# Scalable Source Routing

Martin Koskinen

Åbo Akademi

Tekniska Fakulteten

Handledare: André Kaustell

## Referat

Noderna i ett hemautomationsnätverk består till största delen av olika sensorer och andra apparater med begränsad beräkningskapacitet och ett begränsat minne. För att minska på antalet kablar görs ofta sensorer trådlösa. Nätverkstopologin i ett nätverk där olika trådlösa delar binds ihop med kabel kan vara svårdefinierad. Idag ökar antalet trådlösa mobila enheter och då sådana enheter ansluts till nätverket måste ruttningsalgoritmen klara av mobila enheter väl. Dessa faktorer gör det besvärligt för traditionella ruttningsalgoritmer att skapa bra rutter i nätverket eftersom det kan vara svårt att utläsa en klar heriarki.

*Scalable Source Routing*, i fortsättningen SSR, är en ruttningsalgoritm som med hjälp av peer-to-peer teknik gör det möjligt att effektivt skapa rutter i en svårdefinierad nätverksstruktur. SSR behöver inte heller mycket minne för att spara ruttningstabeller.

## Innehållsförteckning

# 1 Inledning

Med ruttning inom datavetenskapen avses att hitta en rutt i ett nätverk. Det finns flera sätt destinationen kan väljas. Unicast används när målet är en viss nod. Multicast används när destinationerna är en eller flera noder. Anycast väljer den närmaste eller bästa destinationen ur en mängd.

För att detta skall vara möjligt krävs att noderna i nätverket har kunskap om var andra noder i nätverket befinner sig, samt att de klarar av att upptäcka noder i sin fysiska närhet. Målet är att hitta en så kort rutt mellan noderna som möjligt, men i de flesta fall kan en viss avvikelse accepteras.

När noderna införskaffar information om nätverket, krävs att en del meddelanden genereras. I regel vill man hålla detta meddelandeutbyte vid ett minimum för att inte belasta nätverket i onödan. Detta blir särskilt viktigt när noderna har begränsade resurser, exempelvis i ett hemautomationsnätverk med sensorer och andra enheter med låg prestanda.

Det lanseras hela tiden nya produkter med nätverksstöd för våra hem. Med tiden kommer detta innebära att våra hemmanätverk växer i storlek. Ju flera trådlösa enheter som kopplas till nätverket desto större är sannolikheten att nätverkstoplogin går från att vara välstrukturerad mot en mesh-struktur.

I OSI modellen har nätverkslagret huvudansvaret för vidarebefordran av paket till rätt mottagare.  Även om det även sker på andra nivåer så är det i detta lager rutterna beräknas.

# 2 Ruttningens grunder

I detta kapitel tar jag upp de allmänna problemställningarna som finns vid ruttning av data i ett nätverk. Jag tar även upp grunderna i datanätverk, och hur dessa är uppbyggda.

## 2.1 Noder

Ett datanätverk byggs upp av noder och anslutningar mellan dessa. En nod består av hårdvara och mjukvara som har till uppgift att vidarebefordra datatrafik till rätt mottagare.

## 2.2 Protokollstacken och abstraktioner

I nätverksmjukvara är det naturligt att införa lager som alla erbjuder tjänster och skapar en abstraktion av hårdvaran samt av de undre lagren. Lagren ger även nätverksmjukvaran en modulär design där delar kan lämnas bort eller bytas ut mot nya efter behov.

OSI modellen[1] är en referensmodell som består av sju lager där de första tre lagren är implementerade på alla noder. Det första lagret nerifrån, fysiska lagret har hand om den egentlige överföringen av bitar. Data Link lagret, består av nätverksadaptrar och drivrutiner som körs i nodens operativsystem. Detta lager samlar ihop bitar till ramar, som sedan sänds åt något håll i stacken. Nätverkslagret som har till uppgift att växla paket mellan noderna i paketväxlade nätverk. Övriga lager behöver inte finnas på alla noder utan är till för trafikgenerering i värdarna.

## 2.3Ruttningsmetoder

Det finns ett flertal metoder för att bestämma ett pakets rutt i nätverket. Ruttningsalgoritmerna som används idag kan delas in i grupper beroende på hur de upptäcker rutterna.

