Cuprins

Lista figurilor

Capitolul 2

Capitolul 3

Capitolul 4

Capitolul 5

Lista tabelelor

Lista acronimelor

ABR = Associativity-Based Routing

ACK = Acknowledgement

AEADMRA = Ant-based Energy-Aware Disjoint Multipath Routing Algorithm – Rutarea prin căi multiple disjuncte cu cost minim de energie bazată pe furnici.

AODV = Ad Hoc Ondemand Distance Vector - Rutare cu Vector de Distanţă la Cerere.

AOMDV = Ad Hoc Ondemand Multiple Distance Vector - Rutare cu Vectori de Distanţă Multipli la Cerere.

ARP = Address Resolution Protocol – Protocol de rezolvarea a adreselor

CBR = Continuous Bit Rate

CGSR = Clusterhead Gateway Switch Routing

CRC = Cyclic Redundancy Check

CSMA/CA = Carrier sense multiple access/ collision avoidance – Access multiplu cu detecţie de purtărtoare/ Evitarea coliziunilor

CSMA-CD = Carrier Sense Multiple Access with Collision Detection

CTS = Clear To Send- Liber pentru transmisie

DSDV = Destination-Sequenced Distance-Vector - Vector Distanţă cu Destinaţie Dinamică Ordonată

DSR = Dynamic Source Routing – Rutare dinamică folosind sursa

ELFN = Explicit link failure notification technique – Tehincă de notificare explicită a întreruperilor de legătură.

FIFO = First In First Out

FTP = File Transfer Protocol

ICMP = Internet Control Message Protocol – protocolul mesajelor de control din Internet

IEEE = Institute of Electrical and Electronics Engineers

IP = Internet Protocol – Protocol Internet

LAN = Local Area Network

MAC = Medium Access Control

NAM = Network Animation

NS = Network Simulator

OSPF = Open Shortest Path First

RIP = Routing Information Protocol

RERR = Route Error – mesaj de eroare de rută

RREQ = Route Request - mesaj de cerere de rută

RREP = Route Reply – mesaj de răspuns la rută

RTO = Retransmission Time-Out – Expirarea timpului de retransmisie

RTS = Ready To Send- Gata pentru transmisie

RTT = Round Trip Time – Timpul estimat în circuitul dus-întors către şi dinspre o destinaţie

SSR = Signal Stability Routing

TCP = Transport Control Protocol – Protocol de control al transportului

Tcl = Tool Command Language

TTL = TTL- Timp de viaţă

TORA = Temporally Ordered Routing Algorithm

UDP =User Datagram Protocol - Protocol cu Datagrame Utilizator

WRP = Wireless Routing Protocol

WSN = Wireless Senor Network

Cuvânt înainte:

Această lucrare prezintă un studiu al protocoalelor de rutare în rețelele de senzori.

Rețelele de senzori se caracterizează prin mobilitate și dinamicitate, dar în același timp și prin eficiență, simplitate și costuri reduse. Având diverse topologii este necesară autoconfigurarea, dar și eficientizarea consumului de energie pentru a asigura o durată de viață cât mai mare a rețelei. Astfel de rețele sunt o preocupare actuală pentru eficientizarea altor domenii. Aplicațiile acestor rețele sunt vaste și au ca principale caracteristici auto-organizarea și dinamicitatea. Interesul în creștere pentru astfel de rețele face ca dorința de performanță să fie cât mai mare. Principalul mod prin care se poate crește performanța în astfel de rețele este prin alegerea protocolului de rutare optim pentru aplicația dorită. Aceste protocoale optimizează atât găsirea rapidă a celei mai bune rute, siguranța transferurilor cât și consumul de energie al rețelei, ceea ce este un aspect foarte important pentru durata de viață a acesteia. Un aspect ce nu trebuie omis este schimbarea rapidă și frecventă de topologie, lucru la care protocoalele de rutare trebuie să facă fața fără probleme.

Lucrarea de față își propune să prezinte o analiză a celor mai utilizate protocoalelor de rutare în astfel de rețele concepute special pentru mediile mobile de acest fel. Indicatorii de performanță prin care se încearcă evidențierea comportamentului acestor protocoale sunt throughput, întârzierile end-to-end. Am folosit diferite scenarii în care este evidențiat protocolul cu cele mai bune rezultate din punctul de vedere al indicatorilor folosiți. Scenariile au fost gândite în așa fel încât să modeleze aplicații reale ale rețelelor de senzori având drept scop identificarea protocolului cu cea mai bună comportare pentru respectiva aplicație.

1. Introducere

1.1. Concepte generale

Dispozitivele ieftine și de dimensiuni reduse ce au integrați senzori cu capacități de prelucrare, comunicare și memorare, dar și interconectarea acestora în rețele aduce o multitudine de oportunități. Tehnologia cu ajutorul căreia sunt construite rețelele de senzori trebuie să fie aleasă în funcție de aplicația în care aceste rețele sunt utilizate. Dezvoltarea acestor rețele a fost determinată, în mare măsura, de tehnologie și de evoluția rapidă a acesteia. [3]

Astfel un nou tip de rețele a apărut, rețelele de senzori wireless, WSN (Wireless Sensor Network). Rețelele de acest tip sunt alcătuite din noduri ce interacționează cu mediul înconjurător. Acestea măsoară sau controlează diverși parametri fizici. Îndeplinirea unei sarcini este condiționată de colaborarea mai multor noduri ce vor comunica prin comunicație wireless. Chiar dacă aceste rețele includ și elemente de control numele de rețele de senzori wireless a fost ușor acceptat. În literatura de specialitate se mai întâlnește și termenul de rețele de senzori și elemente de execuție wireless. [3]

WSN beneficiază de calități importante și de caracteristici deosebite fiind posibilă implementarea acestora în diverse aplicații din viața reală. Flexibilitatea acestor rețele este factorul care determină interesul crescut pentru aceste rețele. Astfel o clasificare precisă a WSN-urilor este imposibilă întrucât nu exista nici un set de criterii unic. O altă consecință a acestei flexibilități crescute este faptul că nu exista o soluție unică ce să se potrivească oricărei aplicații. [4]

In majoritatea aplicațiilor nodurile sunt alimentate pe baza unor baterii. Dat fiind locurile greu accesibile în care sunt plasați acești senzori și faptul că schimbarea bateriilor este dificilă, iar uneori imposibilă, se dorește o durată cât mai lungă de viață a acestor rețele, iar pentru acest lucru trebuie ținut cont de eficiența energetică a oricărei soluții ce este propusă. De asemenea costurile nodurilor și dimensiunile acestora trebuie să fie reduse pentru a putea utiliza un număr cât mai mare de noduri și pentru a putea plasa aceste noduri în zone greu accesibile. Pe de altă parte precizia unui nod individual este scăzută însă privind în ansamblu o rețea de senzori poate avea rezultate foarte bune prin colaborarea și comunicarea nodurilor constituente în ciuda prețului scăzut și a simplității necesare.

Rutarea poartă sarcina cea mai importantă în astfel de rețele întrucât astfel se poate asigura o comunicație rapidă și sigură, eficiența energetică, auto-organizarea și numai prin aceste protocoale rețele de senzori beneficiază de caracteristicile speciale. Principala sarcină a protocoalelor de rutare este de a descoperi cele mai scurte rute, actualizate, eficiente și sigure pentru a transfera datele de la sursă la destinație. Principala problemă este că flexibilitatea crescută a acestor rețele, caracteristică pentru care sunt și atât de apreciate, îngreunează rutarea. Putem aminti aici de caracterul dinamic al rețelelor de senzori, mobilitatea crescută ce modifică des topologia rețelei. Astfel este nevoie de protocoale de rutare rapide care să găsească ruta dintre sursă și destinație ușor pentru a transporta datele rapid și sigur. Protocoalele de rutare clasice, cele folosite în rețele cu cabluri nu puteau fi folosite și astfel au fost concepute noi protocoale adecvate. Numărul și evoluția acestora a fost una importantă. [3]

1.2. Formularea problemei

Interesul crescut pentru rețelele de senzori a făcut ca problema eficienței protocoalelor de rutare să fie cea care pune piedici în utilizarea rețelelor de senzori și în același timp stârnește ambiția cercetătorilor.

Natura dinamică a acestor rețele face ca protocoalele de rutare să întâmpine o serie de probleme. Fie că este vorba despre un scenariu dinamic prin definiție fie că este vorba despre un scenariu static nivelul energetic al nodurilor transformă orice rețea de senzori într-una cu o topologie variabilă. Rețelele de senzori sunt afectate de apariția, dispariția de rute ce cauzează adesea pierderi bruște de pachete și întârzieri în rețea. Alți factori ce afectează rețelele de senzori sunt dimensiunea rețelei, încărcarea acesteia, lărgimea de bandă sau puterea de transmisie. De asemenea trebuie luate în considerare și problemele moștenite de la rețelele clasice wireless cum ar fi problema terminalelor ascunse, a legăturilor asimetrice, problema localizării, problema congestiei, probleme TCP. [3]

Ținând cont că majoritatea protocoalelor de rutare existente pentru rețelele de senzori au, de multe ori, performanțe scăzute, un studiu cu privire la problemele principale și condițiile în care protocoalele de rutare au rezultate bune sau slabe este binevenit. Este importantă studierea modului în care diverși parametri ai rețelelor variază în diferite condiții de utilizare a protocoalelor de rutare.

1.3. Motivare

In ciuda faptului că mulți algoritmi de rutare pentru reţele de senzori au fost propuși în ultimii ani, în materie de studii de performanță care să compare capabilitățile acestor algoritmi nu putem spune același lucru, în literatura de specialitate existând un număr mic. Foarte puține studii compară comportarea protocoalelor într-o manieră realistă, cu scopul de a contrasta performanțele.

Această cercetare are ca scop oferirea unui studiu cât mai complet și realist al problemelor de rutare prezente în reţelele de senzori, dar și o analiză a performanţelor mai multor protocoale de rutare existente. Se dorește prezentarea unor soluţii de optimizare a acestora şi analiza câștigului de performanţă oferit de aceste soluții de optimizare.

Scenariile luate în calcul, au fost concepute astfel încât să scoată în evidenţă evenimente realiste, cu o probabilitate mare de realizare. Ca și protocoale de rutare de test, am ales două protocoale reactive DSR, AODV şi un protocol proactiv DSDV. Alegerea acestora a fost făcută luând în considerare gama variată de elemente arhitecturale şi concepte de rutare oferite. Printre acestea se numără rutarea multi-salt, actualizări periodice și rutare on demand. Pe lângă acestea, protocoalele alese au fost propuse în standardele IETF (Internet Engineering Task Force).

1.4. Obiective

Principalele obiective ale acestei lucrări este de a scoate în evidenţă principalele probleme de performanţă a algoritmilor de rutare în reţele de senzori şi de a gândi soluţii pentru optimizarea acestora.

De asemenea, studiul vizează utilizarea modalităților de simulare existente cu scopul explorării comportamentului algoritmilor supuși unor situaţii dificile și analiza diferitelor elemente de arhitectură.

Obiectivele principale ale acestei lucrări sunt:

- Scoaterea în evidență a principalelor caracteristici ale reţelelor de senzori;
- Explicarea funcţionării protocoalelor de rutare (AODV, DSDV, DSR);
- Evidențierea constrângerilor şi problemelor din cadrul rutării în reţele de senzori;
- Examinarea câtorva abordări sugerate în literatură pentru a optimiza protocoalele de rutare;
- Prezentarea variantei de optimizare a protocolului AODV, AOMDV, incluzând prezentarea matematică și modul de funcţionare;
- Simularea diferitelor scenarii pentru analiza comportamentului protocoalelor;
- Concluzii

Studiul simulativ cuprinde:

- Descrierea unor scenarii, ce modelează aplicații reale şi în care reţele de senzori au un grad ridicat de aplicabilitate;
- Evaluarea performanţelor protocoalelor în aceste scenarii;
- Evaluarea performanţelor protocolului optimizat, AOMDV;
- Analiza rezultatelor şi concluzii.

2. Rețele de senzori wireless (WSN)

Rețelele de senzori wireless sunt formate din senzori autonomi ce sunt distribuiți într-un spațiu având scopul de a monitoriza mediul. Acești senzori pot urmări condiții fizice precum temperatura, umiditatea, vibrațiile, presiunea, etc. sau pot urmări diferite evenimente cum ar fi fenomene meteo, animale sălbatice etc. Datele pot fi transmise de fiecare senzor către un nod colector sau rețeaua poate fi organizata ierarhic în funcție de aplicația în care sunt folosite rețelele de senzori. Inițial dezvoltarea rețelelor de senzori a fost motivată de aplicațiile militare, de necesitatea supravegherii câmpului de luptă. În zilele noastre rețelele de senzori wireless sunt folosite în aplicații industriale, aplicații de monitorizare și control, etc.[3]

Rețelele de senzori wireless sunt alcătuite dintr-un mare număr de noduri, cu capabilități de detecție, procesare și de comunicație. De obicei în astfel de rețele senzorii sunt împrăștiați aleator și nu este nevoie de cunoașterea prealabilă a poziției acestora. Acest lucru se reduce la faptul că protocoalele trebuie să poată organiza rețeaua fără intervenție din exterior. Pe de altă parte nodurile componente ale unei rețele de senzori wireless trebuie să se coordoneze pentru informații de înaltă precizie despre mediul monitorizat. [3]

2.1. Caracteristicile WSN

Rețelele de senzori wireless sunt caracterizate prin [4] [3]:

Auto-organizare – Parametri de configurare trebuie să fie determinați de rețeaua de senzori în mod automat. Pe lângă acestea sunt incluse operațiile: adresarea, rutarea, găsirea poziției, controlul puterii, etc. În unele aplicații nodurile din rețea își pot coordona pozițiile astfel încât să nu existe zone cu fără acoperire.

Resurse limitate – Capabilitățile rețelelor de senzori wireless sunt limitate tocmai din cauza flexibilității pe care o oferă aceste rețele. Ne dorim să avem posibilitatea de a avea un număr mare de noduri care, eventual, să fie de unică folosință datorită zonelor greu accesibile în care sunt plasați. Acest lucru determină dorința de a avea noduri de rețea ieftine ceea ce înseamnă și resurse limitate oferite de acestea. Resursele la care facem referire sunt energetice, capacitate de procesare, memorie, etc.

Dinamicitate – Aceste rețele beneficiază de schimbări foarte dese de topologie, fie că este vorba despre mișcări ale nodurilor, fie că este vorba de noduri care își opresc activitatea, fie că este vorba de noduri noi adăugate;

Dezvoltarea în arii greu accesibile fără infrastructura existentă reprezintă una din cele mai importante caracteristici;

Rutare multi-salt – Într-o astfel de rețea pachetele de date ajung de la nodul sursă la nodul destinație prin parcurgerea mai multor noduri, a mai multor salturi;

Conservarea resurselor – Fie că este vorba de resurse energetice, fie că este vorba de resurse de procesare, aceasta este o caracteristică esențială întrucât astfel este asigurată o durata mare de viață a rețelei. Protocoalele ce sunt făcute să lucreze în astfel de rețele trebuie să funcționeze astfel încât să gestioneze cât mai bine consumul energetic și chiar să îl distribuie uniform în rețea pentru prelungirea duratei de viață a acesteia.

Conexiunea cu alte rețele – Rețelele de senzori wireless trebuie să aibă posibilitatea conectării la alte rețele sau chiar la Internet pentru a trimite datele prelevate din mediu.

Securitatea – Acesta este un aspect ce trebuie luat în seamă. Rețelele ce au ca mediu de comunicare aerul, rețelele wireless, sunt expuse atacurilor diverse. Apariițiile recente de soluții pentru aceste probleme, pentru prevenire și combatere, au fost implementate și în rețelele de senzori wireless.

Scalabilitate – Multe aplicații ale rețelelor de senzori wireless necesită o creștere a mărimii prin adăugarea de noduri suplimentare. Această caracteristică, deși dificilă din punctul de vedere al rutării, este suportată de rețelele de senzori wireless.

Toleranța la defectare – După cum am precizat mai sus rețelele de senzori wireless sunt alcătuite din noduri ieftine pentru a ne permite pierderea sau defectarea acestora. Din punctul de vedere

al rutării WSN-urile trebuie să se poată reconfigura ușor atunci când o parte din noduri nu mai poate continua activitatea. Prin acest proces se realizează și auto-întreținerea rețelei și asigurarea serviciilor dorite pentru o perioadă lungă de timp fără intervenție umană.

