbara vid riktigt stora nät, då nätverkstrafiken skulle bli alltför hög.

10.3.2 Tester vid samma antal meddelanden

Vi testade bara med en konfiguration på parametrarna. Den valdes för att båda protokollen skulle kunna skicka ett meddelande till lika många medlemmar varje omgång. En jämförelse där de båda protokollen skickade samma antal meddelanden skulle vara en annan rättvis jämförelse. Ett stort test med många olika konfigurationer på parametrarna skulle behövas för att få en förståelse för hur tiden förändras vid ändring av parametrarna.

10.3.3 Ännu större nät

Vi såg i våra tester att tiden för rundsändning ökade då antalet medlemmar ökade. Detta är inte så konstigt då varje medlem endast sprider sitt meddelande till ett visst antal andra medlemmar. Om ökningen i tid är fortsatt linjär även för större nät än de vi har testat är en fråga för framtiden.

De protokoll som beskrivs i kapitel 3 men inte har testats hade alla sina fördelar och skulle tid ha funnits hade alla protokollen testats. Speciellt blev vi intresserade av resurssnålt sannolikhetsbaserat protokoll som var utvecklat för att fungera väl med riktigt många noder. Det var dock bra att testa skvallerprotokollet först för att ha det som referens vid test av resurssnålt sannolikhetsbaserat protokoll då de uppför sig på samma sätt. Att låta skvallerprotokollet skicka ett meddelande till alla medlemmar första gången det skickas skulle sannolikt minska tiden tills alla medlemmar har fått ett meddelande då avståndet i antal hopp till de flesta medlemmar skulle minskas. Hur en sådan konfiguration skulle bete sig skulle vara intressant att testa.

10.3.4 Bättre simulerat Internet

För att kunna testa i en mer realistisk Internet-miljö skulle en studie, av hur förluster och fördröjningar är fördelade och varierar på Internet, behöva göras. Som vi beskrev i avsnitt 9.3.3 skulle länkar med varierande fördröjning sannolikt ge en lägre rundsändningstid. Även länkar med olika fördröjning skulle sannolikt minska rundsändningstiden.

10.3.5 Andra Internet-aspekter som kan simuleras

Som tidigare nämnts är det inte säkert att vi skulle få samma resultat om vi valde en variabel lösning på fördröjningen, men med vilken modell skulle denna fördröjning simuleras? Borde fördröjningen och förlusten ha ett samband? Det här är klart ett område som kräver betydligt mer forskning än vad vi hade möjligheten att göra, och det är absolut inget lätt område.

Naturligtvis är inte förluster och fördröjning de enda företeelserna som kan ha återspeglats från Internet. Andra företeelser som kunnat återspeglas (men som är mycket mer komplicerade i sin natur) är obestämda fördröjningar som resultat av stockning i några routers, den explosionsartade naturen hos trafik på Internet (från engelskans burstyness) och även av en medlems tillgänglighet, om en medlem inte får något meddelande alls under en sändning.

Genom att fokusera på ett nät där inga medlemmar faller bort, och bara koncentrera oss på om enstaka meddelanden faller bort, så kunde vi bortse från tillgängligheten hos medlemmar. Den explosionsartade naturen hos Internet valde att inte simulera, då även detta är ett mycket komplicerat område och vår kunskap inte räckte till för att simulera detta.

10.3.6 Slå samman meddelanden

I ett nätverk där det genereras många meddelanden av alla medlemmar, behövs ett sätt att slå samman meddelanden, detta eftersom nätkapaciteten är begränsad vid varje medlem. Till exempel, i ett nätverksspel med 10 000 deltagare, som spelar ett spel där minst 3 uppdateringar från varje spelare måste skickas till alla andra deltagare varje sekund, betyder varje medlem behöver ta emot 30 000 meddelanden varje sekund! Men, om det finns ett sätt att slå ihop meddelanden från flera medlemmar, kanske det går att minska antalet meddelanden till en bråkdel av detta.

10.3.7 Tidpunkter då medlemmar blir informerade

Det är intressant att studera när alla medlemmar har fått ett meddelande vid de olika rundsändningsprotokollen. Ett protokoll som tidigt i tiden ger många medlemmar ett meddelande skulle göra att många medlemmar kunde använda sig av meddelandet tidigare i tiden och skulle ha en fördel jämfört med protokoll som levererar ett meddelande till många sent i tiden..

10.3.8 Hybrider

Något mycket intressant är att försöka dra nytta av Harary-grafens minimalistiska antal kanter för en given felmodell vilket ger ett lågt antal meddelanden, och samtidigt

skvallerprotokollets fördelar i tidsåtgång för rundsändning. Det här skulle bli något som liknar den resurssnålt sannolikhetsbaserade protokollet men med statiska grannar.

Referenser

[1] A. Demers *Epidemic Algorithms for Replicated Database Maintenance* Proceedings of 6th ACM Symposium on Principles of Distributed Computing Vancouver, Kanada 10-12 Augusti 1987 pp. 1-12

[2] Ambali, A.G., *A demonstration of fast response computer aided control system design too* Proceedings of ICARCV'94. Third International Conference on Automation, Robotics and Computer Vision. Singapore, 9-11 Nov. 1994 (Nanyang Technol. Univ) pp. 122-6 vol.1 database:_INSPEC Copyright 1996, IEE ISBN: 981 00 5793 8

[3] Bjarneby, Tobias. *Super PLAY* Medströms förlag, 2003:4

[4] Indranil Gupta, Kenneth P. Birman och Robbert Van Renesse, John Wiley and Sons *Fighting Fire with Fire: Using Randomized Gossip to Combat Stochastic Scalability Limits*, Ltd, Qual. Reliab. Int 2002; 18: pp. 165-184

[5] Hackman, Mikael och Höglund, Maria Kap 3.2, ”Tidsupplösning” i *Jämförelse av Exekveringstider vid kontraktsprogrammering i Java.*, © 2002 Karlstads Universitet, Datavetenskap. Pub. Num 2002:01.

[6] Kurose & Ross,Kap 1.6: *Computer Networking; A top down Approach Featuring the Interne.*, © 2001 Addison Wesley Longman, Inc. ISBN: 0-201-47711-4

[7] Matrix NetSystems *http://average.matrix.net*, , 2003-04-10

[8] Meng-Jang Lin, Keith Marzullo *Directional Gossip: Gossip in a Wide Area Network* Proceedings of the Third European Dependable Computing Conference Prag, Tjeckien September 1999 (Springer Verlag LNCS 1667), pp. 364-379

[9] Meng-Jang Lin, Keith Marzullo, Stefano Masini *Gossip Versus Deterministically Constrained Flooding on Small Networks*, 2000 Springer Verlag, DISC 2000 LNCS 1914 pp. 253-267

[10] Mullender, Sape *Distributed Systems*, 2nd Ed. Ch.5. sidorna 97-145. Mullender, Sape. Pearson Education ©1993 ACM Press, Tryckt 2002 av Antony Rowe Ltd, Estbourne. ISBN 0-201-62427-3