### 2.3.1Reaktivt

När rutter upptäcks reaktivt skickas paketet ut till alla fysiska grannar. Varje nod lägger till sin adress till paketets rutt, varefter det igen skickas ut till alla fysiska grannar. När paketet når sitt mål innehåller det en komplett rutt från sändare till mottagare. Ifall ett svar skall sändas tillbaks behöver inte nätverket översvämmas igen utan ruttningsinformationen från det ursprungliga meddelande kan användas. På detta sätt genererar inte mera överflödig datatrafik än nödvändigt. I nätverk med få noder och låg belastning är denna metod effektiv då den genererar en kort rutt men mycket onödig trafik som belastar nätverket.

[1] Open Systems Interconnection

### 2.3.2Proaktivt

Proaktiv paketväxling betyder att varje nod upprätthåller sin egen kompletta ruttningstabell över alla noder i nätverket. Noderna utbyter med jämna mellanrum tabeller, vilket ger dem möjlighet att räkna ut en optimal rutt på förhand. Datamängderna som genereras är betydande men mindre än med en reaktiv metod. Denna metod har nackdelen att den är långsam vid förändringar i nätverksstrukturen, eftersom det tar en stund innan varje nod hunnit uppdatera sin ruttningstabell.

### 2.3.3Hybrid

Hybridruttning är som namnet anger en kombination av den proaktiva och den reaktiva metoden. I detta fall används översvämning av nätverket för att bygga upp en ruttningstabell, som sedan används när paketen skall skickas.

Nicklas Beijar beskriver i [7] en typ av hybrid där nätverket delats upp i zoner. När den sändande noden är i samma zon som mottagaren används en reaktiv metod, medan en proaktiv metod används för mottagare i andra zoner.

## 2.4 Skalbarhet

Det är mycket viktigt att en ruttningsmetod skalar väl. Faktorer som spelar in är bland annat mängden kontrollmeddelanden när en nod ansluter till nätverket och hur informationen om denna nod sprids till de övriga noderna. Om den inte gör det så får nätverket stora prestandaproblem när antalet noder överstiger den övre gränsen för vad algoritmen klarar av.

## 2.5 Mobilitet

Med dagens allt mer avancerade mobila teknik är det viktigt att ruttningsalgoritmen klarar av rörliga enheter. En mobil nod skall i bästa fall inte ens märka av att dess trådlösa uppkoppling flyttats från en basstation till en annan. Traditionella ruttningsalgoritmer klara inte av detta särskilt väl, utan anslutningen bryts och en ny måste skapas.

## 2.6 Nätverkstopologier

De vanligaste fysiska nätverkstopologierna är stjärn-, bus- och mesh-nätverk. Dessa fysiska topologier beskriver de verkliga kopplingarna mellan noderna,

medan de logiska topologierna beskriver längs vilka rutter paketen skickas i nätverket.

## 2.7 Datalänkar

En datalänk är det medium som binder ihop noderna. Denna fysiska länk används för att sända data över.

### 2.7.1 Kabel

Kabel används både för korta och långa avstånd. För stora datatyper och längre avstånd används oftast en fiberkabel. För kortare avstånd, exempelvis på kontoret, används typiskt en CAT-5 kabel. Vilken typ av kabel som används beror på vilken teknik som används.

Kabelanslutningar är inte lika utsatta för yttre störningar som trådlösa anslutningar. Fiberkablar är i princip immuna mot störningar från elektromagnetiska fält eftersom data sänds som ljusimpulser. Kabelanslutningar har högre säkerhet än trådlösa anslutningar eftersom de är svårare att avlyssna. Den största fördelen kabelanslutningar har jämte trådlösa är pålitligheten. Kabelanslutningar har i regel en högre datakapacitet än en trådlös anslutning.

Inom hemautomationen använder sig idag flera tekniker av befintliga elkablar för dataöverföring. Detta pågrund av kostnaderna att dra nya kablar.

### 2.7.2 Trådlös

Trådlösa länkar sig vanligen av någon radioteknik för dataöverföring. Det finns flera olika trådlösa standarder, exempelvis Bluetooth (802.15.1) och Wi-Fi (802.11).

Trådlösa enheter kan lätt bilda ett mesh-nätverk då de inte har några begränsningar i vart signalerna sänds, eller vilken nod som tar emot dem. Detta medför att det i teorin vore möjligt för två trådlösa noder att kommunicera över en direkt länk. Den vanliga modellen där noderna är anslutna till en basstation tillåter dock inte detta, utan all trafik måste gå via basstationen.