2.2. Aplicații ale rețelelor de senzori wireless

Rețelele de senzori wireless au numeroase aplicații ce vor contribui la dezvoltarea domeniilor în care vor fi aplicate și vor duce la apariția unora total noi. Totalitatea aplicațiilor în care rețelele de senzori wireless își găsesc locul se pot grupa în funcție de câteva tipuri. Cele mai relevante dintre acestea sunt [3]:

Detectarea de evenimente – La scurt timp după detecția unui eveniment nodurile senzor vor trimite nodului destinație date despre acesta. Evenimentele pot fi simple, cele detectate de un singur nod, de exemplu modificarea temperaturii. Pot exista însă evenimente complexe ce vor necesita colaborarea mai multor noduri;

Măsurări periodice – Poate exista dorința unei măsurări a unor mărimi la diferite momente de timp. De multe ori, comunicarea acestor maturatori se face doar la apariția unui eveniment, iar dacă nu perioada la care se fac este aleasă în funcție de aplicație;

Aproximarea funcțiilor – Modul în care o mărime își modifică valorile de la un loc la altul poate fi privit ca o funcție de locație. O rețea de senzori wireless poate fi folosită pentru aproximarea unor astfel de funcții prin folosirea unui număr limitat de eșantioane obținute de la diverși senzori. Aproximarea făcută este transmisă nodului destinație. Detaliile privind precizia aproximării, consumul energetic depind de aplicație;

Detecția marginilor – Un exemplu pentru acest tip de aplicație este identificarea zonelor izotermice dintr-un incendiu de pădure utile în detectarea marginilor incendiului;

Urmărire – Sursa evenimentelor își poate modifica poziția și poate fi folosită în aplicații de securitate pentru urmărirea intrușilor. Transmiterea evenimentelor la destinație poate include și trimiterea informațiilor cu privire la viteza și direcția de deplasare. Pentru acest tip de aplicație nodurile trebuie să se coordoneze și să coopereze pentru a trimite informații corecte.

Exemple de scenarii:

- aplicații pentru asistența în caz de dezastru
- controlul mediului și cartografierea biodiversității
- clădiri inteligente
- gestiunea clădirilor
- supravegherea mașinilor și întreținerea preventivă
- agricultura de precizie
- medicină și întreținerea sănătății
- logistică
- telematică

2.3. Probleme principale ale rețelelor de senzori

2.3.1. Probleme moștenite de la rețele wireless

2.3.1.1. Problema terminalelor ascunse [1]

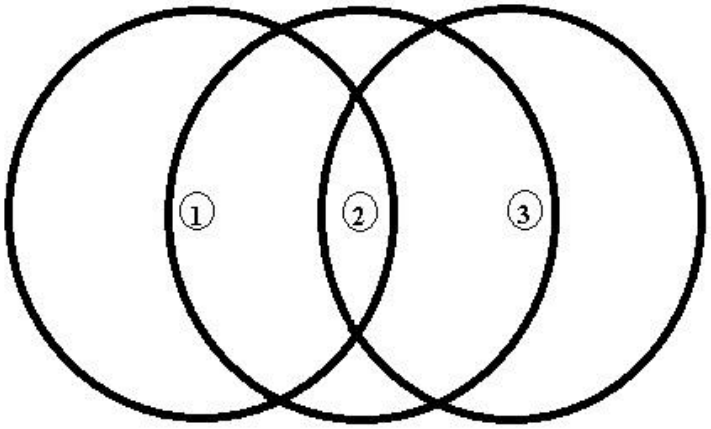


Figura 2.1 Problema terminalelor ascunse

Considerăm trei noduri ca în figura 2.1 și scenariul în care nodul 1 transmite nodului 2. Atunci când nodul 3 asculta mediul, acesta nu va sesiza faptul că nodul 1 emite, pentru că nodul 1 este în afara domeniului sau, și decide ca poate transmite. Dacă nodul 3 transmite, va interfera în nodul 2 cu cadrul de la nodul 1, iar acesta din urma va fi compromis. Faptul că nodul 3 și nodul 1 se afla în raza de acțiune al nodului 2, dar nu se pot detecta între ele implica faptul că mecanismul de evitare a coliziunilor folosit în mediile wireless (CSMA/CA-Carrier Sense Multiple Access/ Colission Avidance), nu va funcționa.

Înainte de a începe o transmisiune, un nod trebuie să știe dacă în preajma destinatarului se desfășoară vreo activitate sau nu. Acest lucru este detectat de CSMA prin simpla detecție a purtătoarei. Prin cablu, toate semnalele se propaga la toate stațiile și astfel la un anumit moment poate exista o singura transmisie. Într-un sistem bazat pe radio cu domeniu mic, se pot desfășura mai multe transmisiuni simultan, dacă acestea au destinații diferite și aceste destinații au domenii disjuncte.

O soluție foarte des întâlnită este coordonarea distribuita a transmisiei. Această tehnică presupune că nodurile vor comunica înainte de a transmite efectiv date prin cererea și acordul de a transmite. Acest mecanism se numește RTS/CTS (Request to send/Clear to send). Ideea de baza este ca nodul ce emite să ocupe canalul și să anunțe nodurile vecine de transmisiile în curs. De exemplu în cazul nostru nodul 1 emite un RTS către nodul 2, iar acesta dacă nu are alta transmisiune în curs va emite un CTS spre nodul 1. în acest mod nodul 3, care nu a recepționat RTS-ul de la nodul 1 pentru ca nu se afla în raza nodului 1, va recepționa CTS-ul nodului 2 și va fi astfel informat că există o transmisiune în curs.

2.3.1.2. Problema terminalelor expuse

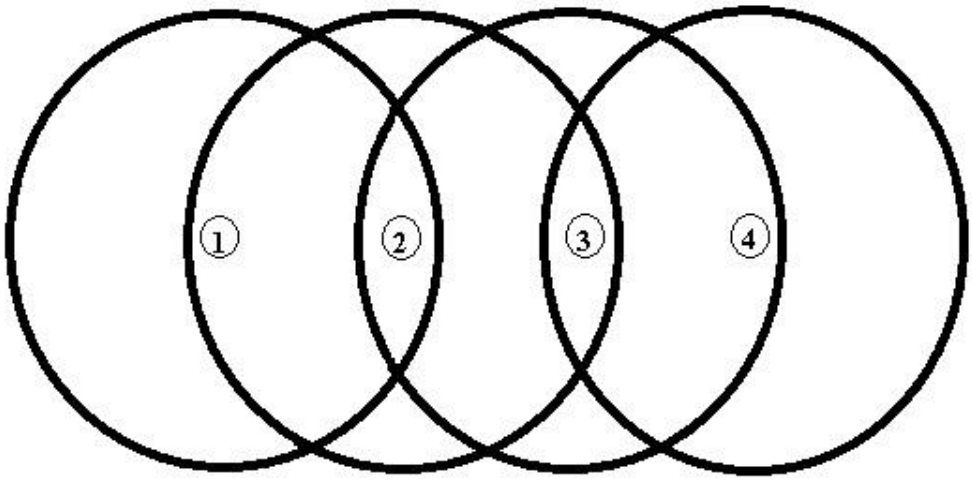


Figura 2.2 Problema terminalelor expuse

Problema terminalelor expuse poate fi privită ca o consecință a implementării mecanismului RTS/CTS(Request to send/Clear to send) ce rezolva problema terminalelor ascunse. Consideram scenariul în care nodul 2 comunica cu nodul 1, iar nodul 3 dorește să comunice cu nodul 4. în momentul în care nodul 3 va asculta canalul va recepționa mesajele de avertizare ale nodului 2 de canal ocupat. Prin urmare nodul 3 va lua decizia eronata ca nu poate transmite către nodul 4 și va aștepta până ce nodurile 1 și 2 vor termina de comunicat. Astfel de erori cauzează întârzieri mari în rețea și poate reduce considerabil rata de transmisie.

S-a demonstrat combinând rezultatele studiilor analitice și rezultatele simulărilor că traficul în rețelele de senzori poate fi îmbunătățit prin neutilizarea mecanismului RTS/CTS. [2]

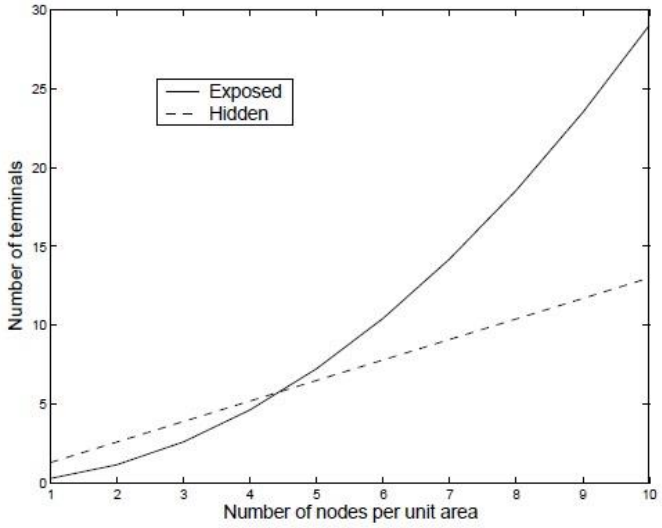


Figura 2.3a Variația terminalelor ascunse și a terminalelor expuse fata de densitatea de noduri. [2]

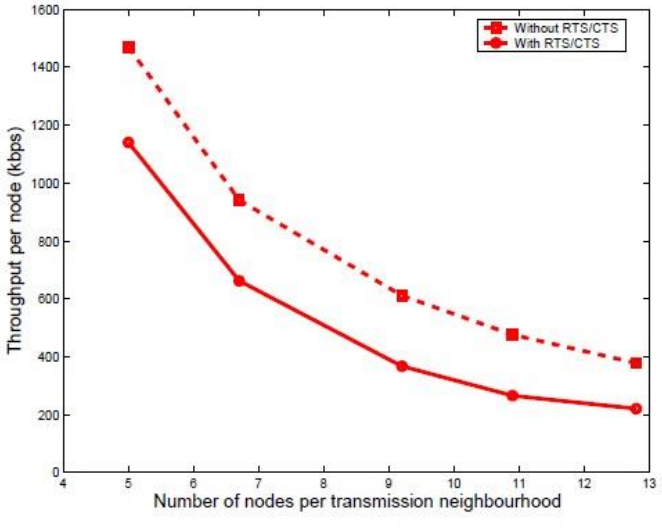


Figura 2.3b Rata transfer cu și fără RTS/CTS. [2]

2.3.2. Probleme specifice rețelelor de senzori

2.3.2.1. Probleme arhitecturale ale nodurilor senzor [3]

1. Mobilitatea reţelei

Rețelele de senzori sunt alcătuite din trei elemente: nodurile rețelei, centrele de colectare şi evenimentele monitorizate. Majoritatea arhitecturilor constau din noduri senzor staţionare. Totuși, poate fi necesar câteodată implementarea suportului pentru mobilitatea centrelor de colectare. Astfel stabilitatea devine o problemă, din cauza necesităţii rutării între noduri mobile. Evenimentul urmărit de rețeaua de senzori poate fi static ceea ce implica doar o monitorizare sau dinamic ceea ce înseamnă detecţie şi urmărire.

Monitorizarea evenimentelor statice permite reţelei o desfăşurare reactivă, generând trafic în momentul raportării. Monitorizarea evenimentelor dinamice necesită o raportare periodică şi generează un volum semnificativ de trafic ce trebuie rutat spre nodul colector.

2. Modul de amplasare al nodurilor

Topologia rețelei de senzori depinde de aplicaţie şi afectează performanţele protocolului de rutare. Putem realiza o clasificare având în vedere modul de amplasare al nodurilor. Putem avea o amplasare deterministă sau putem avea o amplasare cu auto-organizare.

În cazul plasării manuale a nodurilor, obținem o minimizare a coliziunilor şi construirea de rute predeterminate. În cazul formării topologiei în urma unei amplasări aleatoare, poziţia nodului colector sau a nodurilor de agregare este crucială din punct de vedere energetic şi al performanţelor. În special când distribuţia nodurilor nu este uniformă, optimizarea formării clusterelor devine crucială pentru o funcţionare eficientă energetic.

3. Comunicaţia intre noduri

Considerentele energetice influențează puternic procesul de stabilire a rutelor în rețelele de senzori.

Atunci când nodurile se afla la distante mari, transmisia directa nu este o soluție. Dacă nodurile s-ar afla în proximitatea nodului colector rutarea directă ar fi suficientă.

In mod frecvent datele sunt transmise prin mai multe hopuri, ceea ce introduce un overhead semnificativ pentru managementul topologiei şi controlul accesului la mediu.

4. Transferul datelor

Transferul datelor către nodul colector, în funcție de aplicația în care este folosita rețeaua de senzori, se face în mod continuu, la apariţia unui eveniment, la cerere sau hibrid. În modul continuu, fiecare nod transmite date periodic . În modul de transmisie la cerere şi la apariţie, transmisia de date este declanşată de apariţia unui eveniment sau de lansarea unei interogări de către nodul colector. Protocoalele de rutare şi cele de nivel MAC sunt influenţate de modelul de livrare a datelor, în special în privinţa minimizării consumului şi stabilității rutelor.

5. Sarcinile nodului

Într-o rețea de senzori, depinzând de aplicație, fiecărui nod i se asociază o funcţionalitate specifică (retransmisie, analiza, agregare, etc.) deoarece angajarea în mai multe sarcini distincte poate avea un impact major asupra consumului.

Există două moduri de abordare a problemei: alegerea prin software a nodurilor centrale în urma rulării unui algoritm specific sau amplasarea de noduri superioare din punct de vedere al sursei de alimentare, benzii şi memoriei disponibile, care își vor asuma sarcinile liderilor de cluster.

6. Agregarea datelor

Reducerea duratei şi a numărului de transmisiuni se poate face prin eliminarea de informații redundante. Nodurile senzor generează o cantitate semnificativă de informaţii. Este necesara implementarea un sistem care să accepte pachete similare de la mai multe noduri şi să retransmită numai o copie. Agregarea datelor este rezultatul utilizării de funcţii specifice, cum ar fi suprimarea duplicatelor, minim, maxim sau funcţii de mediere. Unele din aceste funcţii pot fi executate parţial sau total de orice nod senzor, permiţând nodurilor să reducă traficul din rețea. Astfel se pot obţine optimizări ale consumului energetic la nivelul reţelei prin faptul că este mai scump să comunici decât să efectuezi calcule. Această tehnică a fost deja folosită într-un număr semnificativ de protocoale de rutare. Există şi implementări în care sarcina agregării este atribuită unor noduri specializate. Din păcate, agregarea complică protocolul MAC, deoarece eliminarea pachetelor redundante necesită arbitraj instantaneu la accesul la mediu. Din această cauza, sunt aplicabile numai protocoale bazate pe CSMA sau CDMA, ducând la o creştere în consum.

2.3.2.2. Problemele implementării rețelelor de senzori [3]

1. Banda limitată

Lăţimea de bandă este o problemă tipică pentru reţele radio de uz general necesar pentru atingerea nivelului dorit de calitate. Pentru reţelele de senzori limitările de bandă sunt de asemenea o problemă. Traficul de date în rețelele de senzori este constituit atât din pachete ce necesita procesarea în timp real cât și din pachete ce pot aștepta. Alocarea benzii disponibile prioritar pentru traficul ce necesita procesare în timp real nu este o soluţie viabilă. O soluție este realizarea unui compromis în calitatea datelor pentru a putea transmite şi traficul neprioritar. Suplimentar, va fi necesara utilizarea de rute multiple independente, pentru a împărţi fluxul de date şi a permite realizarea impunerilor de calitate.

Limitările energetice a puterii de calcul din reţelele de senzori fac dificila construcţia de rute independente pentru acelaşi flux. De asemenea este posibila creșterea numărului de coliziuni pe rutele pe care se transmite informaţia.

2. Eliminarea redundanţei

Reţelele de senzori se caracterizează prin redundanţa multiplă a datelor generate. Agregarea datelor pentru traficul cu QoS este dificilă. Simpla comparare a pachetelor ce constituie o imagine sau un flux video este o sarcină dificila, ce se executa cu multe instrucţiuni, iar acest lucru duce la solicitarea intensa a procesorului și deci la un consum mare de energie.

O varianta viabila pentru a face agregarea datelor cu QoS este construirea unui set de reguli la nivel de senzor şi la nivel de grup. De exemplu, pentru datele ce formează un flux de imagini agregarea

poate fi efectuată prin eliminarea traficul generat de senzorii care au poziții similare, întrucât se putem considera că imaginile sunt asemănătoare..

3. Compromisul între consumul de energie şi întârzierea pachetelor

In rețelele wireless este utilizata aproape exclusiv rutarea multi-hop întrucât puterea de transmisie este proporţională cu pătratul distanţei (chiar mai mare în cazul unui mediu zgomotos). Acest tip de rutare are avantajul ca prin adăugarea de stații intermediare scade semnificativ puterea consumata pentru colectarea datelor, dar are dezavantajul creșterii timpilor de întârziere a pachetelor. Întârzierea introdusă de trecerea pachetului prin memoria tampon este de obicei mai mare decât întârzierea cauzată de propagare. Creşterea numărului de staţii intermediare face ca întârzierea să crească, şi complică analiza şi tratarea traficului prioritar.

Pentru a face fața cu succes, traficului din rețelele de senzori trebuie împărțit în mai multe fluxuri distincte și transmiterea pe mai multe căi.

Ajungerea la un compromis între consumul de energie şi întârzierea pachetelor devine astfel o problema greu de rezolvat.

4. Dimensiunea limitată a memoriei tampon

După cum am văzut, rețelele de senzori folosesc pentru a transmite pachete de date rutarea multi-hop. Acest lucru face ca senzorii intermediari să fie folosiți pe post de relee pentru a transmite informația. Nodurile senzor sunt limitate în resursele disponibile de stocare şi procesare. Atunci când recepționează un pachet de date, nodul senzor încarcă pachetul în memoria tampon în vederea analizei. Dacă constată că pachetul nu ii este adresat nodul senzor îl încărca în coadă pentru retransmisie. în funcție de aplicație, pachetele de date diferă în dimensiune, dar sunt în general mici. Astfel memoria tampon a nodurilor senzor variază în dimensiune în funcţie de aplicaţie (agregarea datelor este facilitata de stocarea mai multor pachete în memoria tampon).