[11] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, A. M. Kermarrec, P. Kouznetsov *Lightweight Probabilistic Broadcast* Proceedings of the international Conference on Depandable Systems and Networks 2001

[12] Opnix, Inc. Analog X *http://www.internettrafficreport.com*, 2003-04-10

[13] Towsley, Don. Dept. of Computer Science, University of Massachusetts. *Network Tomography through End-to-End Measurements* A.L. Buchsbaum and J. Snoeyink (Eds.): ALENEX 2001, LNCS 2153, p. 60, 2001. © Springer-Verlag Berlin Heidelberg 2001

[14] Vännman, K. *Matematisk statistik* Studentlitteratur 1990

A Ordlista

Ord/Term	Engelska/Beskrivning
(Protokollet för) Pålitlig pålitlig rundsändning	Reliable Broadcast på engelska
Icke-oordning	Aanti-entropy på engelska
LAN	Local area network = lokalt nätverkLAN = Lokalt nätverk
Nät med icke-hierarkisk struktur	peer-to-peer-nätverk
Resurssnålt sannolikhetsbaserat protokoll	Leightweight Probabilistic Broadcast
Riktat skvaller	Directional Gossip på engelska
Rundsändning	Broadcast.Att skicka något meddelande till alla som förmår att ta emot det. Broadcast på engelska
Ryktesspridning	Rumor mongering på engelska
Skvaller (protokollet)	Gossip
Tid för rundsändning	Den tiden det tar från det att någon börjar skicka ett meddelande tills tidpunkten då den sista som förmådde ta emot detta meddelandet tog emot detta meddelandet för första gången.
WAN	Wide area network = WAN = globalt nätverk

B Avgränsningar

Vid arbetets början hade vi en vision om att ta fram en formel som gav den bästa logiska topologin, med avseende på tidsåtgång för rundsändning, för olika distribuerade beräkningar. Formeln skulle bygga på de tidpunkter som medlemmarna i systemet beräknades vara klara med sin del av beräkningen. Dessa tidpunkter skulle kunna liknas vid någon fördelning, t ex en normalfördelning. Om man visste vilken fördelning en beräkning hade skulle man kunna få fram den bästa logiska topologin för just den beräkningen. Vi tänkte att tiden för en medlems tidsåtgång för en beräkning kunde vara $T = T_{väntapådata} + T_{beräknadata} + T_{distribueraresultat} + T_{terminationdetection} + T_{deadlockdetection}$. Tiden för att slutföra en beräkning tänkte vi representera med $\max(T_1, T_2, ..., T_n)$ Efter en tid insåg vi att det inte var så enkelt som vi först hade trott. Våra kunskaper var alldeles för begränsade för att inta ett holistiskt perspektiv på distribuerade beräkningar. Vi bestämde oss då för att ett atomiskt perspektiv var det bästa och vi började avgränsa vårt projekt.

Till att börja med delade vi upp en distribuerad beräkning i fem olika delar. Dessa delar var:

- Dela upp en beräkning i delar
- Distribuera delberäkningar
- Utföra delberäkningar
- Distribuera delresultat
- De som skall ha delresultaten har fått dem

Vi studerade sedan litteratur inom de olika delarna för att komma fram till var forskningen står idag. Det område vi fattade intresse för var spridning av ett meddelande med hjälp av ett skvallerprotokoll, då det fungerade väl för att hålla databaser konsistenta och databaser hade en del gemensamma karaktärsdrag med nätverksspel. Detta område föll inom ramen för att distribuera delresultat, nämligen att distribuera ett resultat från en till många medlemmar. Vi ville först testa hur detta protokoll betedde sig då många noder skulle starta en resultatspridning vid samma tidpunkt, men vi bestämde oss sedan för att endast iaktta spridning från en till många medlemmar. Spridning från många till många medlemmar

samtidigt skulle sannolikt bete sig som en multipel av spridning från en till många medlemmar. Detta har vi dock inte hunnit testa.

Vi började sedan dela upp spridning av resultat i ytterligare delmoment. Denna uppdelning var inte beroende av vilken algoritm vi använde. Vi fick då fram följande lista med delmoment:

- Delberäkning är klar
- Vilka är intresserade
- Logiska topologin
- Spridningstopologi sett från en nod
- Fysiska topologin
- Spridningsalgoritm

Som tidigare nämnts så bestämde vi oss för att testa spridning av en medlems information till alla andra medlemmar. Spridningsalgoritmen bestämde i de flesta fall vilken logisk topologi som skulle användas. En fråga vi ställde oss var hur tiden skulle påverkas om man buntade ihop flera meddelanden i ett meddelande när de träffades vid en medlem. Detta skulle kunna ge att mindre data skickades på nätet då overheaden för de hopbuntade meddelandena skulle kunna minskas. Då vi hade bestämt oss för att testa spridning av ett meddelande gick vi aldrig djupare in på denna frågeställning.

Spridningsalgoritmen skapar en spridningstopologi vid varje spridning av ett meddelande. Spridningstopologi är alltså vägen ett meddelande färdas på genom systemet av medlemmar. Vi funderade på om det skulle gå att skapa en generell spridningstopologi för skvallerprotokollet, då skvallerprotokollet bygger på slumpen, vilket kan ge olika spridningstopologier vid varje ny spridning. Vår begränsade tid gjorde dock att vi fick välja bort denna fråga. Det var framförallt tiden att sprida ett meddelande som var viktig i nätverksspel.

Då en medlem skickar ett meddelande så gör medlemmen oftast det till flera andra medlemmar. I vilken ordning medlemmarna valdes ut för att sända ett meddelande till skulle påverka tiden var en annan fråga vi hade. Skulle man tjäna på om sändningsordningen var randomiserad istället för sekventiell. Som nämnts i 2.6.5 valde vi att endast undersöka fall då sändningstiden är försumbar. På grund av det så fördjupade vi oss inte i hur sändningsordningen skulle påverka tiden tills alla medlemmar fått ett meddelande.

Den fysiska topologin skulle vi inte kunna göra så mycket åt, utan ta den som fanns i de tillgängliga datasalarna på Universitetet. Det fanns också för få datorer tillgängliga i de