Trådlösa länkar tillåter en viss mobilitet, dock inte mera än att signalen från basstationen är tillräckligt stark. I ett mesh-nätverk är mobiliteten störst, eftersom det räcker med att noden har kontakt med en annan nod.

Dessa länkar är att föredra när nätverk installeras i gamla utrymmen därför att installationskostnaderna kan hållas nere när få eller inga nya kabelförbindelser behöver skapas. Överföringshastigheten som är lägre än över en kabellänk, är beroende av signalstyrkan, och minskar med avståndet från basstationen.

# 3 Scalable Source Routing

I nätverk där topologin varken är hierarkisk eller Unit Disk, vanligen på grund av att trådlösa anslutningar kombineras med anslutningar över kabel.[1] I sådana omgivningar får ruttningsmetoder där varje nod har en komplett ruttningstabell över nätverket, exempelvis Dijkstra och Bellman-Ford, problem eftersom ruttningstabellerna skulle bli stora. *Flooding*, som ger optimala rutter i Ad-hoc nätverk, har för mycket overhead för att vara effektivt i stora nätverk med liten bandbredd.

*Scalable Source Routing* är en ruttningsalgoritm som möjliggör både minnes och meddelandeeffektiv ruttning i nätverk av denna typ. Detta eftersom ruttningstabellen är spridd över alla noder. Varje nod vet nätverkets struktur baserad på dess logiska position i nätverket och inte beroende på nodens fysiska position. På detta sätt sprids informationen över hela nätverket. [1]

## 3.1 Grundbegrepp

I följande kapitel behandlar jag begrepp som används i SSR. Deras innebörd och vilken funktion de har.

### 3.1.1 Logiska Ringen

Skalbar källruttning placerar varje nod i en logisk cirkelformad nätverkstopologi. Varje nod har en logisk adress som fungerar som positionsindikator. Denna logiska ring tar ingen hänsyn till nodernas adress eller placering i nätverket. Den logiska adressen kan vara av tillverkaren

tilldelade nummer, slumpade nummer eller haschar av kryptografiska nycklar.

### 3.1.2 Logiska grannar

Två noder är grannar i den logiska topologin om det inte finns någon nod, med en logisk adress, som skulle vara mellan dessa två noders logiska adresser. För att denna jämförelse skall vara möjlig krävs det att vi kan mäta logiska avstånd i den logiska topologin.

### 3.1.2 Källrutt

En källrutt är den rutt som avsändaren specificerar i pakethuvudet. För att avsändaren skall kunna definiera källrutten som skall användas, krävs att denne har tillräcklig vetskap om nätverkets struktur för att kunna generera källrutten. Källrutten kan även utökas medan paketet är på väg mot sin destination. Reaktiva ruttningsmetoder fungerar på detta sätt.

## 3.2 Algoritm

Detta kapitel handlar om hur SSR fungerar. Jag kommer att förklara hur noderna hittar sina logiska grannar och hur en källrutt skapas iterativt, ifall inte den sändande noden vet hela rutten. Här tas även säkerhetsaspekten i beaktande.

### 3.2.1 Hur hitta logiska grannar

När en ny nod ansluter sig till nätverket hittar den sin fysiska granne genom att sända ett `Hello` meddelande, eller om det är möjligt med hjälp av någon *Link Layer* mekanism. Noderna lyssnar även efter dessa meddelanden. När noden upptäckt en granne så lagras den ett hopp långa källrutten i nodens *cache* minne.

Varje nod väljer individuellt vilken nod i *cache* minnet som skall vara nästa och föregående granne i den logiska topologin. Detta sker genom att noden sänder ett `NeighborNotification` meddelande åt den valda noden. Noden som valts svarar antingen jakande på frågan eller meddelar om en bättre kandidat via ett `NeighborUpdate`. Detta meddelande innehåller en källrutt från den ursprungliga noden till den nya kandidaten. Eftersom den först tillfrågade noden måste känna till den nya kandidaten, så har den en

källrutt lagrad. Från dessa två källrutter kan en ny rutt skapas. Processen som noderna använder för att identifiera sina logiska grannar, vare sig de är nästa eller föregående, skiljer sig inte på andra punkter än hur noden jämförs med den egna logiska adressen.

Denna metod ger i de flesta fall en globalt konsekvent logisk ring, dock kan fel uppstå. För att undvika dessa finns metoder som tas upp närmare i kapitel 3.3.