Stocarea în memoria tampon a unei cantităţi mari de date este necesara pentru rutarea în condiţiile asigurării QoS pentru o clasă de trafic. Astfel proiectarea senzorilor trebuie să asigure disponibilitatea unei memorii suficiente. în caz contrar, această limitare va introduce o creştere a întârzierii pachetelor ce sosesc pe rute distincte (sau chiar pe aceeaşi rută).

Aceasta problema complica diferențierea traficului prioritar dar și programarea accesului la mediu.

5. Suportul pentru tipuri diferite de trafic

Ne putem imagina ca o aplicaţie poate utiliza un set foarte diversificat de senzori. Putem avea senzori pentru temperatura, presiune, umiditate, detecţia mişcărilor sau chiar şi captură de imagini (urmărirea video a ţintei în mişcare). Senzorii unei astfel de rețele complexe generează informaţiile cu rate diferite atât de achiziţie cât şi de transfer. Acestea se supun unor cerinţe de QoS diferite şi urmează modele distincte de livrare. Poate exista o împărțire a acestor fluxuri de date pe clase de priorități astfel încât întreaga rețea de senzori să respecte un anumit nivel de calitate (figura 2.4).

Protocoalelor de rutare sunt îngreunate de astfel de topologii atât la nivel soft cât şi la nivel hard. Este evidenta necesitatea unor procesoare mai puternice. De exemplu se pot utiliza ASIC-urile pentru tratarea QoS, însa acest lucru duce la creşterea consumului energetic care se traduce printr-o scădere a duratei de funcţionare.

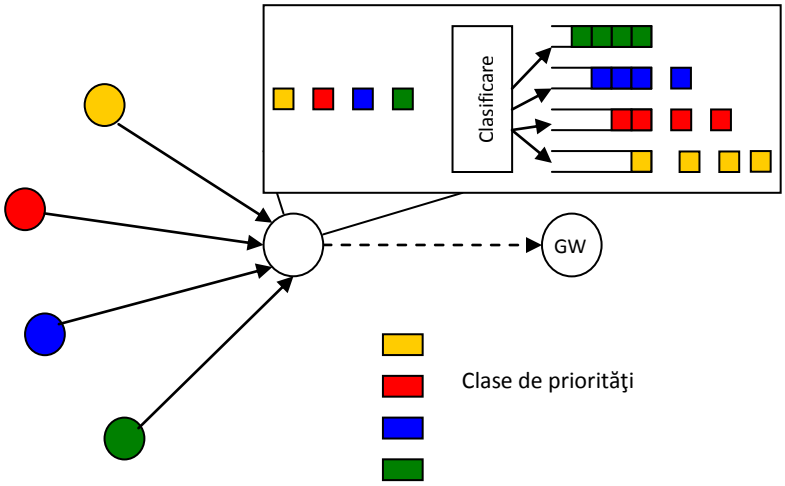


Figura 2.4 Deservirea mai multor cozi de prioritate

2.3.2.3. Problemele de rutare în rețelele de senzori

Într-o rețea multi-salt, nodurile intermediare trebuie să acționeze ca relee pentru pachete de la nodul sursa înspre nodul destinație. în acest caz nodul intermediar va trebui să decidă cărui vecin ii va retransmite pachetul recepționat. în mod frecvent sunt utilizate tabele de rutare, care cuprind cei mai favorabili vecini pentru oricare ar fi destinația unui anumit pachet. Sarcina cea mai importanta a unui protocol de rutare distribuit este de a construi și de a actualiza aceste tabele de rutare.

Pentru buna funcționare și pentru lucrul în timp real care sunt în ziua de azi cerințele obligatorii a oricărei rețele, protocolul de rutare trebuie să urmărească respectarea următoarelor condiții:

1. **utilizarea căilor optime:** Un pachet trebuie să urmeze, pe cât posibil, cea mai buna cale intre sursa și destinație. Calea optima poate însemna lucruri diferite în funcție de aplicația la care este utilizata rețeaua de senzori. Calea cea mai buna poate fi calea cea mai scurta, poate fi calea cu legăturile cele mai sigure, poate fi calea cu cel mai mic cost sau poate fi calea care echilibrează traficul rețelei. De asemenea pentru ca marea majoritatea a rețelelor de senzori se bazează pe noduri ce folosesc baterii, calea optima poate tine cont de nivelul energetic al bateriilor și necesitatea de a transmite datele neapărat prin acel nod;

2. **minimizarea tabelelor de rutare:** Cu cât tabelele de rutare sunt mai mari, cu atât încărcarea rețelei va fi mai mare, iar acest lucru va duce la o convergenta dificila a rețelei către o stare stabila. Integritatea informațiilor despre rute este menținuta de anumite protocoale prin schimb periodic de tabele de rutare. Din aceste motive este de dorit ca tabelele de rutare să fie cât mai scurte. De asemenea în rețele de senzori nodurile dispun de resurse limitate, iar memoria este una din aceste resurse. Astfel o tabela de rutare mare va fi imposibil de memorat;

3. **reducerea numărului mesajelor de control:** Numărul mare al mesajelor de control pe care protocoalele de rutare le generează este o sursa de întârziere a pachetelor de date transmise intre noduri sau duc chiar la congestionarea rețelei. Într-o rețea de senzori dinamica în care topologia rețelei se modifica des mesajele de control sunt inevitabile, dar numărul acestora trebuie redus la minim;

4. **robustețea:** Pentru a evita greșeli de dirijare ale pachetelor informațiile din tabelele de rutare trebuie să fie corecte;

5. **evitarea buclelor:** Este o cerința pentru a ne asigura de consistenta tabelelor de rutare și pentru a evita probleme precum numărarea la infinit ce poate duce la blocarea rețelei;

6. **securitate:** Protocoalele de rutare din rețele de senzori trebuie să adopte anumite politici de securitate. Acestea sunt o cerința importanta în orice tip de rețea, dar mai ales într-o rețea de genul celei de senzori, o rețea dinamica.

La toate acestea se adăuga cerințele specifice rețelelor de senzori pentru protocoalele de rutare:

7. **auto-configurabil:** Este necesar un algoritm de rutare care să se adapteze și să reconfigureze rapid rețeaua de senzori pentru a face fața schimbărilor topologice dese;

8. **protocol de rutare distribuit: întrucât** nu exista un administrator al rețelei, fiecare nod al rețelei de senzori trebuie să aibă implementat protocolul de rutare pentru o administrare distribuita;

9. **conservarea energiei:** Problema energiei este una importanta în rețelele de senzori. Dat fiind faptul ca nodurile sunt alimentate, în general, de la baterii consumul energetic trebuie să fie unul redus și rațional;

10. **admite legături unidirecționale: în** funcție de nivelul energetic al bateriilor sau datorita proiectării nodurile componente ale unei rețele de senzori pot avea puteri de emisie diferite. să presupunem ca un nod transmite către un altul. Acesta din urma nu poate răspunde întrucât puterea să de emisie este mai mica și ca atare primul nod nu se afla în aria să de emisie. Protocolul de rutare trebuie să facă posibila comunicarea intre doua noduri.

11. **simplitate:** Un nod dintr-o rețea de senzori are atât rolul de stație cât și rolul de ruter pe care le desfășoare uneori simultan. Nici una dintre cele doua funcționalități nu trebuie să fie obstrucționata de cealaltă, iar în acest sens un algoritm de rutare simplu este binevenit.

2.3.2.4. Probleme de securitate

Securitatea este o problema importanta în rețelele obișnuite, dar cu atât mai mult în rețelele de senzori, întrucât nu exista o topologie stabila și dacă vorbim de un mediu de transmisie wireless exista și o vulnerabilitate crescuta a canalului și a nodurilor rețelei. Astfel trebuie implementate mecanisme de securitate care să asigure confidențialitatea, consistenta datelor din rețelele de senzori.

Atacurile în rețelele de senzori sunt de doua tipuri: atacuri active și atacuri pasive. Prima categorie, cea a atacurilor active, presupune ca atacatorul pătrunde în rețea, alterează datele și astfel va deregla buna funcționare a protocoalelor de rutate, dereglând astfel rețeaua și performatele acesteia. Cea de a doua categorie, cea a atacurilor pasive, presupune faptul ca atacatorul are acces la datele din rețea fără a produce modificări asupra acestora. Astfel acesta încalcă politica de confidențialitate prin obținerea de informații asupra cărora nu ar avea acces în mod normal, practic, un furt de informație. [6]

Ceea ce pare o sarcina simpla la prima vedere, asigurarea securității în relele de senzori wireless, nu este deloc ușor de atins. La acest lucru contribuie numeroși factori ce îngreunează aplicarea masurilor necesare pentru asigurarea securității. Printre aceștia amintim faptul ca nodurile rețelelor de senzori wireless au capabilități limitate: energetice, capacitați de procesare, memorare. De asemenea trebuie precizat faptul ca inexistenta unei topologii fixe a rețelelor, dar și mediul de transmisie wireless îngreunează asigurarea securității. Din aceste cauze multe protocoale de rutare omit modalitățile de asigurare a securității, iar acest lucru face ca rețelele să fie foarte sensibile atacurilor. [4]

Un protocol de rutare este sigur dacă poate asigura detecția nodurilor corupte, a rutei corecte, dar și asigurarea secretului asupra topologiei rețelei. Cunoașterea topologiei de către un posibil atacator reprezintă un mare avantaj pentru acesta întrucât va ști exact care sunt nodurile vulnerabile ale rețelei. De exemplu atacatorul poate identifica și folosi nodurile în care poate apărea congestia rețelei pentru a altera datele. în cazul în care s-a produs totuși un atac, protocolul de rutare trebuie să aibă capacitatea de a relua activitatea într-un mod normal cât mai repede. [4]

Pentru securizarea transmisiilor de diferite tipuri de date de-a lungul rețelelor se folosesc numeroase tehnici bine cunoscute dintre care enumeram criptografia, steganografia, semnătura digitala și accesul securizat la nivel fizic. [5][6]

Tehnica criptării și a decriptării folosita pentru rețelele tradiționale cu fir nu e fezabil să fie aplicata direct pentru rețelele de senzori. Rețelele wireless de senzori suferă limitări semnificative atunci când vine vorba de capacitate de procesare, memorie, energia disponibila. Prin aplicarea oricărui sistem de criptare e necesar transmiterea mai multor biți ceea ce înseamnă procesare, memorie și baterie suplimentara. Deci, aplicarea criptării poate duce la întârzieri, pierderi de pachete. Mai mult decât atât se pune problema administrării acestor chei cu un minim de intervenție umana sau chiar fără nici un fel de intervenție. [5]

In tabelul 2.1 este prezentat un sumar al diferitelor sisteme de securitate pentru rețelele de senzori wireless. [5]

Sistem de Securitate	Atacuri Suportate	Arhitectura Rețelei	Caracteristică Importantă
JAM	DoS Attack (Jamming)	Traditional wireless sensor network	Avoidance of jammed region by using coalesced neighbor nodes
Wormhole based	DoS Attack (Jamming)	Hybrid (mainly wireless partly wired) sensor network	Uses wormholes to avoid jamming
Statistical En-Route Filtering	Information Spoofing	Large number of sensors, highly dense wireless sensor network	Detects and drops false reports during forwarding process
Radio Resource Testing, Random Key Pre-distribution etc.	Sybil Attack	Traditional wireless sensor network	Uses radio resource, Random key pre-distribution, Registration procedure, Position verification and Code attestation for detecting Sybil entity
Bidirectional Verification, Multi-path multi-base station routing	Hello Flood Attack	Traditional wireless sensor network	Adopts probabilistic secret sharing, Uses bidirectional verification and multi-path multi-base station routing
On Communication Security	Information or Data Spoofing	Traditional wireless sensor network	Efficient resource management, Protects the network even if part of the network is compromised
TIK	Wormhole Attack, Information or Data Spoofing	Traditional wireless sensor network	Based on symmetric cryptography, Requires accurate time synchronization between all communicating parties, implements temporal leashes
Random Key Predistribution	Data and information spoofing, Attacks in information in Transit	Traditional wireless sensor network	Provide resilience of the network, Protect the network even if part of the network is compromised, Provide authentication measures for sensor nodes
REWARD	Black hole attacks	Traditional wireless sensor network	Uses geographic routing, Takes advantage of the broadcast inter-radio behavior to watch neighbor transmissions and detect black hole attacks
Tiny Sec	Data and Information spoofing, Message Replay Attack	Traditional wireless sensor network	Focuses on providing message authenticity, integrity and confidentiality, Works in the link layer
SNEP & μTESLA	Data and Information Spoofing, Message Replay Attacks	Traditional wireless sensor network	Semantic security, Data authentication, Replay protection, Weak freshness, Low communication overhead

Tabelul 2.1 Sisteme de securitate în rețelele de senzori wireless

In timp ce criptografia ascunde conținutul mesajului, steganografia urmărește ascunderea existentei vreunui mesaj. Principalul obiectiv al steganografiei este să modifice purtătoarea astfel încât să nu fie perceptibila. La fel ca la criptografie se pune problema resurselor limitate pentru implementarea steganografiei în rețelele de senzori wireless, iar cercetările sunt deschise în acest domeniu. [5]

Accesul securizat la nivel fizic în rețelele de senzori wireless poate fi oferit prin utilizarea frecventei "săltărețe". O combinație dinamica a parametrilor: frecventele disponibile pentru salt, intervalul de timp intre salturi și secvența în care frecvențele din setul de salt disponibile sunt utilizate poate fi utilizata cu un minim de resurse utilizate. [5]

Încă din anii 2000 se desfășoară cercetări în domeniul securității rețelelor de senzori wireless și de-a lungul acestei perioade mai multe protocoale s-au evidențiat. SPIN (Security protocols for senzor networks) reprezintă o suita de blocuri optimizate pentru utilizarea cu resurse limitate. SPIN are doua componente principale SNEP și µTESLA. Prima dintre acestea, SNEP, oferă următoarele primitive de securitate: confidențialitatea datelor, autentificarea dual și prospețimea datelor. O problema deosebita, greu de oferit este transmiterea eficientă a autentificării care este o problemă importantă în rețelele de senzori. Cea de a doua componentă, µTESLA, este un protocol nou care oferă transmiterea autentificării prin medii cu resurse limitate. [7]

SEAD este un alt protocol ce conferă siguranța. Acesta este un protocol proactiv ce se ocupa de garantarea mesajelor de actualizare și se bazează pe protocolul DSDV. Folosind funcțiile hash[1], acest protocol autentifică valorile ce se găsesc în câmpurile metrica și numărul de secvență dintr-un mesaj de actualizare. [8]

De asemenea, un alt protocol ce are la baza lanțurile hash și semnătura digitală, SAODV, este utilizat pentru securitatea AODV. [9]

2.3.2.5. Problema localizării

Nivelul Legăturilor de Date are ca principală sarcină descoperirea vecinilor. Informațiile cu privire la vecinii disponibili, dar și calitatea legăturii cu aceștia trebuie să fie cunoscută de nivelele superioare, dar mai ales de către protocolul de rutare. Cunoașterea acestor informații este esențială în decizia rutării, mai ales atunci când există o situație dificilă pentru rețea din punctul de vedere al resurselor evitânduse astfel pierderea de pachete.

Inexistentă unei topologii stabile, în rețelele de senzori wireless, face ca problema localizării vecinilor să creeze dificultăți datorită faptului că trebuie folosită inundarea (flooding) mediului. Probleme importante precum suprasolicitarea rețelei sau chiar congestia rețelei apar atunci când vorbim de rețele cu dimensiuni mari. Aici se impune aplicarea flooding-ului treptat, pe suprafețe mici, până la

[1] În sens matematic, **funcțiile hash** (clasă de funcții denumite în lucrări de specialitate și **funcții de dispersie** sau **funcții de rezumat**) sunt funcții definite pe o mulțime cu multe elemente (posibil infinită) cu valori într-o mulțime cu un număr fix și mai redus de elemente. Funcțiile hash nu sunt inversabile. În informatică, funcțiile hash sunt folosite pentru a accelera căutările în tabele, cum este cazul în bazele de date mari sau comparările de date. Valoarea unei funcții hash este denumită ***rezumat***, ***valoare hash***, ***cod hash***, ***sumă hash*** sau doar ***hash***. De asemenea, pot fi folosite drept sume de control sau coduri corectoare de erori (deși nu trebuie confundate cu acestea două), sau, în criptografie, drept componente în schemele de semnătură digitală.

găsirea nodului dorit. Acestă tehnică de inundare a mediului este utilizată de majoritatea protocoalelor de rutare, dar cu succes în rețele de dimensiuni mici.

Protocoalele proactive folosesc operațiile de interogare pentru descoperirea vecinilor, însă au dezavantajul de a avea nevoie de un serviciu de mentenanță a locațiilor, având astfel o complexitate crescuta. Protocoalele reactive au avantajul de a nu avea nevoie de serviciul de mentenanță a locațiilor, reducânduse astfel complexitatea, dar utilizează operație de inundare ce se pretează rețelelor de dimensiune mică.