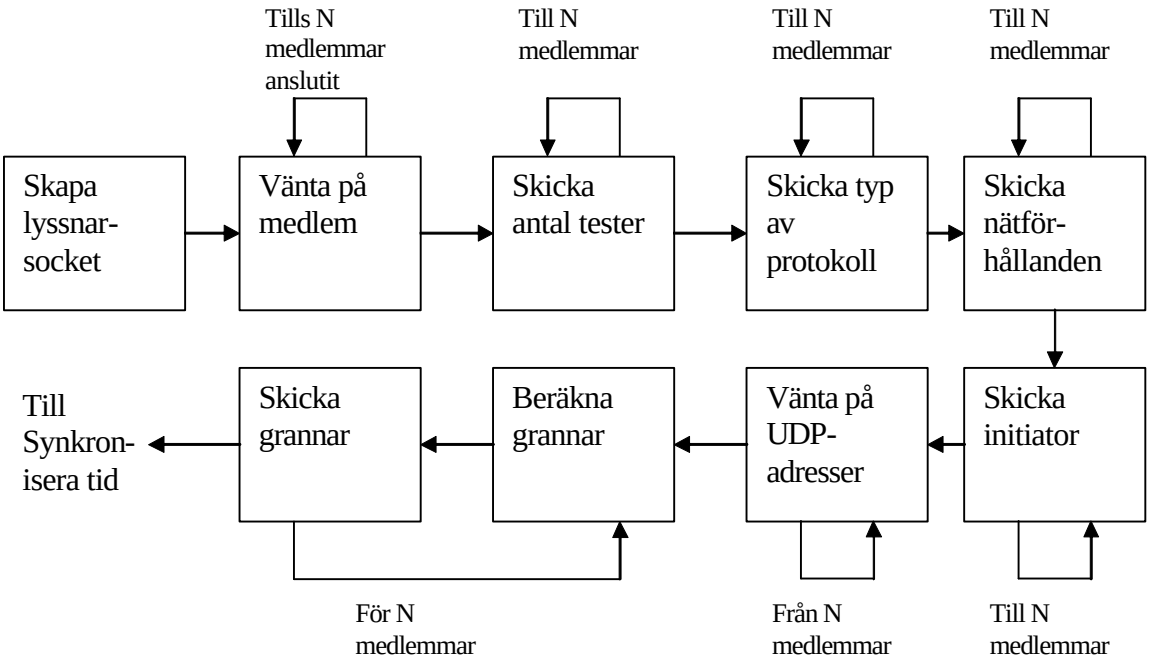
datorsalar där vi kunde kontrollera nätverkets beteende. Vi skulle därför behöva köra flera medlemmar på samma dator. När vi testade hur mycket processorkraft som gick åt för att köra våra testprotokoll upptäckte vi att flera hundra noder kunde köras på en dator utan att processorutnytjandet översteg några få procent. Detta tydde på att experimenten inte skulle påverkas av att alla noder samsades om samma processor. Vi bestämde oss då för att simulera ett nätverk på en dator. I de artiklar vi läst var sådana simuleringar vanliga.

Då vi läste mer om skvallerprotokollet fann vi att skvallerprotokollet blivit jämfört med protokollet för pålitlig rundsändning vid ett tillfälle [9] vid nätverk med upp till 64 noder och olika antal noder som kraschade vid sändningens början. Det var dock främst protokollens pålitlighet som jämförts, så vi bestämde oss för att göra en djupare analys av hur lång tid det tar från att ett meddelande börjar spridas till dess att alla fått det, och därmed var vår avgränsning klar. Nu hade vi fått något som var överskådligt och då vi sedan tidigare hade kunskaper om protokollet för pålitlig rundsändning, skulle studier av skvallerprotokollet ge oss det mesta av den information vi behövde för jämförelsen. Vid dessa studier hittade vi flera variationer av skvallerprotokollet. De avgränsningar som då gjordes kan ses i kapitel 4.

C Testmiljön

Denna bilaga beskriver hur testmiljön är designad. För att få största behållning av denna bilaga skall 7.1.1 läsas först. Synkroniseringen av tid har inte använts i testerna då alla medlemmar befann sig på en dator. Informationen om hur tidssynkroniseringen går till finns dock med om vår utvecklade testmiljön någon gång skall användas på flera datorer. Sist i detta kapitel finns också en beskrivning av hur testmiljön fungerade i våra tester.

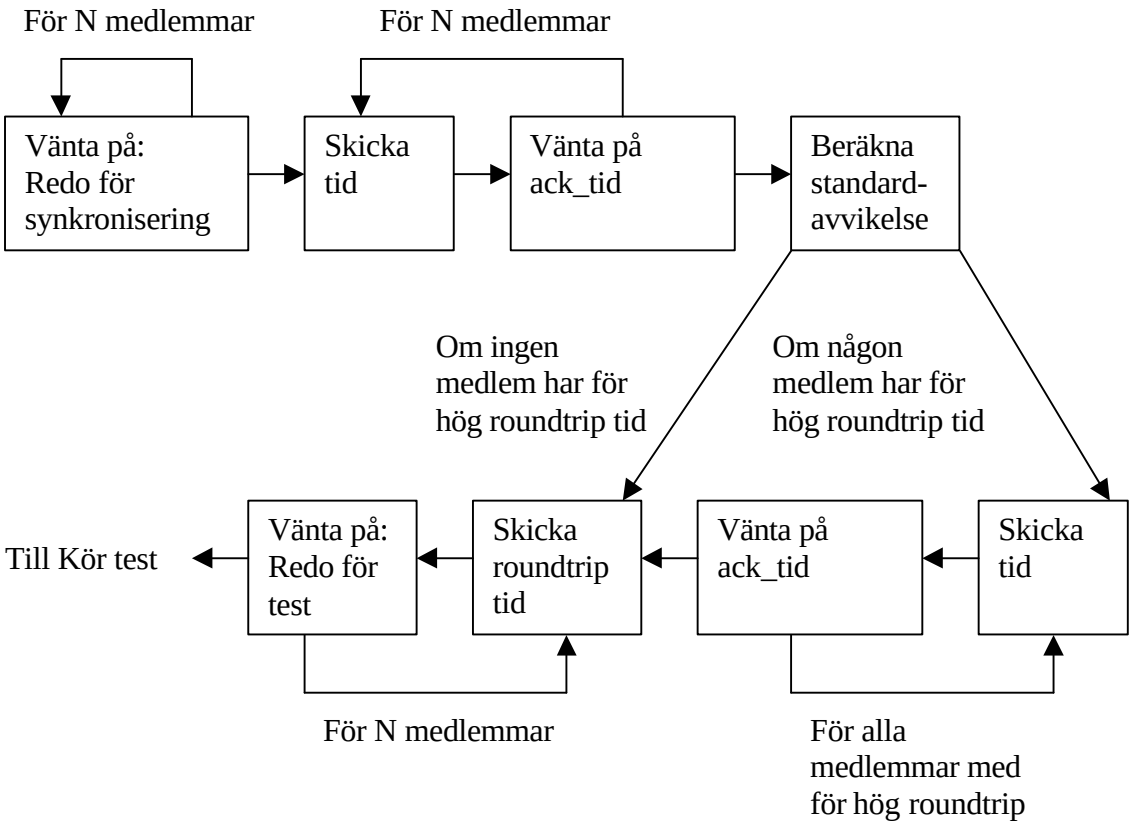
C.1 Experimentövervakare



Figur C-1: Experimentövervakarens uppstartsfas

I Figur C-1 visas hur experimentövervakaren sätter upp ett test. Först väntar experimentövervakaren på att N (N bestäms vid uppstart av experimentövervakaren) antal medlemmar skall ansluta. När N medlemmar anslutits så meddelas alla medlemmar om hur många tester som skall utföras och vilket protokoll som skall testas. Alla medlemmar får också ett meddelande beträffande vilken medlem som skall starta testet. Sedan inväntar experimentövervakaren de UDP-adresser som medlemmarna kommer att använda sig av i testet. Om protokollet som skall testas är pålitlig rundsändning, så beräknas medlemmarnas

grannar med den modifierade algoritmen för att skapa en Harary-graf. Skall skvallerprotokollet testas så skall alla medlemmar ha alla andra medlemmar som grannar. Sedan meddelas medlemmarna om hur många och vilka grannar de skall ha i testet. Meddelandet innehåller UDP-adressen till respektive granne.