### 3.2.2 Ruttningsuppgiften

Ruttningsuppgifter beskriv enligt följande av K. Kutzner, C. Wallenta, and T. Fuhrmann i [3]. Före nätverket är i ett är i ett globalt konsekvent tillstånd kan det inte garanteras att ett ruttningsfösök lyckas. För att en nod skall kunna rutta ett meddelande kräv det att den känner till källrutterna till sin nästa logiska granne. Förutom detta måste noderna kunna mäta logiska och fysiska avstånd i nätverket. De fysiska avstånden mäts i SSR genom att räkna hopp i källrutten. Medan de logiska avstånden baserar sig på skillnader i logiska adresser.

Ifall en nod sedan tidigare känner till rutten till den mottagande noden så skapas hela rutten från nodens cache minne. Om detta inte är fallet kan en källrutt skapas iterativt genom att skicka meddelandet till en mellanliggande nod. En mellanliggande nod utökar i sin tur källrutten så att paketet närmar sig sitt slutgilltiga mål. Mellanliggande noder väljs ur sändarens ruttningstabell enligt följande regler.

1. Den mellanliggande noden skall vara virtuellt närmare målet än den sändande noden. Detta betyder att paket alltid sänds medsols i den virtuella cirkeln.
2. Av dessa noder väljs de noder som fysiskt är närmast den nuvarande noden.
3. Av dessa noder väljs den nod som är närmast slutdestinationen i den virtuella topologin.

### 3.2.3 Säkerhet

Från början var SSR konstruerat att lita på all information som noderna får via kontrollmeddelanden. På grund av detta designval var metoden sårbar för attacker från falska noder. Illasinnade noder kunde lätt modifiera nod adresser och därmed ruttningsinformationen som användes. Detta ger skadliga noder möjlighet att undvika trafik och därmed spara energi. Attacken ger även möjlighet till vidare attacker som att sniffa trafik eller helt ta bort den. Vidare kan nätverket störas genom en *denial of service* attacker. Genom att införa ett system med ett privat/publik nyckelpar säkerställs nodernas identifierare av en central auktoritet och problemet med felaktiga nodpekare elimineras helt.[3]

## 3.3 Garanterade av global konsekvens

För att garantera att den logiska ringen är globalt konsekvent måste varje nod känna till noden med högst logisk adress. Från början föreslogs en metod, ISPRP[2], där den nod som tror sig ha den högsta adressen floodar nätverket med denna information. När en nod märker ett fel kan detta rättas till genom att informera noderna om valt fel logiska grannar om detta. Denna metod löser problemet men skapar mycket kontrolltrafik. [5]

Som alternativ till denna metod har det föreslagits en lineariseringsmetod som eliminerar floodingen från ISPRP.

André Kaustell ger i sitt Diplomarbete [2] ett förslag där informationen om en representant sänds tillsammans med `Hello` meddelandena. Även denna metod eliminerar behovet av att flooda nätverket. Dock måste en nod låsa rutten till representanten i sin cache för att kunna lägga med den i `hello` meddelanden. Noden behöver inte spara rutten till representanten längre än att dens grannar godtagit denne. Efter detta räcker det med att representantens adress sparas.

## 3.4 Prestanda

Simuleringar gjorda av Thomas Fuhrman [4] visar att källrutterna som SSR ger upphov till kan vara 2,5 gånger längre än den kortaste rutten, just efter

---

[2] *Iterative Successor Pointer Rewiring Protocol*

att nätverket stabiliserats. Efter ett tag så blir dock rutterna kortare och stabiliseras på 1,2 gånger längre.

Vidare har Thomas Fuhrman med simuleringar visat att det räcker med att varje nod har utrymme för 255 noder i sin *cache* för ett nätverk med över 100 000 noder.

### 3.5 Användningsområden

Hemautomationen idag har ingen heltäckande standard utan det finns några vanligare tekniker. Dessa är tyvärr inkompatibla med varandra och det behövs gateways mellan de olika nätverken. Dessutom har enheter i hemautomationsnätverk ofta enklare uppgifter och har därför begränsade resurser för att minska kostnaderna. SSR kräver inte att alla noder har en komplett bild av nätverket, detta minskar avsevärt på minnesanvändningen på de enskilda noderna.

SSR fungerar i nätverkslagret vilket betyder att befintliga applikationer inte behöver modifieras för att kunna använda sig av tekniken. Dock krävs det att SSR implementeras i nätverkshårdvaran.