O abordare diferită, ce elimină disfuncțiile operației de inundare constă într-un server capabil să localizeze fiecare nod. Sunt folosite funcțiile hash pentru a asocia identificatorul fiecărui nod de aria să de localizare. O parte din nodurile rețelei de senzori se ocupă mai apoi de stocarea acestor informații. Astfel informațiile referitoare la un anumit nod vor fi returnate de aria să de localizare atunci când un nod este căutat. [10]

Protocolul DREAM, prezentat în lucrarea [11], oferă fiecărui nod posibilitatea de a controla frecvența și aria în care transmite mesajele de control. În acest mod sunt limitate efectele negative ale inundării, dar corectitudinea informațiilor privitoare la localizare descrește odată cu distanța.

2.3.2.6. Problema legăturilor unidirecționale

Problema legăturilor unidirecționale este una importantă în rețelele de senzori.

O clasificare foarte des întâlnită împarte legăturile dintre noduri în bidirecționale, asimetrice și unidirecționale. O legătură bidirecțională este de obicei definită ca fiind o legătură dintre două noduri care poate fi folosită pentru a transmite un mesaj dinspre oricare nod către celălalt. Atunci când vorbim de legături asimetrice sau de legături unidirecționale, în general, nu se face o distincție foarte clară intre cei doi termeni și uneori aceștia sunt considerați sinonimi. De obicei, printr-o legătură asimetrică se înțelege ori o variație a valorii RSSI-ului (Received Signal Strenght Indicator) sau a pachetelor pierdute (rata de livrare). Atunci când se folosește rata de livrare legăturile unidirecționale pot fi văzute ca o subclasă a legăturilor asimetrice unde rata de livrare într-o direcție este 0.

In această lucrare, legăturile unidirecționale apar atunci când, de exemplu, două noduri au puteri diferite de transmitere și vor să comunice unul cu celălalt. Să presupunem că nodul A are o putere mai mică decât nodul B. Deci nodul A se află în raza de acțiune a nodului B, dar nodul B nu se află în raza de acțiune a nodului A. Nodul B trimite nodului A un mesaj pe care acesta îl recepționează, însă, datorită puterii de transmisie mai mică acesta din urmă nu va putea răspunde. Putem astfel concluziona că legătura dintre A și B este unidirecționala. Această situație este foarte des întâlnită în rețelele de senzori atât datorită nodurilor de rețea diferite constructiv, dar și datorită consumului energiei senzorilor într-un mod inegal. Trebuie astfel atrasă atenția asupra necesității utilizării unor protocoale de rutare eficiente din punct de vedere energetic.

Această problemă, a legăturii unidirecționale, poate apărea chiar și atunci când cele două noduri au putere de transmisie egală din cauza variațiilor aleatoare a propagării semnalelor sau a prezenței unei surse de zgomot în apropierea unuia din cele două noduri. În lucrarea [12] sunt prezentate

rezultatele unui experiment în care legăturile sunt clasificate în funcție de rata de livrare în cele trei categorii: bidirecționale (rata de livrare>90%), asimetrice(10%<rata de livrare<90%), unidirecționale (rata de livrare<10%).

Tabel 2.2 : Calitatea legaturilor vs Puterea de transmisie

nivel putere de transmisie	bidirecționale	asimetrice	unidirecționale	numărul de legături
1	50%	43%	7%	500%
3	65%	22%	13%	1038%
9	88%	6%	6%	1135%

Apariția unei astfel de probleme are o influență mare în ceea ce privește buna funcționare a protocoalelor de rutare, mai ales că multe protocoale (DSDV, TORA) sunt concepute numai pentru legături bidirecționale, iar atunci când se confruntă cu o astfel de problemă devin inutile. Alte protocoale (WNX, AODV) dirijează pachetele de date numai de-a lungul legăturilor bidirecționale, dar atunci când există un număr mare de legături unidirecționale își dovedesc ineficienta. În lucrarea [12] este prezentat modul de funcționare al protocolului WNX (Wireless Neighborhood Exploration) care detectează legăturile unidirecționale, bidirecționale și le elimină pe cele unidirecționale.

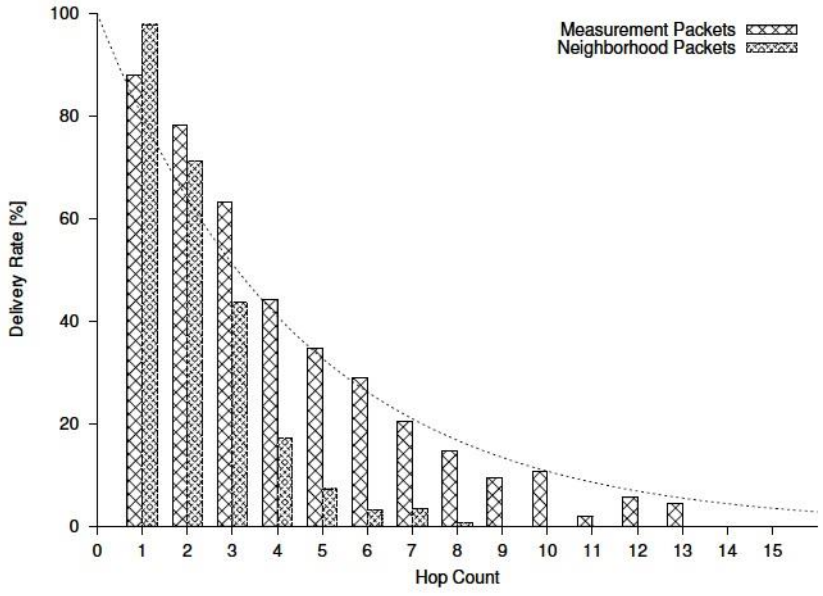


Figura 2.5 Pachete livrate vs. Numărul de hopuri

In figura 2.5 este prezentată rata de livrare a pachetelor pentru measurement packets (pachete de dimensiuni mici) și neighborhood information packets (pachete de dimensiuni mari) în funcție de distanța pe care o au de parcurs. După cum era de așteptat, observam ca rata de livrare a pachetelor pentru protocolul WNX scade cu distanța pe care o au de parcurs pachetele.

Există protocoale ce manipulează cu succes pachetele de date indiferent de tipul legăturilor, așa cum este protocolul DSR. Rata de transfer a rețelei este totuși redusă semnificativ din cauza mecanismelor sofisticate ce sunt utilizate.

O tehnologie ce permite protocoalelor de rutare folosirea atât a legăturilor bidirecționale cât și a legăturilor unidirecționale este SRL (Rutare Sub-Nivel). Acest protocol lucrează intre nivelul MAC și nivelul rețea, elimină problemele legăturilor unidirecționale prin descoperirea tipului de legături, găsirea rutelor inverse și oferirea fiabilității hop cu hop. O altă variantă este DSRL (Dynamic SRL) ce este o variantă îmbunătățită ce oferă posibilitatea scalarii nivelului de încărcare a rețelei în funcție de gradul de asimetrie al rețelei. Principiul de bază al acestui protocol este acela că pentru o rută inversă a unei legături unidirecționale nu se va mai încerca utilizarea aceleași rute ca cea inițială. Chiar dacă ruta inversă va fi mai lungă utilizarea acestui protocol se dovedește a fi rentabilă. [13]

2.3.2.7. Probleme din punctul de vedere al energiei

Considerentele energetice au o influență majoră în proiectarea infrastructurii unei rețele de senzori. După cum știm puterea de transmisie wireless este proporțională cu distanta la pătrat sau chiar mai mare atunci când avem obstacole. Nodurile rețelelor de senzori sunt alimentate de regulă de la baterii, iar aceasta este limitată și reprezintă una din constrângerile majore ale proiectării algoritmilor de rutare. în viitorul apropiat, conform actualelor previziuni, nu vor exista îmbunătățiri semnificative în ceea ce privește capabilitățile bateriilor. Astfel principala metodă de eficientizare a rețelelor din punct de vedere al energiei este consumul rațional al energiei dirijat de algoritmii de rutare implementați.

Se dorește ca prin intermediul rețelelor de senzor nu numai să se transporte date ci aceasta să fie doar sarcina curentă. Ceea ce se dorește este ca prin aceste rețele de senzori să se poată face controlul diverselor utilaje. Eficiența energetică în ceea ce privește transportul datelor este doar o cerință necesara, dar nu suficientă. Scopul final este ca rețeaua să fie funcțională cât mai mult timp. Nu este încă foarte clar cum este posibilă menținerea unei astfel de rețele în stare funcțională pentru cât mai mult timp. Constrângerile existente în ceea ce privește energia bateriilor cu care sunt dotate nodurile pot influența puternic deciziile de rutare ale datelor. Aceasta este cea mai mare constrângere în ceea ce privește energia. [3]

Există mai multe strategii prin care se încearcă eficientizarea consumul de energie. Una dintre acestea este bazată pe faptul că un nod de rețea ce nu desfășoară nici un fel de activitate este trecut într-o stare de hibernare în care consumul de energie este redus semnificativ. Un nod ce nu desfășoară nici un fel de activitate la un anumit moment de timp nu poate fi total închis întrucât, datorită capabilităților multiple ale unui nod de rețea, nu s-ar mai putea asigura conectivitatea rețelei.

O altă strategie de conservare a energiei constă într-o rutare inteligentă. Prin această strategie se dorește identificarea nodurilor strategice și salvarea energiei acestora prin redirijarea unei părți din trafic prin intermediul altor noduri mai puțin utilizate și deci cu rezerve de energie mai mari. Astfel este menținut egal nivelul de energie al nodurilor la nivelul întregii rețele cu dezavantajul introducerii unui minim timp de întârziere a pachetelor de date datorat rutelor ușor ocolitoare. Plecând de la această idee există mai multe modalități de alegere a rutei în funcție de diferite criterii:

- **capacitatea totală maximă disponibilă**: se calculează suma capacității bateriilor și se alege ruta în care aceasta este maximă fără includerea unor devieri inutile;
- **costul minim energetic**: criteriul de alegere a unei rute este suma reluctanței[2] rutei și astfel nodurile cu energie putină vor fi evitate;
- **reluctanța maximă**: costul unei rute este dat de reluctanța maximă a unui nod al căii respective și se va alege ruta cu cost minim;
- **nivelul real de putere disponibil:** atunci când există rute având toate nodurile cu un nivel al energiei peste un anumit prag se va alege ruta cu cel mai scăzut nivel energetic per bit, iar dacă nu există așa ceva se va alege ruta cu cel mai mare nivel minim al bateriei. [3]

Existența sau nu a unei legături, în sistemele wireless, este dată atât de puterea de transmisie cât și de rata transmisiei. Atunci când se creste puterea transmisiei, odată cu creșterea numărului de legături, va creste și consumul energetic. Există foarte multe lucrări ce se concentrează pe modificarea topologiei rețelelor de senzori prin modificarea puterii de transmisie și astfel obținerea conectivității totale a rețelei, inutilă în unele situații. Astfel puterea transmisiei este într-o puternică legătură cu energia consumată, determinând atât numărul legăturilor cât și costul energetic al unei transmisii. Astfel putem observa că prin reducerea hopurilor, adică creșterea puterii de transmisie mărește consumul energetic per-packet și se impune găsirea unei căi de mijloc intre cele două. [4]

Un algoritm de rutare care tine cont de consumul energetic este EAQRP (Energy-Aware QoS Routing Protocol), despre care se vorbește în lucrarea [14] al cărei autori sunt Akkaya și Younis. Acest protocol identifică calea ce are costul minim dar cu un maximum de eficientă energetică și pe de asupra în timpul descoperirii conexiunii este respectat un criteriu end-to-end de întârziere. Atunci când vorbim despre costul unei legături ne referim la mai mulți parametri printre care energia de emisie, energia disponibilă, rata de eroare, etc.

Acest protocol folosește o diferențiere a traficului ce are prioritate față de cel fără utilizând o coadă și un mecanism de clasificare. Nodul sink stabilește procentul benzii ce este alocat celor două tipuri de date, iar în cazul unei congestii va fi folosit. În figura 2.6 este prezentat modul de acțiune al cozii implementate și al clasificatorului.

O variantă îmbunătățită a acestui protocol ar acorda mai multă flexibilitate în alegerea procentului celor doua benzi destinate tipurilor diferite de trafic și astfel ar permite eficientizarea legăturilor.

Un alt algoritm de rutare ce tine cont de costul minim al energiei consumate este AEADMRA (Ant-based Energy-Aware Disjoint Multipath Routing) și este o realizare relativ recentă. Acesta este prezentat în lucrarea [15] și are la bază tehnica bine cunoscută a "inteligenței roiurilor" folosind optimizarea dată de coloniile de furnici (ACO – Ant Colonz Optimization). AEADMRA rezolva probleme dificile prin capacitatea de cooperare a furnicilor. Prin utilizarea acestui algoritm se reduce consumul energetic prin limitarea numărul de mesaje de control, eliminarea transmisiilor

[2] inversul capacității bateriei

redundante și prin găsirea eficientă a rutelor. Acest algoritm introduce o încărcare nesemnificativă în rețea și are aptitudinea de a descoperi, folosind un cost mic, mai multe rute diferite. Un pachet de date, generat de sursa, ce are rolul de a descoperi o ruta este modelat sub forma unei furnici și va produce proporțional distantei dintre sursă și destinație un număr de transmisii/recepții.

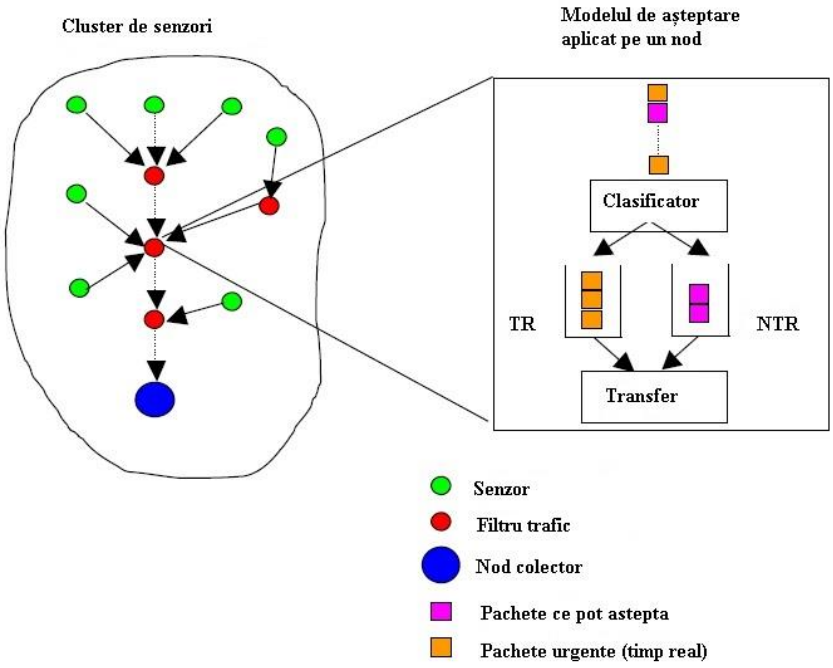


Figura 2.6 Coada și clasificatorul protocolului EAQRP

2.3.3. Probleme la nivelul TRANSPORT [3]

Spre deosebire de rețelele clasice în care nodurile intermediare transportă secvențe independente de octeți, iar aceasta este toată informație cunoscută de aceste noduri, în rețelele de senzori datele ce sunt transportate trebuie să fie cunoscute. Nodurile unei rețele de senzori colaborează și în această bază acționează asupra mediului. Siguranța în rețelele de senzori se referă atât la siguranța transportului, la livrarea efectivă a pachetelor cât și la detecția fenomenelor fizice.

Dacă vorbim, de exemplu, despre Internet putem spune că multitudinea de utilizatori ce folosesc aplicații diverse apreciază singuri QoS-ul. Putem asocia Internetului imaginea unui vehicul ce transportă fluxuri de octeți de la sursă la destinație printr-un sistem multi-hop, iar pentru consumatori se rezumă la protocoalele de transport TCP, UDP ce ii ajută în accesarea acestui serviciu. Astfel, aprecierea calității serviciului are legătură cu protocoalele folosite și nu de aplicație. Întârzierea, viteza cu care se lucrează, rata de eroare pot fi masuri de evaluare. În paralel dacă vorbim despre rețelele de senzori lucrurile nu sunt identice. Nodurile acestor rețele au scopul monitorizării și controlului mediului. Prelucrarea datelor se face ori local, ori în timpul transportului spre destinație și acesta este motivul pentru care informațiile transportate trebuiesc cunoscute. Din acest motiv Internetul și aceste

rețele de senzori sunt diferite, iar acestea din urmă au capacitatea de a executa aceste sarcini dependente de aplicație. Acest lucru reprezintă o trăsătura de bază pentru QoS în rețele de senzori.

Protocolul de transport are alocate în rețelele de senzori wireless, de obicei, următoarele sarcini:

- **Transportul în siguranță al datelor** – presupune detecția și remedierea pierderilor de pachete;
- **Evitarea congestiei** – Atunci când apar mai multe pachete decât capacitatea de transport a rețelei, aceasta începe să arunce din pachete. Acest lucru se traduce printr-o pierdere de energie inutilă și este o piedică în îndeplinirea obiectivului, acela de a avea un transport sigur. Fie se încearcă evitarea acestei situații, fie se dorește reacția promptă la apariția acestei probleme;
- **Control al fluxului** – Din cauza resurselor limitate în ceea ce privește memoria unui nod din rețelele de senzori pot apărea situații în care receptorul să nu poată, pentru moment, să primească un pachet. Astfel este necesar un control al fluxului pentru evitarea pierderilor de pachete;
- **Abstractizarea rețelei** – Din dorința de a evita complicarea aplicațiilor și ferirea acestora de problemele transportului de date se dorește oferirea unei interfețe de programare. Din cauza faptului că nu există standardizat un protocol de transport nu s-a stabilit nici o interfața.