Figur C-2 Experimentövervakarens synkroniseringsfas

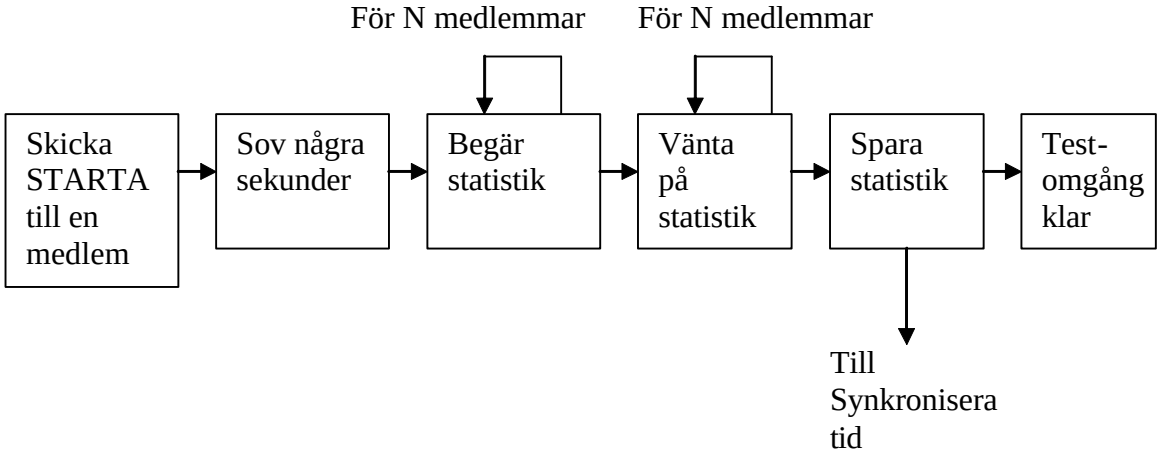
För att medlemmar skall kunna få en tidtagning som stämmer överens med experimentövervakarens tid så synkroniseras medlemmarnas klockor med experimentövervakarens klocka innan varje test. Denna synkronisering sker med hjälp av fyra meddelanden och visas i Figur C-2. Experimentövervakaren inväntar ett meddelande från varje medlem som säger att medlemen är redo att synkronisera sin klocka. Experimentövervakaren inväntar svar från alla medlemmar innan den börjar synkronisera medlemmarna så att inga meddelanden skall störa synkroniseringen. Experimentövervakaren tar först sin tid och skickar tiden till en medlem. Sedan inväntar experimentövervakaren svar från medlemmen. När experimentövervakaren fått svar tar experimentövervakaren sin tid igen. Experimentövervakaren kan då räkna ut roundtriptiden. Detta förfarande upprepas för varje medlem. En största tillåtna roundtrip tid räknas sedan ut med formeln $md + 2 * s$. Denna

formel har tagits ifrån boken Matematisk statistik [14]. S är standardavvikelsen och beräknas med formeln

$$\sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right)}$$

Formel C-1: Beräkning av största tillåtna roundtriptid

Alla medlemmar som har för hög roundtriptid får ett meddelande om detta. Experimentövervakaren inväntar svar från den medlem som just meddelats om den för höga roundtriptiden. Sedan försöker experimentövervakaren synkronisera medlemmens klocka igen på samma sätt som ovan. Denna omsynkronisering upprepas till dess att roundtriptiden är mindre än eller lika med den tillåtna. Varje medlem som har för hög roundtrip tid får omsynkronisera sin klocka. När experimentövervakaren har uppmätt en godtagbar roundtrip tid för varje medlem så meddelas klienterna om sina respektive roundtrip tider. Sedan inväntar experimentövervakaren ett meddelande från medlemmarna om att de är redo att starta testet.

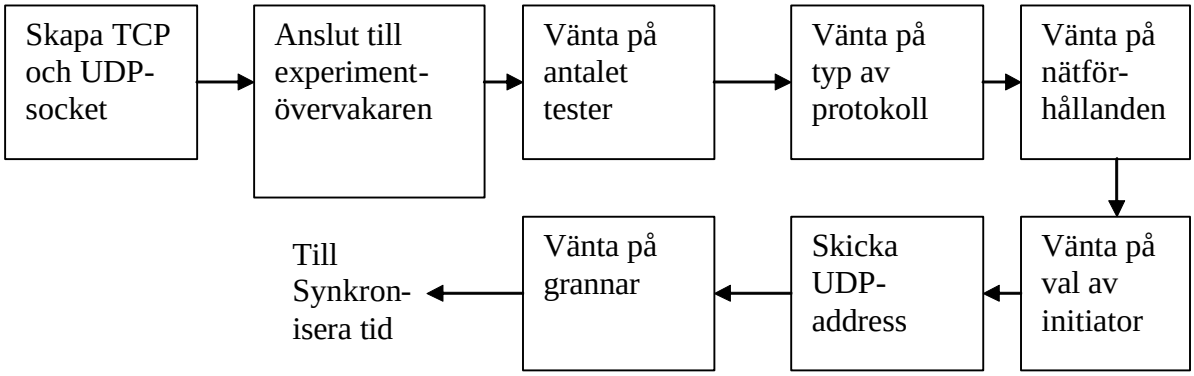


Figur C-3: Experimentövervakarens sammanställningsfas

I Figur C-3 visas faserna ”kör testet” och ”spara statistik”. Till den medlem som är initiator av testet skickas meddelandet ”starta”. Experimentövervakaren sover sedan i några sekunder för att inte påverka testet genom att begära statistik från medlemmarna. När experimentövervakaren kommit igång igen hämtas statistiken från alla medlemmar. Statistiken består av de tidpunkter medlemmarna fick det första meddelandet, antalet skickade och mottagna meddelanden vid varje medlem och hur många hopp ett meddelande gjort på väg till varje medlem. Experimentövervakaren sammanställer statistiken, d v s beräknar tiden mellan början av testet (initiatorns tidpunkt) och den senaste tidpunkten för mottagandet av ett