## 4 Avslutning

### 4.1 Egna Tankar

I och med att noderna håller reda på både sin logiska föregångare och logiska efterföljare samt den högsta logiska adressen, så kunde man tänka sig att rutter kunde skapas åt båda håll. På det sättet kunde algoritmen eventuellt ytterligare förbättras. I och med att den logiska ringen är glest populerad blir det svårt för noderna att avgöra vilken nod som utgör gränsen för halvcirkeln, men det borde vara möjligt att göra en grov uppskattning baserat på hur stor den största logiska adressen är.

Reglerna för hur en mellanliggande nod väljs när en källrutt skapas iterativt medför att rutten redan efter de första hoppen kommer att nå långt fram i den logiska topologin. Pågrund av dessa faktorer är det inte troligt att en sådan förändring skulle ge en nämnvärd prestandaökning åt algoritmen.

## 4.2 Avslutande diskussion

SSR har många egenskaper som gör den lämplig för användning i ett hemautomationsnätverk. Den logiska ringtopologin ger en bra abstraktion av den faktiska nätverkstopologin, som i ett hemautomationsnätverk inte alltid behöver vara välorganiserat.

Genom att kunskapen om nätverkets struktur inte behöver finnas på varje nod så blir topologiska förändringar lättare att hantera. Detta medför att algoritmen är rätt flexibel när det gäller förändringar i en nods fysiska position. Om en nod försvinner ur nätverket kommer paket adresserade till den noden att levereras till noden som är närmast.

SSR har även egenskapen att vissa noder kommer att samla på fördelaktiga rutter som binder ihop olika delar av nätverket. Dessa noder kan sägas vara knytpunkter.

Givetvis finns det även faktorer som gör SSR mindre lämpligt för användning i hemautomationsnätverk. Till dessa hör bland annat energiförbrukningen. I och med att ruttningsbördan är distribuerad över alla noder så kräver det att noderna konstant är uppkopplade. Detta leder till att energiförbrukningen är betydligt högre i ett sensornätverk som använder SSR än om en ruttningsalgoritm som är anpassad till denna typ av nätverk används. Energiaspekten blir speciellt viktig när det finns batteridrivna noder, vilka är vanliga i sensornätverk.

## Litteraturlista

[1] T. Fuhrmann, The Use of Scalable Source Routing for Networked Sensors, Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors, Sydney, Australia, May 30-31, 2005 pp. 163-165
[Online]. Tillgänglig på: http://i30www.ira.uka.de/research/publications/p2p/ Hämtad: 1.4.2010

[2] K. André, Scalable routing in mesh networks for devices with limited resources : scalable source routing and optimization proposals, 2009

[3] K. Kutzner, C. Wallenta, and T. Fuhrmann, Securing the Scalable Source Routing Protocol, Proceedings of the World Telecommunications Congress 2006 (WTC 2006), Budapest, Hungary, April 30 - May 3, 2006
[Online] Tillgänglig på http://i30www.ira.uka.de/research/publications/p2p/ Hämtad: 1.4.2010

[4] T. Fuhrmann, A Self-Organizing Routing Scheme for Random Networks, Proceedings of the 4th IFIP-TC6 Networking Conference, LNCS 3462, Waterloo, Canada, May 2-6, 2005 pp. 1366-1370
[Online]. Tillgänglig på http://i30www.ira.uka.de/research/publications/p2p/ Hämtad: 1.4.2010

[5] C. Cramer and T. Fuhrmann, Self-Stabilizing Ring Networks on Connected Graphs, Universität Karlsruhe (TH), Fakultät für Informatik, Technical Report 2005-5, March 2005
[Online]. Tillgänglig på http://i30www.ira.uka.de/research/publications/p2p/ Hämtad: 1.4.2010

[6] K. Kutzner and T. Fuhrmann, Using Linearization for Global Consistency in SSR, Proceedings of the 4th Int. IEEE Workshop on Hot Topics in P2P Systems, Long Beach, CA, 26-30 March 2007
[Online]. Tillgänglig på http://i30www.ira.uka.de/research/publications/p2p/ Hämtad: 1.4.2010

[7] N. Beijar, Zone Routing Protocol (ZRP), [Online]. Tillgänglig på: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.5568&rep=rep1&type=pdf Hämtad: 1.4.2010