Provocările protocoalelor de transport din rețelele de senzori wireless sunt:

- probleme ale protocolului TCP datorită mediului de lucru (rețele multi-salt de noduri omogene);
- limitările resurselor energetice, de procesare, de memorie;
- variația topologiei rețelei.

2.3.3.1. Probleme ale siguranței, calitatea serviciului

Nivelurile Rețea și Transport au un rol principal în asigurarea siguranței în Internet. Livrarea în siguranță, verificarea fluxului și al congestiei sunt sarcini ale nivelului Transport. "Black box" este considerat tot ceea ce se află sub nivelul Rețea. În rețelele de senzori nu stau la fel. Unul dintre motivele pentru care nu ne putem limita în a considera că siguranța este realizată de un protocol situat deasupra nivelului Rețea sunt resursele limitate de care dispune un nod din rețelele de senzori (energie, memorie, capacitate de procesare). Un alt motiv este reprezentat de faptul că rețelele de senzori sunt privite ca un întreg ce li se alocă o anumită sarcină și este de dorit ca protocoalele să fie proiectate împreună și astfel acestea să se cunoască intre ele și chiar să se poată controla.

Siguranța este una din caracteristicile importante ale QoS. Aceasta nu înseamnă numai livrarea unui pachet ci poate fi descrisă din mai multe puncte de vedere.

- **Siguranța detecției**: Se pune problema existenței posibilității detectării evenimentelor dorite, adică dacă domeniile de detecție ale unui număr de senzori acoperă zonele posibile ale producerii evenimentului dorit;

- **Precizia măsurării**: Atunci când sunt folosiți senzori de o calitate mai slabă o singura interogare a senzorului nu e suficientă pentru o precizie satisfăcătoare. Precizia este o variabilă a diferitelor aplicații în care sunt utilizate rețelele de senzori. Pentru a ne putea baza pe datele obținute de la senzori sunt necesare mai multe citiri, însă numărul acestora trebuie să fie echilibrat întrucât prea multe citiri înseamnă o risipă de energie. Pe de altă parte în unele momente se poate face concesie între precizie și energie în funcție de resursele disponibile sau de aplicație;
- **Comunicarea sigură de la sursa la destinație**: Odată ce un eveniment a fost detectat, acesta trebuie transmis în condiții sigure de la sursă la destinație. Acest lucru se realizează, de obicei, prin traversarea mai multor hop-uri. Uneori este necesar și transportul invers al comenzilor;
- **Livrarea în timp util**: Există aplicații în care e necesar transportul informației într-un timp scurt. Furnizarea informației după un anumit timp nu mai are nici o valoare sau, mai rău, poate produce pagube.

2.3.3.2. Problema TCP

TCP este protocolul de transport orientat pe conexiune prin care se realizează controlul transmisiei și următoarele servicii: inițierea de legături explicite cu acordul ambelor părți, fiabilitate capăt la capăt, transmiterea datelor fără multiplicare și în ordine, controlul fluxului, evitarea congestiei, siguranţa transmisiei, multiplexarea mai multor procese. Aceste caracteristici ale TCP fac ca acesta să fie cel mai utilizat protocol de transport în aplicaţii de Internet. Exemple de aplicaţii sunt: www (HTTP), e-mail (SMTP), transfer de fişiere (FTP) şi serviciu de acces de la distanţă (Telnet).

Protocolul a fost inițial proiectat pentru rețelele clasice, prin cablu, în care principala cauză a pierderii de pachete era congestia. Aceasta este și principala problemă a acestui protocol în rețelele de senzori wireless. Cercetarea continuă face ca TCP să facă față în continuare la diversele tipuri de rețele în care se întinde Internetul.

Performantele protocolului TCP suferă scăderi importante de performanță în rețelele de wireless multi-hop, întrucât nu este pregătit să folosească caracteristicile acestor rețele. De exemplu, în rețelele clasice, pentru a preveni congestia, TCP micșorează rata transferurilor atunci când un pachet este detectat ca fiind pierdut. În cazul rețelelor de senzori wireless pot exista pierderi ale pachetelor ce au alte cauze decât congestia, iar protocolul TCP nu este pregătit pentru aceste cazuri.

Așa cum am spus în paragraful introductiv al acestei secțiuni, modificările dese ale topologiei rețelelor de senzori wireless provoacă pierderi de pachete. Performanța rețelelor de senzori este afectată destul de mult atunci când este folosit ca protocol de transport TCP. Câteva cauze ale pierderilor în rețelele de senzori sunt:

- **mobilitatea nodurilor**: cea mai des întâlnită cauză a pierderilor este reprezentată de mobilitatea nodurilor, implicit a topologiei rețelei;
- **erorile ratelor de bit**: sunt provocate de efectului Doppler şi a atenuărilor de semnal. Astfel va fi activat mecanismul de control al congestiei datorită pierderilor de segmente de date TCP;

- **probleme MAC**: provocate de problemele terminalelor ascunse/expuse;
- **interferentele din mediu;**
- **legăturile unidirecționale:** în acest va fi activat mecanismul de control al congestiei din cauza faptului că protocolul TCP așteaptă mesajul de feedback.

Protocolul TCP nu poate face diferența intre cauzele pierderilor de pachete, adică nu este capabil să identifice dacă este vorba de congestie sau dacă este o problemă tipică rețelelor de senzori wireless. Acesta este motivul pentru care performantele rețelelor wireless sunt cu mult mai scăzute, deși, teoretic, protocolul TCP ar trebui să funcționeze la fel în orice tip de rețea.

Atunci când nu se tine cont de caracteristicile transmisiei wireless apar manifestări nedorite ce cauzează probleme importante în rețea. Cu toate că, în rețelele wireless, protocolul TCP are suficiente cauze de erori, acesta a continuat să fie utilizat ca bază de dezvoltare pentru numeroase încercări de adaptare la mediile wireless.

Protocolul TCP utilizează mai multe mecanisme prin care atinge performante ridicate, evită congestia rețelei și prin care rata de transmisie a datelor este menținută la un nivel corespunzător. Pentru evitarea congestiei există o fereastră de congestie la emițător și una la receptor prin care este stabilit numărul maxim de pachete pe care rețeaua le poate suporta. Pe lângă acest sistem este implementat la nivelul emițătorului o modalitate prin care se determină timpul de retransmitere a datelor, ce poartă numele de RTO (Retransmission Time-Out). Acest sistem este bazat pe RTT (Round Trip Time), sistem ce estimează timpul de călătorie între nodul emițător și cel receptor.

Există două modalități prin care în cadrul TCP este detectată pierderea de pachete. Prima metodă folosește timpul RTO. Se așteaptă până la expirarea acestui timp ceea ce înseamnă că nu a fost recepționat nici un mesaj de confirmare (ACK). Această modalitate nu este eficientă atunci când este folosită singură întrucât nu e necesar să așteptăm atât de mult pentru a trage concluzia că pachetele de date nu au ajuns la destinație. Din acest motiv a fost introdusă a doua metodă bazată pe conceptul de „Retransmisie Rapida” ce pornește atunci când au fost recepționate trei mesaje de confirmare duplicat.

Pierderea rutei

Principala cauză a pierderilor de rută este reprezentată de modificarea frecventă a topologiei rețelei determinată de schimbarea poziției nodurilor. În momentul pierderii rutei bufferele nodurilor intermediare se golesc, adică un procent mare de pachete vor fi aruncate. În rețelele ce folosesc TCP pierderile acestea determină expirarea RTO-ului și de fiecare dată când are loc un astfel de eveniment valoarea să se dublează. Pe lângă aceste efecte protocolul TCP va considera că rețeaua este congestionată și va acționa ca atare. Fereastra de congestie își va micșora dimensiunea ca efect al mecanismelor de control al congestiei. O altă problemă ce apare într-o astfel de situație este timpului de restabilire a rutelor întrucât protocolul TCP nu are nici o atribuție în acest sens. Această sarcină revenind protocoalelor de rutare ce stau la bază.

Din cauza acțiunilor protocolului TCP intervin următoarele efecte negative:

- in momentul în care se restabilește ruta transferul de date va fi îngreunat din cauza dimensiunii reduse a ferestrei de congestie;

- timpul în care TCP va răspunde va fi mărit întrucât RTO va avea o valoare mare, iar acest lucru se va întâmpla și după ce ruta va fi redescoperită având astfel latență mare;
- rata medie a transferurilor va fi puternic afectată din cauza că un timp îndelungat, până la redescoperirea rutei TCP s-a aflat în așteptare.

Îmbunătățiri ale performantei TCP în rețelele de senzori wireless

Îmbunătățiri ale performantelor TCP în rețelele de senzori se pot realiza prin două tipuri de abordări. Astfel există abordări de nivel și abordări de inter-nivel. Abordările de nivel implementări în nivelurile transport, rețea sau legătura de date asigurând o performanță sporită a protocolului TCP și implicit a întregii rețele. După cum ii spune și numele, abordările de inter-nivel își pun baza pe interacțiunea a două nivele din modelul OSI (Open System Interconnection). Pot exista interacțiuni intre nivelul transport, protocolul TCP și nivelul rețea sau intre nivelul transport și nivelul legătura de date. Această abordare a fost gândită pornind de la ideea că straturile de deasupra vor funcționa mai bine în cazul în care beneficiază de mai multe informații de la straturile inferioare.

2.3.3.2.1. Abordări de nivel REȚEA

2.3.3.2.1.1. Rutarea folosind interfețe multiple [15]

Această variantă se bazează pe folosirea mai multor interfețe wireless prin care fiecărui nod de rețea ii sunt atribuite două interfețe wireless diferite. Prima dintre acestea va fi de tipul 802.11a prin intermediul căreia un protocol reactiv, spre exemplu DSR, ar menține principala rută, iar cea de a doua interfață ar fi de tipul 802.11b și ar avea drept scop menținerea unei căi secundare, printr-un protocol proactiv, spre exemplu DSDV. Astfel, prima rută va fi traversată de segmente de date TCP, iar cea de a doua rută va fi utilizată pentru mesajele de feedback. Atunci când ruta a fost pierdută recuperarea pachetelor va fi făcută de protocolul TCP prin intermediul căii secunde și astfel își va menține și dimensiunea ferestrei. Până la momentul restabilirii legăturii prin interfața mai performanță, comunicația se va face prin interfața secundă, cu rata de transfer mai mică. 802.11b este interfața secundă întrucât aceasta are o arie mai mare de acoperire decât 802.11a, dar o rată de transfer mai mică.

In lucrarea [15] rezultatele simulărilor au reliefat faptul că performanța rețelei crește că rata de transfer și există și un câștig în ceea ce privește întârzierile prin utilizarea interfeței secunde.

Dezavantajul principal al acestei abordări sunt limitările pe care le întâmpină în unele rețele de senzori și în unele momente datorită resurselor limitate. Presupunerea că fiecare nod să fie echipat cu două interfețe este costisitoare și uneori câștigul performanței ar putea să conteze mai puțin decât costul pe care îl implică.

2.3.3.2.1.2 Folosirea de rute multiple

Identificarea mai multor căi intre nodul sursă și nodul destinație de către protocoalele de rutare îmbunătățește performanța protocolului TCP și implicit a întregii rețele. Asigurarea mai multor căi face ca atunci când una din acestea a fost întreruptă să mai existe cel puțin o cale pregătită. Sarcina protocolului de rutare va fi aceea de a comuta cât mai repede transferul de date pe ruta de rezervă micșorând astfel impactul pe care îl are în mod normal un astfel de eveniment asupra rețelei datorat de mobilitatea crescută a nodurilor într-o rețea de senzori wireless. În acest mod protocolul TCP nu va întâmpina probleme.

Această soluție deși pare bună, în momentul implementării pot exista unele dificultăți determinate de limitările specifice nodurilor din rețelele de senzori. Aici mă refer pe de o parte la

limitările energetice, iar pe de altă parte la limitările de memorie. Este posibil să descoperim mai multe rute pentru un anumit transfer de date, însă să nu fie necesară folosirea rutei de rezervă și astfel consumul energetic este inutil.

2.3.3.2.2. Abordări la nivel TRANSPORT

2.3.3.2.2.1 TCP-Vegas [17]

O altă variantă de îmbunătățire a protocolului TCP utilizează rata de transmisie ca pârghie a mecanismului de control al congestiei și se numește TCP-Vegas prezentată pe larg în lucrarea [17]. Această abordare utilizează ca idee de bază ajustarea în funcție de rata de transmisie dorită a acesteia. Atunci când diferența dintre rata de transmisie reală și rata transmisiei dorite este mai mică decât o limită inferioara, în cursul următorului RTT, fereastra congestiei va creste liniar. Fenomenul invers, de scădere a ferestrei de congestie, se va produce când diferența depășește limita superioară.

Protocolul TCP-Vegas folosește întârzierile în rețea, față de varianta în care erau folosite informații cu privire la pierderea pachetelor, pentru a stabili rata de transmitere a datelor. Mai mult, rata medie de transfer de bună recepție este îmbunătățită prin modificarea mecanismelor de control al congestiei.

Protocolul TCP-Vegas nu a fost conceput pentru rețele wireless, însă, utilizarea acestuia îmbunătățește performantele protocolului TCP prin modificările aduse mecanismului de control al congestiei și evaluării ratei de transmisie. Numărul de pachete mic din rețea face ușoară manevrarea eventuală a congestiilor și micșorează pierderile de pachete.

Detectarea rapidă a congestiei, prin modificarea ferestrei de congestie pe bază modificării valorii RTT (Round Trip Time) este principalul avantaj al acestui protocol față de celelalte variante TCP.

2.3.3.2.2.2. RTO fix

Tehnica aceasta este concepută astfel încât să diferențieze congestionarea rețelei de o eventuală pierdere a rutei, îmbunătățind astfel performanța protocolului TCP. Această tehnică nu se bazează pe confirmarea nivelului inferior. Atunci când în urma unei transfer de pachete nu s-a recepționat nici un feedback TCP va dubla valoarea RTO și va retrimite pachetul pentru care nu a primit confirmare. Dublarea intervalului RTO se va face până când se va recepționa un feedback de primire a pachetelor și prin aceasta protocolul TCP se poate ocupa cu succes de rezolvarea congestiei. Pierderile cele mai mari într-o rețea de senzori wireless nu apar datorită congestionării rețelei și astfel protocolul TCP nu mai poate asigura nivelul de performantă adecvat. Luând în considerare faptul că rutele se pot restabili într-un timp scurt există studii conform cărora e mai bine ca în loc să se aștepte intervale din ce în ce mai mari între retransmisii, pachetele neconfirmate să fie retrimise de către protocolul TCP la intervale periodice de timp. Astfel prin această tehnică intervalul RTO rămâne nemodificat. În loc să aștepte timpi din ce în ce mai mari, exponențial mai mari, protocolul TCP va retransmite mai repede pachetele neconfirmate după restabilirea rutei. Prin această modificare întârzierile din rețea sunt înlăturate și performanța rutării este îmbunătățită. În prezența congestiei, e nevoie de mai multe analize pentru a admite ideea că două expirări de timp consecutive sunt determinate de eșecul rutelor.

2.3.3.2.3. Abordări inter-nivel ELFN [18]

Această metodă se bazează pe o colaborare între protocolul TCP și protocoalele de rutare și se numește ELFN (Explicit link failure notification technique). Protocoalele de rutare oferă informații cu privire la pierderea legăturii nivelului superior, protocolului TCP. Astfel TCP poate face diferența între congestia rețelei și pierderea legăturii de date.

In cazul unei pierderi de legătura, protocolul de rutare va trimite pe lângă notificarea specifică, un semnal ELFN ce va fi prelucrat de protocolul TCP la nodul sursă. Odată primit acest semnal, cronometrele de retransmisie vor intra într-o stare de inoperativitate. Ruta urmează a fi testată prin trimiterea periodică a unor mesaje de control. În momentul confirmării rutei, prin primirea unui mesaj corespunzător, transferul de date va fi reluat împreună cu activitatea normală a protocolului TCP. Prin acest proces atunci când transferul este reluat, acesta se va realiza la o rata mare.

Putem concluziona că această metodă este o îmbunătățire eficientă a protocolului TCP, însă are deficiența faptului că nodurile intermediare trebuie să anunțe protocolul TCP de modificările efectuate în rețea, aglomerând astfel rețeaua.

3. Protocoale de rutare

Rutarea este procesul prin care se stabilește ruta ce o vor urma datele în timpul unui transfer între două noduri ale rețelei. Această atribuție revine, conform stivei de protocoale OSI (Open System Interconnection), nivelului Rețea. Atunci când nodul destinație este departe de nodul sursă, nu este în aria de acoperire a nodului sursă, soluția pentru transferul de date este tehnică multi-salt. Majoritatea rețelelor existente sunt rețele multi-salt și într-o astfel de rețea, nodurile intermediare trebuie să se comporte asemenea unor relee pentru a transporta pachetele de date de la nodul sursă la nodul destinație. Un nod intermediar trebuie să decidă cărui vecin îi trimite pachetul de date ce nu i se adresează. Această decizie este luată, de obicei, pe baza unor tabele de rutare. Tabele de rutare conțin cei mai favorabili vecini pentru oricare din posibilele destinații ale unui pachet. Realizarea și întreținerea acestor tabele de rutare este sarcina principală a unui protocol de rutare.