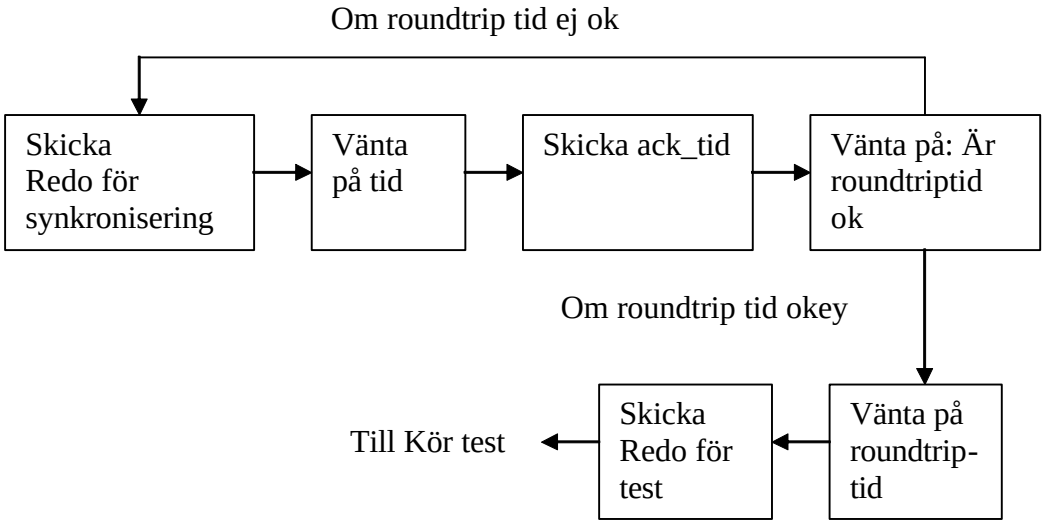
meddelande vid någon medlem. En graf över när ett visst antal medlemmar har nåtts av meddelandet skulle kunna framställas då experimentövervakaren har tidpunkterna för mottagande av meddelande hos varje medlem. Dessa data fick vi dock inte tid att analysera. Alla medlemmars skickade och mottagna antal meddelanden räknas samman. När statistiken är sparad görs ovanstående förfarande om i antal gånger från och med tidssynkroniseringen.

C.2 Medlem



Figur C-4: Medlemens uppstartsfas

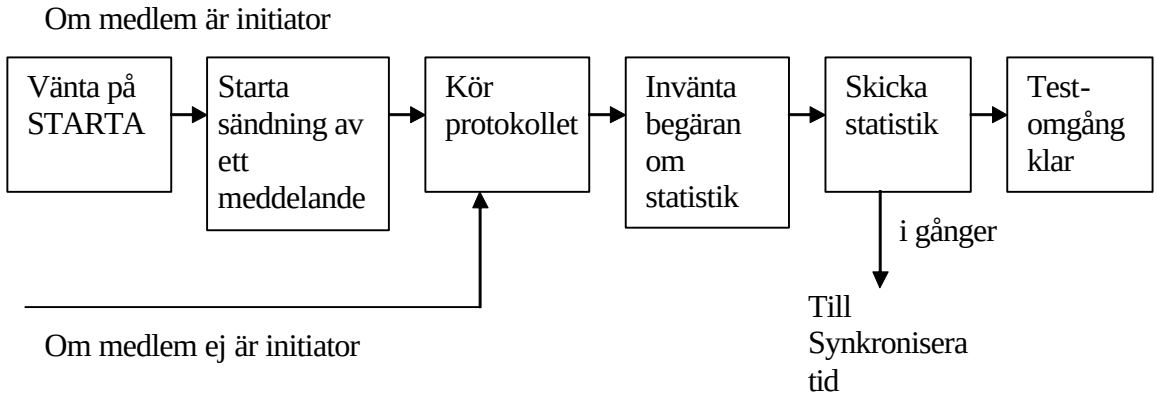
Medlemmarna är ansvariga för att genomföra testerna. Figur C-4 visar hur en medlem förbereder sig för ett test. Först skapar varje medlem en UDP-socket och en TCP-socket. Sedan ansluter medlemmarna sig till experimentövervakaren. De blir sedan meddelade om hur många tester som skall göras, vilket protokoll som skall testas, nätförhållandena för testet (fördröjning och paketförluster) och vilken medlem som skall skicka det första meddelandet i testet. Sedan skickar varje medlem sin UDP-adress till experimentövervakaren. experimentövervakaren meddelar då medlemmarna vilka grannar de skall ha i testet.



Figur C-5: Medlemmens synkroniseringsfas

I Figur C-5 visas hur en medlem synkroniserar sin klocka gentemot experimentövervakarens klocka. Medlemmen meddelar experimentövervakaren om att den är redo att synkronisera sin klocka. Experimentövervakaren meddelar då medlemmen om sin tid. Medlemmen tar sin tid och skickar ett ack till experimentövervakaren. Anledningen till att medlemmen tar sin tid innan ack skickas är att man då för med medlemmens tidtagning i roundtriptiden och kan räkna bort halva denna. Medlemmen får veta om roundtriptiden var okej eller inte. Var tiden för hög görs synkroniseringsproceduren om. När medlemmen får ett meddelande om att roundtriptiden låg nedanför den tillåtna roundtriptiden väntar medlemmen på ett meddelande om vad roundtriptiden blev. Medlemmen beräknar nu skillnaden mellan medlemmens tid och experimentövervakarens tid. Genom att dela roundtriptiden med två och dra bort denna tid från skillnaden fås en mer exakt tidsskillnad. När tidsskillnaden är uträknad förbereder sig medlemmen för testet. Sedan meddelas experimentövervakaren om att medlemmen är redo för testet.

All kommunikation mellan medlemmarna och experimentövervakaren sker med TCP. Det finns dock möjlighet att utföra tidssynkroniseringen med UDP för att få en mer exakt tid. Risken med UDP är att paket kan tappas och detta skulle få medlemmen och experimentövervakaren att vänta i evighet på det förlorade paketet. Då vi ansåg oss ha tillräckligt med processorkraft för att ha alla medlemmar på en dator, behövde vi inte använda oss av tidssynkroniseringen, då alla medlemmar redan har samma tid, men funktionen beskrivs ändå om den skulle behöva användas vid senare experiment.



Figur C-6: Medlemens testfas och rapporterinsfas

I Figur C-6 beskrivs hur en medlem kör testet och rapporterar statistiken. Om medlemmen är initiator väntar den på ett startmeddelande från experimentövervakaren. När detta fås tar experimentövervakaren sin tid och startar sedan en rundsändning med det protokoll som har valts. Om medlemmen ej är en initiator startas rundsändningsprotokollet direkt utan att invänta något meddelande från experimentövervakaren. Medlemmen tar sin tid vid den tidpunkt då medlemmen först får ett rundsändningsmeddelande. När testet är avklarat väntar medlemmen på att experimentövervakaren skall be om statistik. När experimentövervakaren ber om statistik, så skickar medlemmen den tidpunkt då meddelandet togs emot för första gången, modifierad tidsdifferansen mellan medlemmen och experimentövervakaren. Medlemmen skickar också antalet skickade och mottagna meddelanden till experimentövervakaren. Medlemmen väntar sedan åter på att experimentövervakaren skall synkronisera sin tid med medlemmen. Ovanstående förfarande upprepas i antal gånger från och med tidssynkroniseringen.

C.3 Testmiljöns beteende

Att beräkna tiden för rundsändning verkar ganska enkelt. En första anblick säger att det inte borde vara mer än att multiplicera tiden för fördröjningen mellan medlemmarna med antalet hopp som varje meddelande tagit. Vi studerar ett fall och ser vad som händer.