3.1. Protocoale fără tabele de rutare [3]

Prima categorie este reprezentată de protocoale ce nu folosesc o tabela de rutare. Această categorie este pe cât de simpla pe atât de ineficientă.

3.1.1. Protocol de inundare

Cea mai simplă metodă de retransmitere a datelor este inundarea rețelei, adică transmiterea mesajului către toți vecinii. Astfel avem certitudinea ca, atâta vreme cât nodul sursă și destinație se află în rețea, mesajul va fi recepționat.

Acest protocol este ușor de implementat și este potrivit pentru diverse tipuri de rețele, de diferite distribuții și în diferite medii. Principalul avantaj al acestei metode de rutare este reprezentat de fiabilitatea sporită. Acest protocol are însă câteva deficiente:

- **Bucla infinită:** Există riscul ca unele pachete să circule în rețea în buclă infinită și din acest motiv fiecare nod va redirecționa doar pachetele pe care le-a primit pentru prima oara. Acest lucru presupune că pachetele să conțină un identificator al sursei și numere de ordine. O altă modalitate prin care se poate evita circulația în buclă infinită a pachetelor poate fi implementarea unui timp de expirare;
- **Implozia**: Același pachet de date ajunge în momente diferite de timp prin căi diferite la destinație fără ca nodul receptor să știe dacă este vorba de același pachet ca cel primit anterior sau de un altul. Acest lucru se întâmplă întrucât pachetele sunt difuzate tuturor nodurilor din rețea. Această problema se poate remedia prin marcarea pachetului cu un identificator unic în funcție de momentul când a fost emis;
- **Suprapunerea de pachete**: Atunci când vorbim de o rețea de senzori se poate întâmpla ca mai mulți senzori vecini să detecteze același eveniment și să trimită aceeași informație către sursă. Acest lucru poate fi în unele rețele de senzori un lucru de dorit, de asemenea poate asigura precizia, dar în altele poate reprezenta o informație redundantă.

3.1.2. Protocol de unicast

Această metodă reprezintă o alternativă la retransmiterea pachetului către toți vecinii și constă în trimiterea pachetului de date unui singur nod, ales arbitrar. Acest protocol face ca pachetul de date să hoinărească prin rețea în căutarea destinației, fără a avea nici un fel de garanție ca pachetul va ajunge la destinație. Pentru această problema o soluție ar fi ca unui pachet de date să nu i se permită să treacă de două ori prin același nod. Acest lucru ar presupune să se implementeze un sistem de distingere a pachetelor și de memorare a nodurilor prin care a trecut ceea ce nu e deloc fezabil într-o rețea de senzori în care resursele sunt limitate (energetice, capacitate de procesare, memorie). Chiar dacă un astfel de sistem ar fi implementat timpul în care un pachet ar ajunge de la sursă la destinație într-o rețea de dimensiuni mari, ar fi inacceptabil, informația devenind inutilă.

3.2. Protocoale cu tabele de rutare

Construcția tabelelor de rutare este una din atribuțiile algoritmului de rutare, prin intermediul protocolului de rutare. În rețelele clasice, cu fir, protocoalele de rutare se bazează algoritmii de evaluare a conexiunii ori pe vectori de distantă. Într-o rețea de senzori wireless multi-salt, în care nodurile sunt mobile, protocoalele de rutare trebuie să fie auto-configurabile, să facă fața schimbărilor topologice dese, să fie distribuite și să genereze eforturi suplimentare reduse.

Rețelele de senzori au stârnit un mare interes în ceea ce privește cercetările recente, iar acest lucru a dus la dezvoltarea de noi protocoale de rutare. Astfel protocoalele de rutare folosite în rețelele de senzori se clasifică în felul următor [19]:

a) protocoale de rutare plate (flat);
b) protocoale de rutare ierarhice;
c) protocoale de rutare pe bază de informații geografice;

In funcție de modul de administrare al tabelei de rutare protocoalele de rutare plate sunt împărțite în două categorii [3]:

- protocoale **pro-active** caracterizate ca fiind conservative pentru că încearcă să retină în tabele de rutare informația exactă. Exemple din această categorie sunt: Destination-Sequenced Distance Vector (DSDV), Clusterhead Gateway Switch Routing (CGSR) și Wireless Routing Protocol (WRP);

- protocoale **reactive**, ce construiesc tabelele de rutare în momentul în care e nevoie transmiterea de date către o destinație despre care nu se cunoaște nici un fel de informație de rutare. Exemple din această categorie sunt: Dynamic Source Routing (DSR), Ad Hoc Ondemand Distance Vector (AODV), Temporally Ordered Routing Algorithm (TORA), Associativity-Based Routing (ABR) și Signal Stability Routing (SSR).

De obicei, atunci când rețelele wireless cresc în dimensiuni rutarea bazată pe scheme plate își arată limitările din cauza supraîncărcării legăturilor și a nevoii de procesare intense.

Protocoalele bazate pe rutarea ierarhică își pun baza pe organizarea nodurilor în grupuri și atribuirea fiecărui nod a unei sarcini. Astfel dimensiunile tabelei de rutare și a pachetelor de actualizare sunt reduse prin utilizarea doar a unei parți din rețea și nu a întregii rețele. Cel mai utilizat mod de grupare a nodurilor este în clustere în funcție de poziția lor. Comunicarea cu alte clustere este realizată de nodul lider numit și clusterhead. [19]

Un alt mod de realizare a ierarhiei este cea implicită în care fiecare nod are o arie locală de aplicare.

In ceea ce privește rutarea pe baza informațiilor geografice putem spune că este posibilă prin utilizarea GPS-ului (Sistem de Poziționare Globala) și datorită evoluției acestui sistem ce oferă o precizie foarte bună în localizarea obiectelor. Astfel se obțin îmbunătățiri semnificative în funcționarea rețelelor de senzori. [19]

Protocoalele plate sunt cele mai dezvoltate ce funcționează cu performanțe ridicate ce se adresează unei game largi de aplicații și din acest motiv am ales studierea în continuare a acestui tip de protocoale.

3.2.1. Protocoale pro-active

Această categorie de protocoale mai este numită și table-driven întrucât sunt protocoale ce pun accent pe tabela de rutare. Atunci când se dorește efectuarea unei transmisii, întârzierile cauzate de căutarea rutelor către nodul destinație sunt evitate întrucât nodul destinație are disponibile informațiile de rutare. Algoritmii clasici utilizați în Internet, cum ar fi algoritmi vector-distanță, algoritmi de stare a legăturii, se pot regăsi și în cazul rețelelor de senzori. Protocoalele clasice precum RIP, OSPF nu sunt adecvați pentru aplicarea în rețelele de senzori și din acest motiv există dezvoltate mai multe variante optimizate. [21]

In cazul algoritmilor de tip vector distanță, fiecare nod are un vector cu distanțele actuale până la oricare nod cunoscut. Acest vector este difuzat tuturor vecinilor unui nod. Astfel prin informațiile pe care le trimite și prin informațiile pe care le primește, fiecare nod calculează pentru fiecare destinație drumul cel mai scurt. Până când rețeaua atinge o stare stabila, procesul de calcul se repeta, iar în cazul apariției de noi informații distanțele se pot modifică. Algoritmul Bellman-Ford stă la baza acestei metode prin care se calculează căile optime.

Problemele principale ale acestei metode constau în faptul că rețeaua este supraîncărcată cu mesaje de control ceea ce poate duce la congestionarea acesteia și că timpul necesar actualizării este foarte mare. O altă problemă importantă pentru utilizarea acestui algoritm în rețelele de senzori este posibila apariție a buclelor infinite cauzată de necoordonarea nodurilor. Se pot face modificări în tabela de rutare folosind informații eronate. Astfel un pachet de date poate intra într-o buclă infinită ceea ce înseamnă un consum al resurselor inutil. Pentru rețelele de senzori, în care există o limitare importantă a resurselor, această eroare constituie o risipă importantă și inacceptabilă a resurselor energetice și a capacității de calcul. [21]

Pe lângă aceste lucruri, în rețelele de senzori nodurile au o mobilitate ridicată, iar acest lucru va face ca atingerea stării de convergență să dureze mult. Astfel performanțele rutării sunt mult reduse.

Rezultatul multor eforturi făcute pentru a optimiza algoritmul Bellman-Ford pentru rețelele de senzori au dus la apariția algoritmului DSDV (Destination-Sequenced Distance-Vector), ce reprezintă unul din cele mai utilizate protocoale. [21]

In cadrul algoritmilor stării de legătura este prezentat un alt mecanism prin care se face schimbul informațiilor de rutare. Fiecare nod are o viziune proprie asupra rețelei, fiind incluse informații despre topologie, starea legăturilor și interfețele de ieșire proprii. Fiecare nod trimite periodic informații prin inundare despre starea legăturilor cu vecinii proprii către toate nodurile din rețea. Acest lucru se întâmplă pentru ca imaginea rețelei să fie una reala. Nodul ce primește un astfel de mesaj, actualizează informațiile topologice și utilizează pentru fiecare destinație un algoritm de calcul al căilor optime. Algoritmul Dijkstra este cel mai utilizat în astfel de operații. [21]

In cazul acestui tip de algoritm nu există problema buclei infinite, iar convergența rețelei se atinge mai ușor. Aceste avantaje fac ca această metodă să fie mai uzitată în rețelele de senzori.

Dezavantajele acestei metode sunt reprezentate de spațiul mare de stocare de care are nevoie pentru a menține imaginea globală a rețelei pentru fiecare nod și numărul mare de mesaje de control ce duc la suprasolicitarea rețelei.

Acești algoritmi de rutare constituie baza pentru protocoalele de rutare proactive, fiind adaptați rețelelor de senzori.

Avantaje [20]:

- disponibilitate rute;

- întârzieri mici.

Dezavantaje [20]:

- consumul ineficient al resurselor;
- lipsa de scalabilitate;
- convergența întârziată în cazul rețelelor de dimensiuni mari.

3.2.1.1. Protocolul DSDV [22]

Protocolul DSDV (Dynamic Destination Sequenced Distance Vector) aparține lui C. Perkins și a lui P. Bhagwat și a fost special conceput pentru o mobilitate crescută a rețelei. Acest protocol are la bază clasicul algoritm Bellman-Ford la care s-au adăugat câteva îmbunătățiri. Acest protocol a avut ca obiectiv construirea unei rețele de dispozitive mobile ce să poată schimba date fără necesitatea unei stații de bază. Mai mult, este oferită posibilitatea conectării la alte tipuri de rețele devenind astfel unul dintre cei mai utilizați algoritmi în rețelele de senzori.

Acest protocol presupune faptul că nodurile schimbă periodic pachete de control. Fiecare nod trimite propria tabelă de rutare prin intermediul acestor mesaje de control. În tabela de rutare sunt stocate pentru toate destinațiile posibile nodul vecin căruia trebuie să i se transmită pachetele pentru a ajunge la destinația dorită printr-un drum optim. Intrările tabelei de rutare au un număr de ordine individual (SN).

Pentru a menține consistența datelor din tabelele de rutare se trimit periodic mesaje de control sau atunci când s-a detectat o modificare topologică. Aceste mesaje de control conțin destinațiile posibile ale unui pachet de date ale unui nod cât și distanțele până la acestea. Rutele sunt marcate prin intermediul unui număr de secvență, iar gradul de noutate al acestora este stocat într-un alt câmp. Acest trafic prin care se realizează controlul rețelei se poate transmite prin adrese MAC, de nivel 2, sau prin adrese IP, de nivel 3.

3.2.1.1.1. Manipularea tabelei de rutare și transferul de pachete

După cum am precizat, protocolul DSDV presupune că fiecare nod are o tabelă de rutare prin intermediul căreia poate trimite pachete de date către oricare nod vizibil al rețelei. Chiar și atunci când se dorește trimiterea de pachete către un nod ce nu se află în raza de acțiune a sursei, acest lucru este posibil prin rutarea multi-salt. Tabela de rutare este accesată de fiecare nod ce dorește să transmită pachete pentru a verifica ce vecin trebuie să preia datele pentru a parcurge drumul optim spre destinație.

La fel ca la metodele consacrate ce folosesc metode vector distanță, mesajele de actualizare asigură consistența rutelor. La un anumit interval de timp sau atunci când se produc modificări ale topologiei fiecare nod trimite informațiile de rutare către celelalte noduri. Astfel va putea fi asigurată atât o comunicație sigură între două noduri cât și calitatea legăturilor rețelei.

Fără a se tine cont de metoda prin care se trimit mesaje de actualizare, la un moment dat, informația de rutare proprie fiecărui nod va fi anunțată de acesta fie prin broadcast, fie prin multicast a tabelelor de rutare.

Spre deosebire de algoritmii clasici, aici, nodurile sursă trimit pachete de date cu metrica inițială 1 ceea ce arată faptul că nodurile ce vor recepționa acest pachet se află la un salt distanță față de nodul inițial. Tabelele de rutare ale vecinilor sunt actualizate, dacă este cazul, apoi metrica este incrementată cu o unitate, iar pachetul de actualizare va fi trimis mai departe corespunzător cu tabela de rutare a nodului curent. Procesul este oprit atunci când toate nodurile rețelei au primit o instanță a pachetului de actualizare având metrica corespunzătoare. Modificarea tabelelor de rutare nu se realizează instant. Mai întâi este așteptată obținerea celui mai bun drum spre o anumită destinație, iar actualizările sunt păstrate într-un buffer pentru un anumit timp. În cazul în care în timpul de așteptare sunt recepționate mai multe pachete de informații de actualizare către aceeași destinație prin mai multe căi se vor alege drumurile având metrica cea mai mică. Sunt preferate rutele cu cele mai recente numere de secvență, acesta fiind un alt criteriu de alegere a unei rute. Modificarea rutelor ce urmează a fi updatate poate fi întârziată până la stabilizarea rețelei, până când a fost găsită ruta optimă minimizând retrimiterea pachetelor de actualizare și a variațiilor tabelei de rutare.

Este de dorit păstrarea consistenței datelor din tabela de rutare chiar și în cazul modificărilor frecvente ale topologiei, iar pentru acest lucru tabelele de rutare sunt modificate dinamic. Astfel informația de rutare trebuie să fie actualizată rapid și frecvent pentru a asigura transferurile end-to-end.

Nodurile din rețelele de senzori trebuie să trimită vecinilor tabela de rutare periodic. Este foarte probabil ca acest transfer să se realizeze utilizând mai multe pachete. Atunci când nu există modificări ale poziției nodurilor și implicit ale topologiei rețelei acest tip de mesaje sunt rare. Modificările minore ce pot apărea în rețea sunt anunțate prin comunicarea prin mesaje de actualizare incrementale, între două trimiteri succesive ale tabelei de rutare. De regulă astfel de mesaje încap într-un singur pachet, însă cu cât mesajele incrementale cresc în dimensiune cu atât schimbările din rețea sunt mai semnificative.

3.2.1.1.2. Managementul schimbărilor topologice

Faptul că nodurile din rețelele de senzori au o mobilitate crescută poate duce la ruperea legăturilor. Detecția acestor pierderi de rute se poate face ori prin hardware-ul comunicației ori prin mesajele trimise de noduri. O astfel de legătură va avea metrica infinit, acest lucru fiind setat atunci când legătura se pierde. Tot atunci se va actualiza și numărul de secvența. Evenimentele de acest fel modifică într-un mod semnificativ topologia, iar nodul ce detectează astfel de schimbări va anunța acest lucru către toate celelalte noduri ale rețelei împreună cu actualizările referitoare la noile căi. Acest lucru se realizează prin faptul că orice nod mobil, exceptând destinația, va genera un număr de secvență mai mare decât cel mai recent ce a ajuns la destinație. Astfel metrica împreună cu numărul nou de secvență vor fi împachetate și răspândite în toată rețeaua. Nodurile vor genera pentru ele numere de secvență pare, iar vecinii vor genera numere impare pentru nodurile ce răspund modificărilor legăturii, pentru a evita producerea de conflicte a numerelor de secvență în momentul schimbărilor topologice.

Nodul cu care s-a pierdut legătura o va restabili atunci când își va evidenția prezența prin mesaje de actualizare având metrica și numărul de secvență corespunzătoare. Mesajele de acest gen vor fi difuzate în rețea pentru ca toate nodurile să fie înștiințate. Datele ce prezintă metrica finită și un număr de secvență mai recent vor înlocui intrările din tabela de rutare ce conțineau metrica infinită. [23]

3.2.1.1.3. Câmpurile mesajului de actualizare

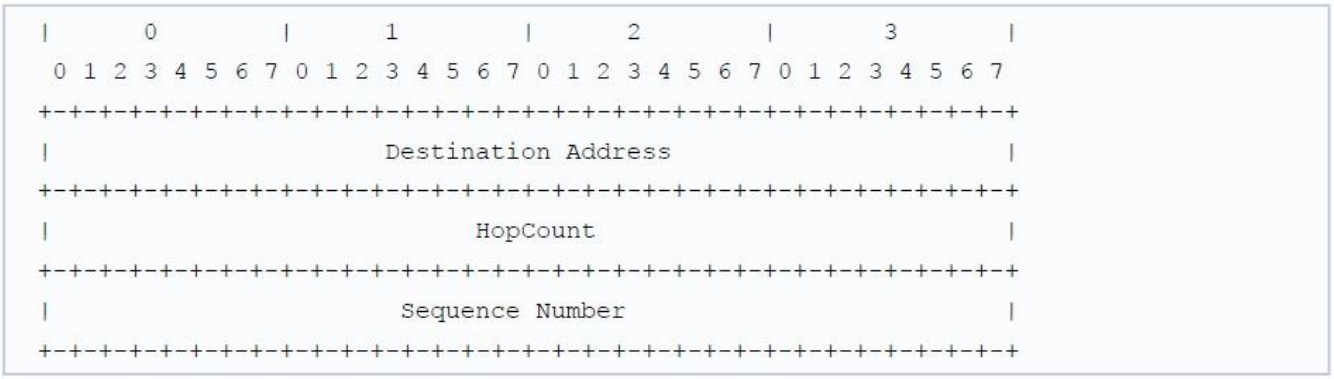


Figura 3.1 Câmpurile mesajului de actualizare [24]

Câmpurile conținute de antetul unui mesaj de actualizare sunt:

- adresa destinației;
- contor de salturi;
- numărul secvența;

3.2.1.1.4. Dezavantaje ale protocolului DSDV

Protocolul DSDV are la bază principii ale funcționarii rutării cu vector distanță și se dorește a fi corespunzător pentru utilizarea în rețelele de senzori. Față de varianta clasică, cei ce au gândit acest protocol au dorit eliminarea problemei buclei infinite de rutare ceea ce au și reușit prin folosirea numărului de secvență care se incrementează, dar și prin mesajele de actualizare. Aceste mecanisme de combatere a buclelor pot cauza, însă, variații ale căilor, ce duc la scăderea performanțelor din rețelele de senzori. [23]

Atunci când o cale are un număr de secvență mai mare este aleasă în defavoarea căilor cu numere de secvență mai mică, iar atunci când numerele de secvență sunt egale, dar există metrici diferite va fi aleasă calea cu o metrică mai mică.

Existența multor noduri mobile, scenariu plauzibil în cazul rețelelor de senzori, ce trimit actualizări cu o frecvență diferită și într-un mod independent e posibil ca un nod să primească mesaje astfel încât va schimba intrările din tabelul de rutare în mod constant, chiar și atunci când nu e cazul.

Astfel vor fi emise mesaje de actualizare ce nu sunt utile, ba chiar încurcă. Rezolvarea acestei probleme a venit prin introducerea unui timp de așteptare dintre momentul modificării tabelei de rutare și momentul transmiterii pachetelor de actualizare ce conțin modificări noi. Determinarea acestui timp este greu de făcut și cu cât este mai mare cu atât creste probabilitatea apariției de întârzieri sau nepotriviri în rețea.

O problemă importantă a protocoalelor vector distanță nu a fost eliminată nici în protocolul DSDV. Această problemă este clasică pentru mediile wireless și este cea a legăturilor unidirecționale. Protocolul DSDV face presupunerea că legăturile sunt bidirecționale, dar este posibil ca acest lucru să nu fie valabil și în realitate. Problema legăturilor unidirecționale apare datorită puterii diferite de transmisie a nodurilor ceea ce face posibilă trimiterea de date de la un nod către alt nod, însă nu și invers. Această problemă este una importantă ce poate avea un impact negativ destul de major asupra rețelei. [23]

Alte probleme ale acestui protocol ar fi faptul că nu suportă rutare prin multiple căi și faptul că mesajele de actualizare periodică suprasolicită rețeaua, dar și faptul că o tabelă de rutare completă trebuie menținută de fiecare nod.

3.2.2. Protocoale reactive

Protocoalele de rutare on demand (la cerere) sau protocoale reactive oferă o alternativă la protocoalele clasice ce salvează tabela de rutare pentru orice destinație posibilă. Aceste protocoale nu mai memorează distanțele pentru toate destinațiile. Atunci când un nod de rețea dorește să efectueze un transfer de date către un alt nod, calea va fi creată și menținută pe tot parcursul transferului. Odată cu încheierea comunicației, ruta va fi ștearsă. [21]

Practic, fiecare rută trebuie mai întâi descoperită. Pentru acest lucru nodul sursă, în căutarea rutei, va inunda rețeaua cu cereri de cale prin multi-salt. Atunci când a fost găsit nodul destinație acesta va răspunde printr-un mesaj corespunzător utilizând ruta tocmai găsită. Atâta timp cât este utilizată ruta este salvată la nodul sursă. Din cauza mobilității ridicate a nodurilor și datorită schimbărilor topologice dese trebuie să existe un proces continuu de actualizare a căii dintre sursă și destinație. [21]

Avantaje [20]:
- dimensiuni reduse ale tabelelor de rutare;
- număr mic al mesajelor de control ;

Dezavantaje [20]:
- întârzieri datorită căutării rutei;
- congestii cauzate de inundarea rețelei;

3.2.2.1. Directed Diffusion

Acest protocol este primul protocol propus special pentru rețelele de senzori wireless și este un protocol centrat pe date. Această metodă este compusă din câteva elemente: interese, pachete de date și direcții.

Pentru început, nodul destinație cere date printr-o cerere în care își exprimă interesele. Un mesaj de interes reprezintă o interogare a nodului de la care se doresc informații. Datele sunt adresate folosind perechea atribut-valoare și sunt colectate și prelucrate în funcție de interesul exprimat de un anumit nod. Interesele unui anumit nod sunt transmise prin inundare întregii rețele. Ori de câte ori un nod primește un interes, verifică dacă acel interes a mai fost exprimat sau nu. Dacă este un interes nou nodul va trasa o direcție către expeditorul acelui interes pentru a satisface acel interes. Cu alte cuvinte atunci când există un interes nou se stabilește o legătură intre nodul ce și-a manifestat interesul și nodul ce poate satisface acel interes. După această etapă nodurile își satisfac interesele între ele prin transfer de pachete de date. Protocolul este optimizat fie pentru întârzieri minime, fie pentru un număr maxim de pachete primite.[33]

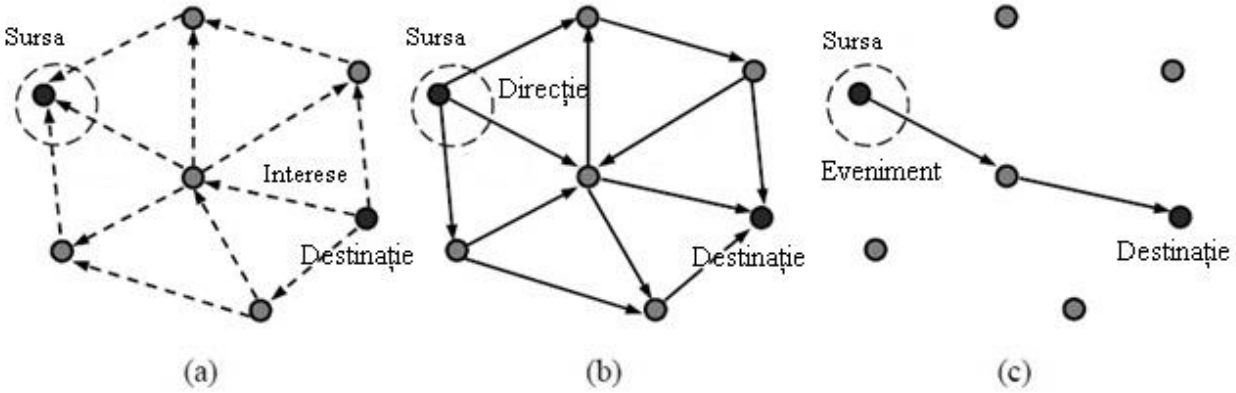


Figura 3.2 Directed Diffusion: a) propagarea intereselor, b)stabilirea direcțiilor, c) transferul de date [34]

In cadrul acestui protocol datele sunt descrise folosind perechi atribut-valoare, fiecare atribut fiind asociat cu un interval de valori. Valoarea atributului poate fi orice subset din gama sa. În funcție de sistemul și aranjarea aleasă pot afecta înțelesul sarcinilor și performanța difuziei. Interesul este, de obicei, exprimat de nodul colector și pentru fiecare sarcina, periodic, va difuza interesele sale tuturor vecinilor săi. Intervalul atributului specifică rata de date a evenimentului. Întrucât poziția sursei nu se cunoaște cu exactitate, interesele trebuie să fie difuzate cu o gamă mai mare decât cea reglementată de posibila sursă a informației. Ca urmare, în cazul în care nodul colector a ales o frecvență mai mare de date decât cea reala, va exista un consum energetic mai mare decât cel minim necesar. Interesul poate fi periodic actualizat de nodul colector și e necesar să se întâmple acest lucru, pentru a evita erorile, pentru că acestea nu sunt transmise într-un mod sigur. Rata cu care se produc aceste actualizări de interes nu sunt fixe și depind de aplicație. Fiecare nod menține o lista cu interesele distincte recepționate. Nodul ce furnizează informații nu deține informații cu privire la cine a emis interesul respectiv ci doar informații despre nodul vecin prin care se realizează legătura. [34]

Un interes conține câteva câmpuri [33]:

- momentul primirii ultimului interes;
- direcția din care a venit;
- rata de date;
- timpul de viață;

3.2.2.2. Protocolul AODV

Protocolul AODV (Ad-Hoc Ondemand Distance Vector) este varianta reactivă a protocolului DSDV ce a fost gândit pentru a mări performanța protocolului pro-activ. După cum îi spune și numele, rutare cu vector distanță la cerere, are la bază ideea de vector distanță.

Acest protocol a fost astfel gândit încât să rezolve problemele protocolului DSDV, dar și pentru a acoperi o parte din problemele wireless din acel moment.

Un prim obiectiv a fost micșorarea mesajelor de difuzare și a întârzierilor din comunicații, probleme importante în rețelele ce aveau ca protocol de rutare DSDV, în special atunci când dimensiunea acestora creștea. Protocolul AODV prezintă mari avantaje pentru rețelele cu multe noduri.[25]

De asemenea o problemă importantă a mediilor wireless este aceea a terminalelor ascunse ce avea loc din cauza gamelor de transmisie mici ale mediilor. Un nod ce folosea astfel de game și beneficia de mobilitate crescută putea trimite către nodurile vecine mesaje de difuzare, dar nu putea detecta nici un vecin și nici nu putea fi detectat. În acest caz puteau să apără coliziuni întrucât atunci când două noduri vecine ar dori să comunice cu respectivul nod nici unul nu știe când poate emite. Un caz nefericit ar fi când ar încerca să transmită în același timp, iar algoritmii de evitare a coliziunilor (CSMA/CA) nu ar funcționa. [25]

Utilizând protocolul AODV orice nod ce dorește să transmită pe mai multe canale o poate face. Se precizează și faptul că protocolul AODV poate fi utilizat atât în rețele wireless cât și în rețele cablate. De asemenea acest protocol permite și rutarea multi-salt între noduri ce doresc să comunice prin folosirea unor extensii și evită inconvenientul buclelor infinite, o problemă a oricărui protocol ce folosește algoritmul Bellman-Ford. [25]

Asemenea protocolului DSDV acest protocol utilizează rutarea bazată pe vector distanță. Spre deosebire de protocolul pro-activ, AODV nu menține rutele în permanență către toate destinațiile. Acesta descoperă rutele atunci când trebuie utilizate.

Identificarea vecinilor se poate face folosind mai multe metode. Cea mai folosită tehnică este prin transferul de mesaje de tip „HELLO”. Organizarea tabelelor de rutare a nodurilor vecine sunt gândite astfel încât întârzierile să fie minime la schimbări topologice locale.

Țintele principale ale algoritmului sunt [25]:

- descoperirea rutelor prin difuzare de pachete numai atunci când este nevoie;
- deosebirea între topologia locală și cea globala;
- trimiterea mesajelor de actualizare în cazul schimbărilor topologice locale către nodurile vecine care au nevoie de astfel de actualizări.

Mecanismele ce stau la baza protocolului AODV sunt aflarea rutelor și păstrarea rutelor, ce sunt utilizate și de protocolul DSR. Pe lângă acestea este folosită rutarea multi-hop și împrumutare principiului numerelor de secvență de la protocolul DSDV. Cele mai recente informații de rutare sunt

reținute de AODV prin principiul numerelor de secvență. Față de protocolul DSDV, acest protocol reține numerele de secvența monoton crescătoare utilizate pentru înnoirea rutelor.

Spre deosebire de protocolul DSR, AODV stabilește dinamic intrările în tabelele de rutare la nodurile intermediare. În cazul protocolului DSR, rețeaua este încărcată prin faptul că informații legate de sursă sunt conținute de pachete.

Aceste tehnici utilizate în comun duc la un algoritm ce utilizează în mod eficient lărgimea de bandă, reacționează bine la modificări și evită buclele de rutare.

3.2.2.2.1. Căutarea rutelor

Aflarea rutelor este un proces ce se declanșează atunci când nodul nu dispune de o cale către destinație. Un pachet RREQ (route request) este lansat de nodul ce dorește să efectueze o transmisiune. Pachetele de acest fel conțin adresa IP a sursei, adresa IP a destinației și numărul de secvență. De asemenea există și un identificator de difuzare ce se incrementează la fiecare apel al unei cereri de rută efectuat de sursa. În acest mod se creează un identificator unic al pachetului RREQ, alcătuit din adresa IP a sursei și identificatorul de difuzare. După ce a fost creat un astfel de pachet, acesta este trimis la toate nodurile din rețea, iar apoi se contorizează timpul în care sosesc răspunsurile. [36]

Atunci când un nod primește un RREQ, este configurată calea inversă în tabela sa de rutare. Aceasta conține adresa IP a nodului curent, numărul de secvență, numărul de hop-uri până la nodul sursă, dar și adresa IP al ultimului nod prin care a trecut pachetul. În acest mod nodul va ști în ce direcție să trimită răspunsul către nodul sursă în momentul în care va fi trimis.

Răspunsul unei cereri de rută poate fi efectuat dacă nodul curent cunoaște o cale către destinație. Numărul de secvență alocat acelei destinații trebuie să fie mai mare sau egal decât cel din pachetul de cerere de ruta, RREQ. Prin această cerință este prevenită crearea de bucle, garantând că ruta propusă este suficient de noua. În momentul când cele două cerințe sunt îndeplinite, nodul trimite răspunsul către nodul ce a declanșat cererea. În cazul în care informații privitoare la destinație nu pot fi oferite, numărul de hop-uri atribuit pachetului cerere este incrementat și trimis nodurilor vecine. Natural, răspunsul este oferit în toate cazurile unei cereri de ruta, RREQ. [36]

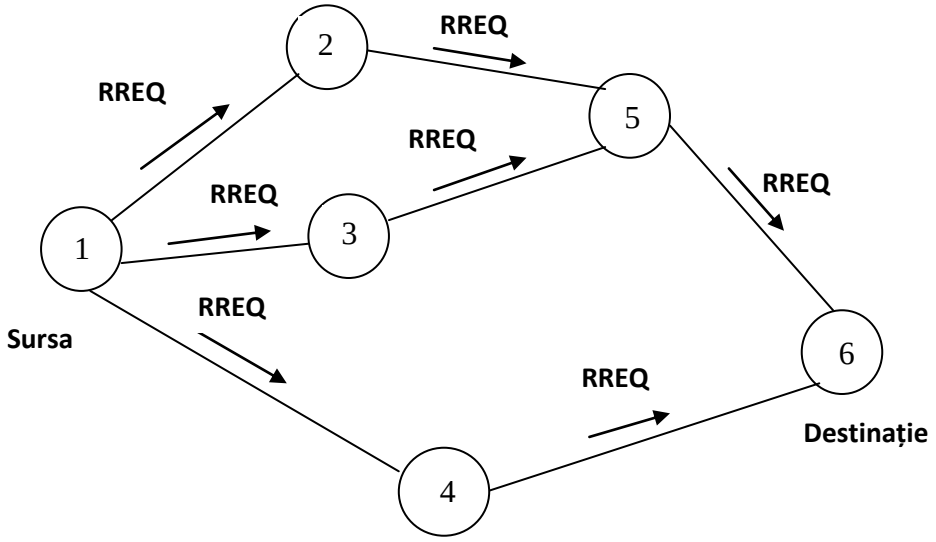


Figura 3.3 Trimiterea cererilor de rută

Atunci când pachetul cu cererea RREQ se pierde, o noua încercare de căutare este lansată. Numărul optim de încercări suplimentare este 2 pentru depistarea unei destinații. în cazul rețelelor de dimensiuni reduse trimiterea unui pachet RREQ nu afectează rețeaua, având o influență redusă. Atunci când vorbim de rețele de dimensiuni mari, efectele acestei acțiuni sunt vizibile prin încetinirea fluxului și apariția inevitabilă de întârzieri. Controlul cererilor de rută RREQ în rețelele ce au dimensiuni mari se face prin creșterea treptată a zonei de căutare, prin tehnica cercului de rază crescătoare. Se pornește de la o arie mică a cercului și dacă destinația nu a fost identificată se mărește progresiv aria și se reîncepe căutarea. Pachetul RREQ are o valoare TTL (time to live), ce arată timpul de expirare setată inițial la o anumită valoare. Această valoare este incrementată în momentul în care, în intervalul de timp alocat, nici un răspuns nu a fost recepționat. Într-un caz limită, atunci când valoarea timpului de viață trece de maxim, rețeaua este inundată cu cereri de rută de un număr egal de ori cu valoarea „rreqq_retries”. [36]