Låt oss säga att vi har 150 noder, inga förluster och en fördröjning på 80 ms vid användning av protokollet för pålitligt rundsändning, som skickar meddelanden över en Harary-graf, på det sätt som vi beskrivit tidigare. Först beräknar vi hoppstorleken som programmet borde använda vilken beskrivs i Formel 4-3.

$$m = \lfloor \sqrt{N} \rfloor = \lfloor \sqrt{150} \rfloor = 12$$

För att nu beräkna antalet hopp ett meddelande som mest kommer att göra vid rundsändning (i medeltal) använder vi Formel 4-4.

$$h = \frac{N}{2m} + \frac{m}{2} - \frac{m}{4} = \frac{150}{2*12} + \frac{12}{2} - \frac{12}{4} = 9{,}25$$

Alltså, ungefär 9,25 hopp kommer varje meddelande göra. Och med en fördröjning på 80 ms borde alltså tiden för rundsändning i ett sådant nät bli ungefär

$$t_{allmänsändning} = 9{,}25 * 80 = 740ms$$

I Figur C-7 återfinns avrundade värden till närmaste 10-tal ur G.1.3 i bilaga G. Dessa värden är från tester med protokollet för pålitlig rundsändning och 150 medlemmar.

		förluster					
		0,0%	**0,9%**	**1,8%**	**2,7%**	**3,6%**	**4,5%**
fördröjning	**0**	100	100	110	90	90	100
(ms)	**40**	690	700	700	710	720	720
	80	*1070*	1070	1100	1130	1120	1110
	120	1440	1490	1460	1490	1500	1490
	160	1800	1820	1820	1860	1890	1860

Figur C-7: Uppmätta tider för ett trendtest

Det kursiverade värdet i tabellen är det verkliga värdet för det ovanstående beräknade värdet. Uppenbarligen är 1070 ms inte i närheten av 740 ms. Alltså måste det finnas något mer som gör att rundsändningen går långsammare.

Skulle vi upprepa denna beräkning för varje fördröjning, och sen dividera den beräknade fördröjningen med den verkliga fördröjningen vid varje punkt, får vi tabell Figur C-8.

		fördröjning					
		0%	**0,90%**	**1,80%**	**2,70%**	**3,60%**	**4,50%**
förluster	**0**	0,00	0,00	0,00	0,00	0,00	0,00
(ms)	**40**	0,53	0,53	0,53	0,52	0,51	0,52
	80	0,69	0,69	0,67	0,66	0,66	0,67
	120	0,77	0,75	0,76	0,75	0,74	0,74
	160	0,82	0,81	0,81	0,79	0,78	0,80

Figur C-8: Beräknade tider för ett trendtest

Genomgående ser vi att den beräknade tiden är lägre än den uppmätta, men att vid högre fördröjningar så blir skillnaden i tid för rundsändning allt mindre. Här följer ett resonemang till att försöka förklara dessa värden.

Det finns alltså en fördröjning som inte beror på att meddelandet ligger i en kö i väntan på att tiden för fördröjningen skall passera. Låt oss kalla denna tid för $t_{ofrivillig}$. Vidare kan vi se att tiden ökar något när vi ökar antalet förluster, låt oss kalla denna tid för $t_{förluster}$. Om detta vore sant, skulle en mer korrekt formel för beräkning av tiden bli

$$t_{allmänsändning} = h * t_{hopp} + c_1 t_{ofrivillig} + c_2 t_{förluster}$$

Om c_1 och c_2 kanske, förutom att innehålla en konstant, också är relativa till förluster och fördröjningar. I det här fallet kan vi bryta ut dessa två värden ur både c_1 och c_2. Vi får då en ny formel

$$t_{allmänsändning} = h * t_{hopp} + (c_{ofrivillig} + c_{förluster})(c_3 t_{ofrivillig} + c_4 t_{förluster})$$

Denna formel, om den är riktig, är långt ifrån lika enkel som den formel vi använde för att beräkna tiden för rundsändning till att börja med. Vi gjorde ett 50-tal försök med att försöka bestämma dessa konstanter, på numerisk väg, men vi lyckades inte. Tydligt är att vår testmiljö behöver studeras vidare för att få en förståelse för vad som orsakade de fördröjningarna i vår testmiljö.

D Protokollöversikt över Internet-stacken

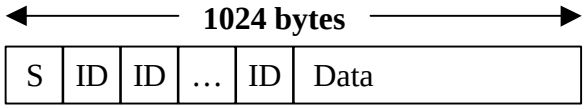
Här följer storleken på den overhead som tänkt skickas med varje paket då testerna först var planerade att utföras på ett lokalt nätverk. Dock utfördes testerna på en dator och därmed gick ingen trafik på nätet. De lager som ett meddelande består av visas i Figur E-1. Nedan följer en beskrivning av lagren i Figur E-1.

Figur E-1: Internetstacken

D.1 Applikationslagret

De tester som utförts har 1024 byte information i applikationslagret. Då skvallerprotokollet ger att en nod som får ett meddelande skall känna till vilka noder som tidigare haft detta meddelande, så kommer de tidigare nodernas id-nummer att finnas här. Det kommer att börja med en siffra som talar om antalet id-nummer som finns i meddelandet. Applikationssegmentet visas i Figur E-2. Om varje nod har 1 byte långt id-nummer kommer listan med id-nummer att ha en största storlek 1 * antal noder. Då antalet noder blir stort kan denna lista uppta en stor del av meddelandet. Det är detta sätt vi har valt. Då alla noder vet om alla andra noder och nodernas respektive id-nummer, skulle 1 bit räcka för att representera om en viss nod fått ett meddelande eller inte. Genom att varje nod har en sorterad lista över nodernas id-nummer har alla noder alla andra noder på samma plats i den sorterade listan. Vid användning av skvallerprotokollet med partiell kunskap skulle en sorterad lista över alla noder se olika ut vid olika noder. Det skulle alltså inte gå att representera en nod med en bit. Men då

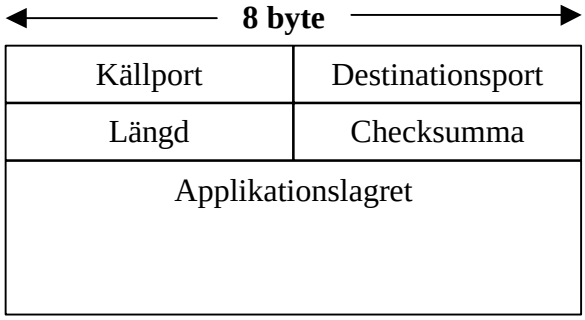
listan vid varje nod aldrig överstiger en förutbestämd storlek, så kommer ej heller ett meddelandes lista att bli större än denna förutbestämda storlek, om man endast lägger in de noder man själv har i listan.



Figur E-2: Applikationslagret

D.2 Transportlagret

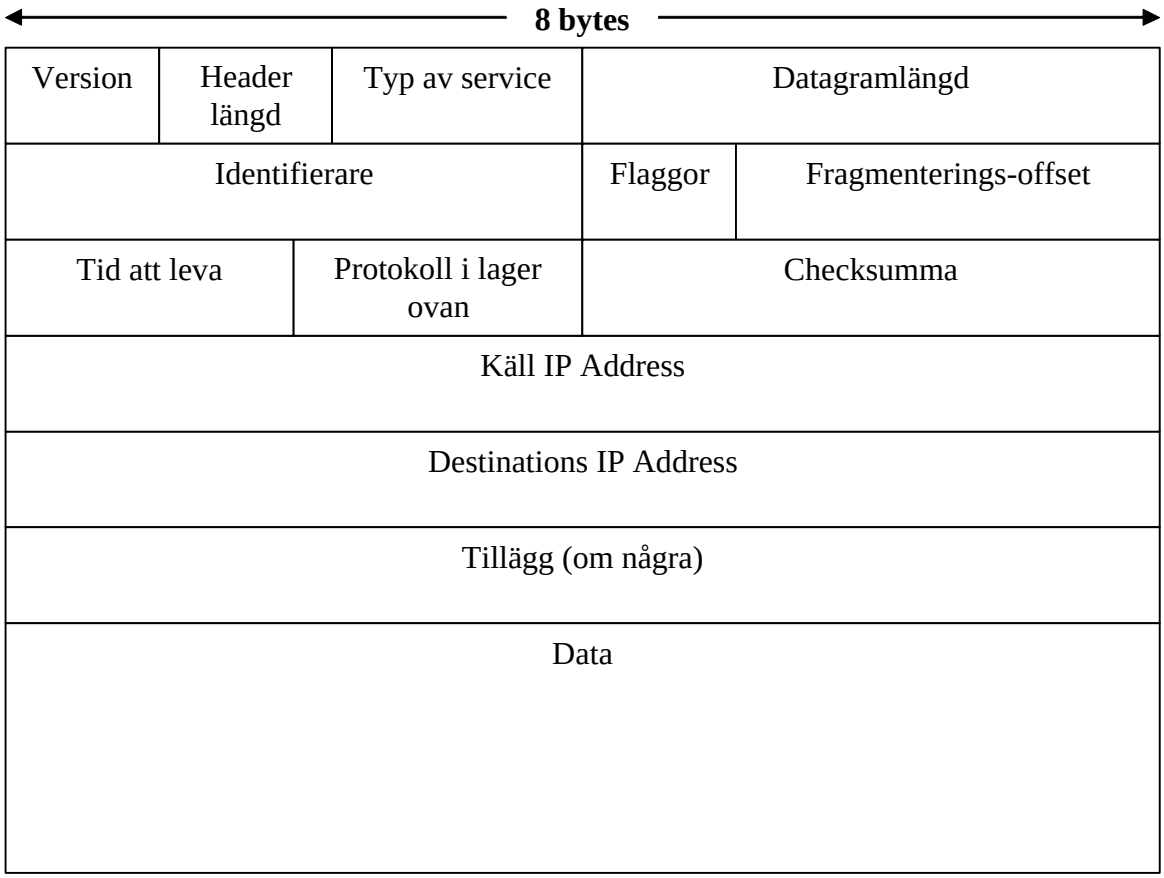
Som transportprotokoll använde vi UDP. Ett UDP-segment visas i Figur E-3 och har fem fält. Overheaden som kommer av UDP är 16 byte.



Figur E-3: Tjänstelagret

D.3 Nätverkslagret

Ett IPv4 segment har olika längd beroende på vilka tillägg man använder. Vi använde inga tillägg och fick då storleken 40 bytes + datan från de övre protokollen. Ett IPv4 segment visas i Figur E-4.



Figur E-4: Nätlagret

D.4 Länklagret

Ett Ethernet-pakets overhead är 20 bytes. Ett Ethernet-paket visas Figur E-5. Datafältet i ett Ethernet-paket kan vara upp till 1500 byte. Blir datafältet större än 1500 byte delas paketet upp i flera Ethernet-paket. Om datans storlek understiger 46 byte så kommer datafältet att utökas med extra bytes för att storleken på datafältet skall bli 46 bytes.

Introduktion	Dest. Address	Käll Address	Typ	Data	CRC

Figur E-5: Länklagret

E Internetstatistik

Paketförlusterna på Internet är i genomsnitt 1,10 procent enligt Matrix [7] (uppmätt under perioden 13 mars, 2003 – 10 april, 2003) och 1,30 procent enligt Opnix [12] (uppmätt under perioden 10 mars, 2003 – 9 april, 2003). För att mäta dessa siffror använder Internet Traffic Report sig av flera servrar runt om i världen som skickar flera paket under några sekunder till en och samma router och inväntar sedan ett svar. De kan sedan räkna ut roundtriptiden och paketförlusterna. Dessa data sammanställs sedan på deras webbsida. Internet Traffic Report har inte några servrar i Afrika eller på Antarktis. Dessa kontinenter representeras ej i Matrixs tester. Matrix utför sina tester på ett liknande sätt och har även servrar som mäter tiden i Afrika.

Denna siffra har historiskt sett varit betydligt högre. Mellan 1993 och 1998 låg paketförlusterna mellan 20 och 30 procent. Under 1999 började denna siffra att sjunka och under 2001 låg paketförlusterna i genomsnitt på 5,5 procent.

Både Matrix och Internet Traffic Report mäter också roundtriptiden för de paket som servrarna får svar på. Denna roundtriptid låg mellan 40 och 200 ms med ett medelvärde på 80 ms enligt Matrix (uppmätt under perioden 13 mars, 2003 – 10 april, 2003) och medelvärdet på Internet enligt Internet Traffic Report var 100 ms (uppmätt under perioden 10 mars, 2003 – 9 april, 2003). Enskilda kontinenter t.ex. Asien låg betydligt sämre till än de övriga världsdelarna, både beträffande roundtriptid och paketförluster. Europa och Nordamerika hade de bästa värdena.

F Stickprover till djuptestet

I den här bilagan redovisar vi exempel på stickproven till djuptestet. Vi redovisar här bara 7-8 slumpvis utvalda stickprover av totalt 200 från varje tabell. Datan är hämtad direkt ur filerna varpå konfidensintervallet skapades. Kolumnerna är förfluten tid i µs (tid), Antal sända meddelanden (sända), Antal mottagna meddelanden (Mgna), Max första antal hopp (Max F.) och Max max antalet hopp (Max Max). Den första kolumnen anger tiden från då den första noden började sända meddelandet till dess att den sista noden tog emot meddelandet för första gången. De två efterföljande kolumnerna kan användas för att verifiera att förlusterna varit de vi ville. Max första betyder det maximala antal hopp som ett meddelande tog för att tas emot för första gången av någon nod. Max max finns bara för testet med 32 noder, och betyder det maximala antalet hopp som ett meddelande någonsin gjorde under det testet. Om ordet ”incomplete” förekommer efter en rad betyder det att i denna omgång fick inte alla noderna meddelandet.

Tabellerna i sin helhet kan fås på begäran av författarna. Patrik kan förmodligen nås på sidan www.kirtap.se.

F.1 Protokollet för pålitlig rundsändning

F.1.1 68 medlemmar

Förluster 0.012
Fördröjning i ms 80

Tid	Sända	Mtgna	Max F.
704183	272	270	6
718393	272	269	6
729299	272	268	6
732347	272	269	6
839057	272	270	7
705690	272	267	6
727748	272	271	6
704752	272	271	6

F.1.2 105 medlemmar

Förluster 0.012
Fördröjning i ms 80

Tid	Sända	Mtgna	Max F.
936370	420	412	8
945378	420	416	8
936610	420	418	8
948597	420	415	8
915320	420	414	8
938230	420	413	8
922885	420	415	8

F.1.3 150 medlemmar

Förluster 0.012
Fördröjning i ms 80

Tid	Sända	Mtgna	Max F.
1095409	600	593	9
1103887	600	597	9
1063698	600	591	9
1115785	600	593	10
1075245	600	594	9
1100079	600	589	9
1052837	600	591	9
1078810	600	592	9

F.2 Skvallerprotokollet

F.2.1 68 medlemmar

Förluster 0.012
Fördröjning i ms 80

Tid	Sända	Mtgna	Max F.
577910	612	609	5
678486	612	604	6
573732	612	608	5
565929	609	601	5
651225	612	603	5

575517	609	606	5	
564810	600	591	5	Incomplete

F.2.2 105 medlemmar

Förluster		0.012	
Fördröjning i ms		80	
Tid	Sända	Mtgna	Max F.
659881	945	932	6
710822	945	938	6
645920	942	931	6
692856	945	930	6
648866	942	925	6
655761	942	928	6
666439	942	926	6

F.2.3 150 medlemmar

Förluster		0.012	
Fördröjning i ms		80	
Tid	Sända	Mtgna	Max F.
675452	1350	1327	6
691790	1347	1334	6
702354	1344	1326	6
672148	1347	1334	6
770472	1350	1335	7
769467	1344	1330	7

G Stickprover till trendtestet

I denna bilaga finns medelvärdet för vardera av de tio testerna som utfördes för varje konfiguration av protokollet.

G.1 Protokollet för pålitlig rundsändning

G.1.1 68 medlemmar

tid(µs)		förluster					
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	55 292	58 100	60 437	59 793	66 659	65 116
	40	480 546	469 627	476 432	488 767	471 990	502 168
	80	715 110	729 326	729 627	750 129	764 318	805 325
	120	968 541	969 899	956 532	998 161	999 273	1 010 899
	160	1 203 619	1 207 245	1 225 527	1 237 060	1 256 095	1 286 020

hopp		förluster					
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	11,0	11,3	11,3	11,2	10,7	11,2
	40	6,1	6,0	6,1	6,4	6,2	6,6
	80	6,0	6,1	6,1	6,2	6,4	6,8
	120	6,0	6,0	6,0	6,2	6,2	6,4
	160	6,0	6,0	6,1	6,2	6,2	6,4

G.1.2 105 medlemmar

tid(µs)							
		förluster					
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	68 782	67 520	69 376	71 346	87 420	85 616
	40	594 083	590 912	580 807	595 708	618 626	609 437
	80	938 599	931 994	947 992	952 272	948 540	959 039
	120	1 259 682	1 265 069	1 256 671	1 266 671	1 267 269	1 292 974
	160	1 585 523	1 580 994	1 579 422	1 601 059	1 604 803	1 579 454

hopp							
		förluster					
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	16,1	15,0	17,3	13,1	12,3	13,0
	40	8,0	8,0	8,0	8,0	8,0	8,2
	80	8,0	8,0	8,0	8,1	8,1	8,2
	120	8,0	8,0	8,0	8,0	8,0	8,2
	160	8,0	8,0	8,0	8,1	8,1	8,0

G.1.3 150 medlemmar

tid(µs)							
		förluster					
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	95 177	95 695	109 262	93 798	94 241	95 165
	40	694 092	698 993	696 704	714 897	720 163	716 037
	80	1 071 333	1 071 333	1 104 530	1 127 264	1 122 359	1 112 216
	120	1 436 648	1 486 985	1 461 843	1 488 940	1 503 087	1 493 793
	160	1 798 636	1 823 612	1 822 802	1 864 311	1 885 450	1 855 114

hopp							
		förluster					
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	21,9	18,0	19,5	18,5	19,1	17,3
	40	9,1	9,3	9,2	9,5	9,6	9,4
	80	9,0	9,0	9,3	9,5	9,5	9,5
	120	9,0	9,4	9,2	9,4	9,5	9,4
	160	9,0	9,2	9,1	9,4	9,5	9,3

G.2 Skvallerprotokollet

G.2.1 68 medlemmar

	tid(µs)					
		förluster				
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	75 256	83 888	89 210	85 784	93 965	96 861
	40	386 389	409 160	384 857	408 942	410 298	403 658
	80	622 487	631 814	615 894	639 303	599 454	622 178
	120	820 173	852 803	845 312	881 144	884 388	866 054
	160	1 066 326	1 016 745	1 083 247	1 024 873	1 088 231	1 074 915

	hopp					
		förluster				
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	8,1	8,2	8,5	8,1	8,6	9,0
	40	5,4	5,7	5,5	5,8	5,7	5,6
	80	5,5	5,5	5,3	5,6	5,3	5,5
	120	5,4	5,7	5,4	5,7	5,7	5,5
	160	5,5	5,2	5,6	5,3	5,6	5,5

G.2.2 105 medlemmar

	tid(µs)					
		förluster				
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	99 086	100 550	99 725	100 181	98 754	110 212
	40	424 755	426 972	424 394	434 828	443 174	442 333
	80	676 909	669 220	675 569	678 448	670 164	694 014
	120	928 411	915 534	929 492	917 935	921 860	947 735
	160	1 168 089	1 131 948	1 148 808	1 220 854	1 207 190	1 160 096

	hopp					
		förluster				
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	8,7	9,1	9,2	9,3	9,6	8,6
	40	6,1	6,0	6,0	6,2	6,1	6,2
	80	6,0	6,0	6,0	6,0	6,0	6,1
	120	6,1	6,0	6,1	6,0	6,1	6,2
	160	6,1	5,9	6,0	6,3	6,3	6,0

G.2.3 150 medlemmar

	tid(µs)						
		förluster					
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	134 296	131 316	136 324	139 617	132 109	132 108
	40	455 329	478 498	448 555	476 644	470 834	473 946
	80	716 889	712 635	742 688	743 933	741 364	758 053
	120	980 492	946 643	955 222	1 021 448	957 444	986 289
	160	1 190 611	1 237 938	1 209 770	1 260 313	1 220 486	1 304 780

	hopp						
		förluster					
		0%	0,90%	1,80%	2,70%	3,60%	4,50%
fördröjning (ms)	0	9,8	10,6	10,5	10,2	10,4	10,7
	40	6,3	6,8	6,5	6,5	6,9	6,6
	80	6,3	6,3	6,7	6,6	6,6	6,6
	120	6,4	6,1	6,2	6,7	6,2	6,3
	160	6,2	6,4	6,2	6,5	6,3	6,8