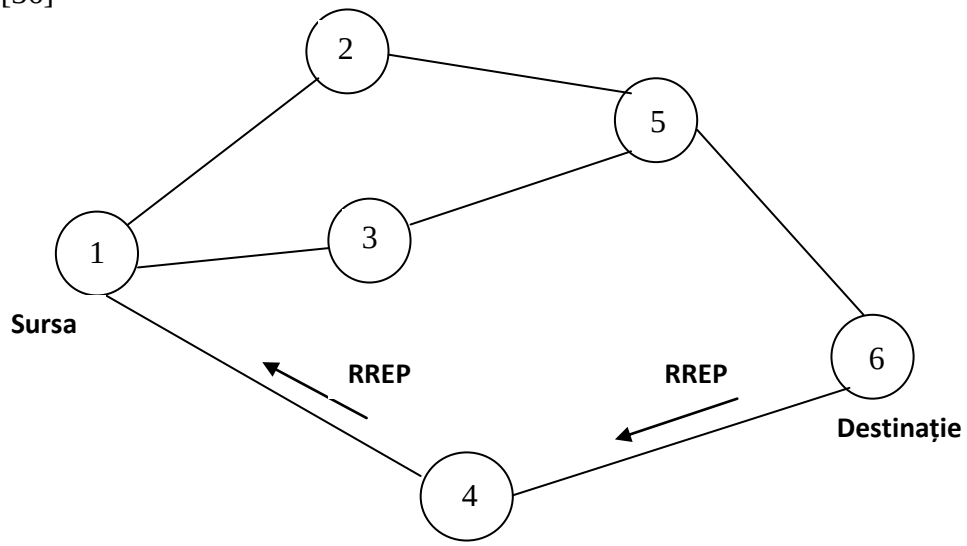


Figura 3.4 Modul de propagare a mesajului răspuns

Răspunsul la cererea de rută este formulat de un nod care decide că ruta sa este suficient de nouă și creează astfel un mesaj RREP. Dacă răspunsul este formulat de un nod intermediar între sursă și destinația pachetului RREQ, acesta atașează propriile date referitoare la numărul de secvență al destinației, numărul de hop-uri până la destinație și timpul pentru care ruta este menținută. Dacă nodul destinație al pachetului RREQ este în situația de a răspunde, numărul de secvență este atașat, numărul de salturi este inițializat cu 0 și setează timpul în care ruta este validă. Pachetul RREP este trimis ca răspuns la cererea de ruta RREQ prin intermediul nodului sau nodurilor prin care a ajuns această cerere. Nodul intermediar ce recepționează un pachet RREP va seta o rută către destinație în tabela sa de rutare. Ruta va memora adresa IP a destinației, adresa IP a nodului prin intermediul căruia a primit mesajul, dar și distanța până la destinație. [36]

Răspunsul la trimiterea pachetului RREQ prin care este căutată o rută către o anumită destinație poate avea mai multe rezultate sosite de la diferite noduri vecine. Primul răspuns RREP este, de obicei, luat în considerare și numai atunci când numărul de secvență al unui pachet ulterior este mai mare sau există un număr mai mic de salturi va fi trimis mai departe răspunsul întârziat. Alte pachete în afară de

cel ce se trimite mai departe vor fi pierdute pentru a scădea numărul de mesaje RREP trimise sursei și asigurând astfel informații actualizate sursei.

După recepționarea răspunsului RREP, nodul ce a emis pachetul RREQ poate comunica instant cu nodul destinație având opțiunea de a modifica ruta dacă una mai avantajoasă va fi descoperită.

3.2.2.2.2. Păstrarea rutelor

Atunci când o ruta ce leagă nodul sursă de nodul destinație a fost identificată, aceasta se menține pe timpul comunicației celor două noduri. Mobilitatea nodurilor afectează doar rutele active, adică acele căi ce leagă nodurile care comunica. Orice modificare a unor noduri ce nu fac parte dintr-o rută activă nu impune apelul protocolului de rutare. În schimb, atunci când unul din nodurile ce este implicat într-o rută activă își modifică poziția un mesaj eroare RERR este trimis nodurilor afectate. Mesajul de eroare este expediat atunci când pierderea conexiunii este sesizată de către nodul ce depistează acest lucru. Odată cu mesajul de eroare RERR este trimisă și o listă cu nodurile afectate de această modificare. Acest pachet de informații primit de vecinii nodului detector face ca rutele afectate să fie notate ca fiind neaccesibile, configurează distanța la o valoare infinit și înștiințează mai departe nodurilor precedente. În acest caz dacă este dorită continuarea comunicării între noduri este reinițializată procedura de explorare a rutelor. [36]

Căile cărora metrica le-a fost setată ca fiind infinit nu sunt șterse instant ci sunt menținute o perioadă de timp întrucât pot conține informație de rutare folositoare. Astfel AODV consideră că orice informație de rutare este importantă pentru o regiune a rețelei de senzori întrucât au fost consumate resurse ce sunt costisitoare, pentru descoperirea lor. Pentru algoritmii de reparare locală acest lucru este important, reducând schimbul de mesaje și implicit traficul pentru reconstrucția rutelor.

In figura 3.5 sunt ilustrate etapele prin care trece rețeaua în cazul pierderii de rută atunci când se folosește protocolul AODV. În prima etapă rețeaua are o conexiune ce leagă nodul sursă de nodul destinație. Această conexiune traversează nodul 5 ce urmează să își schimbe poziția în etapa a doua. Acum când nodul 5 nu mai este în aria nodului 3, acesta din urmă trimite nodului sursă un mesaj eroare. Întrucât transmisia de date nu a luat sfârșit și pentru că a primit mesajul de eroare de la nodul 3, nodul sursă reapelează procedura de căutare a rutei către nodul 6. În final este prezentată ruta prin care se reia transferul datelor între cele două noduri prin intermediul nodului 4.

3.2.2.2.3. Tabela de rutare [26]

Tabela de rutare are următoarele câmpuri ce sunt completate pentru fiecare înregistrare:

- Adresa IP destinaţie;
- Numărul de secvenţă destinaţie;
- Numărul de salturi până la destinaţie;
- Următorul salt – vecinul care a fost identificat pentru transmiterea pachetelor spre respectiva destinaţie;
- Durata de viaţă – timpul pentru care calea rămâne validă;
- Lista vecinilor activi – vecinii ce comunică prin acea cale la acel moment.

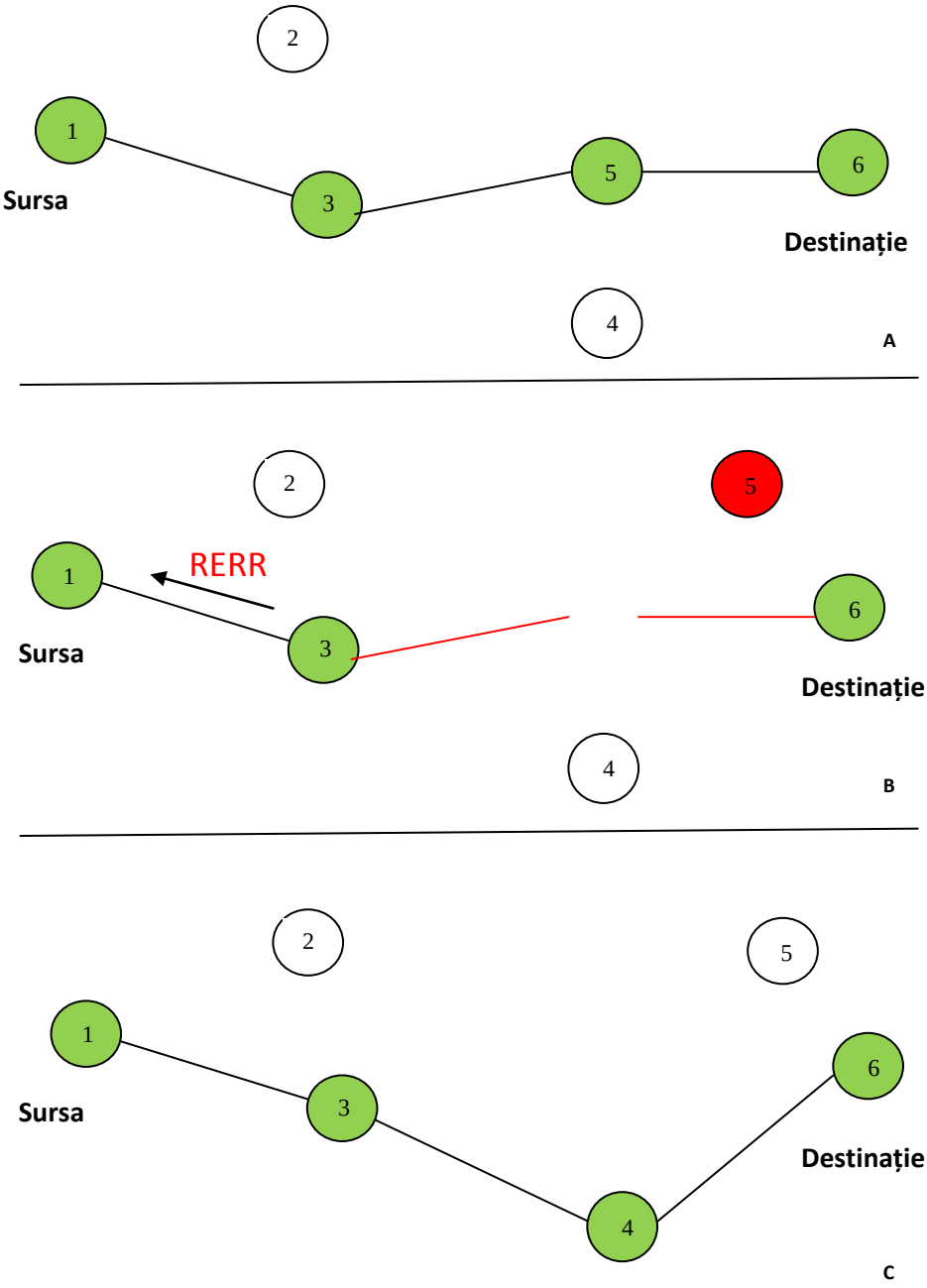


Figura 3.5 Mentenanţa rutelor

3.2.2.2.4. Formatul pachetelor protocolului AODV

Protocolul AODV folosește trei tipuri de pachete :

- cerere de rută (RREQ);
- răspuns la cererea de rută (RREP);
- eroare de rută (RERR).

Adresarea se face asemeni oricărei reţele și se folosesc adrese IPv4, iar ca protocol de transport se utilizează TCP.

3.2.2.1.4.1. Formatul pachetelor RREQ [36]

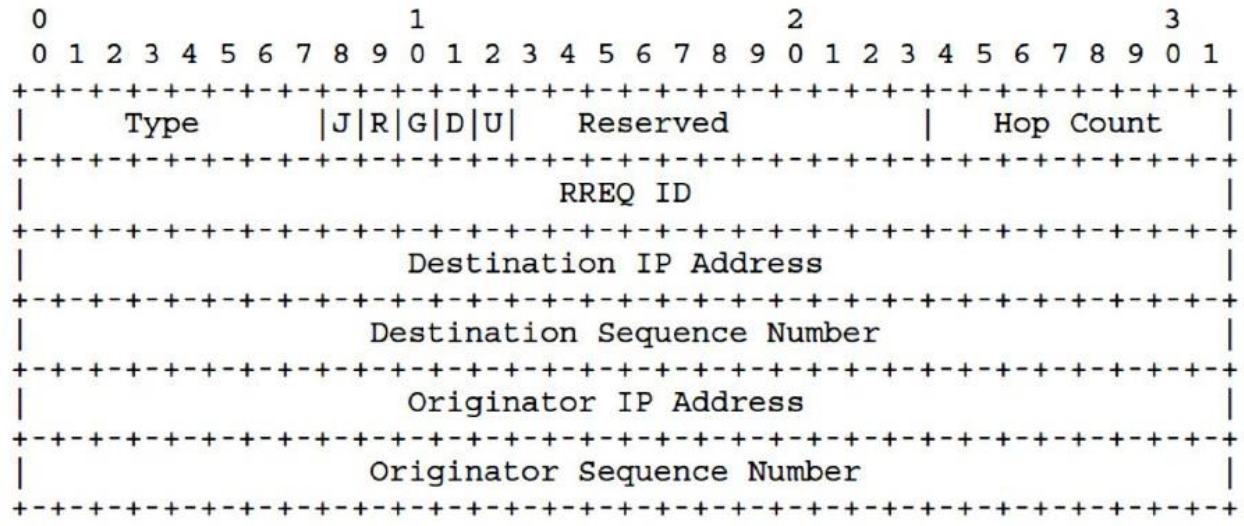


Figura 3.6 Antetul mesajului de cerere a rutei

Câmpurile ce alcătuiesc mesajul RREQ sunt:

- Tip – mesajele de cerere a rutei au valoarea câmpului Tip 1;
- J – fanionul Join, utilizat pentru multicast;
- R – fanionul Repair, utilizat pentru multicast;
- G – fanionul pentru Route Reply Free, arată dacă un mesaj RREP liber trebuie transmis ca unicast către adresa IP destinaţie;
- D – fanionul pentru Destinaţie, arată faptul că numai nodul destinaţie poate să răspundă la acest pachet RREQ;
- U – număr de secvenţă necunoscut, arată faptul că nu este știut numărul de secvenţă al destinaţiei;
- Rezervat – este 0, ignorat la primire;
- Numărul de hopuri – numărul de salturi de la IP-ul nodului sursă până la cel care procesează acest mesaj;
- RREQ ID – numărul de secvenţă ce identifică împreună cu adresa IP a sursei în mod unic pachetul RREQ;

- Adresa IP destinaţie – adresa IP a nodului destinaţiei pentru care se vrea aflarea unei căi;
- Număr de secvenţă al destinaţiei – ultimul număr de secvenţă primit de către nodul sursă al acestui pachet de la respectiva destinaţie în trecut;
- Adresa IP sursă – adresa IP a sursei ce a iniţiat pachetul RREQ;
- Număr de secvenţă al sursei – numărul de secvenţă curent ce trimite la emiţătorul mesajului RREQ.

3.2.2.1.4.2. Formatul pachetelor de răspuns RREP [36]

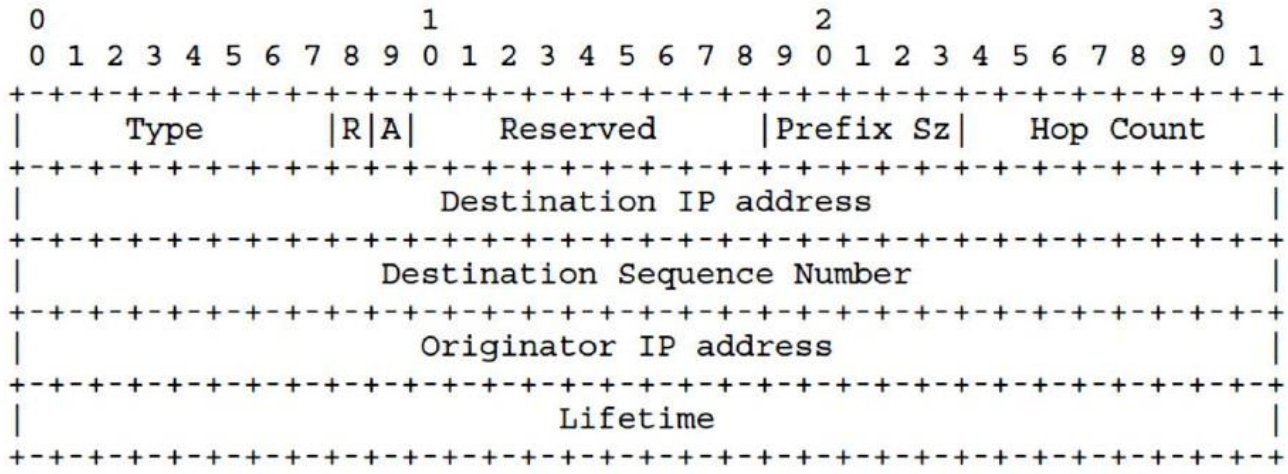


Figura 3.7 Antetul mesajului de răspuns

Antetul pachetelor răspuns, ilustrat în figura 3.7 este alcătuit din câmpurile:

- Tip – mesajele Route Reply, au valoarea câmpului Tip 2;
- R – fanionul Repair, utilizat pentru multicast;
- A – confirmare indispensabilă;
- Mărimea prefixului – atunci când este diferit de 0, cei 5 biţi ai acestui câmp arată faptul că următorul nod poate fi utilizat de toate nodurile ce au acelaşi prefix de rutare cu adresa IP destinaţie;
- Numărul de salturi – Numărul de salturi de la adresa IP a sursei până la adresa IP a destinaţiei. Pentru rute de multicast, arată numărul de salturi până la rădăcina arborelui de multicast care a declanşat mesajul RREQ;
- Adresa IP a destinaţiei;
- Număr de secvenţă a destinaţiei – numărul secvenţă alocat căii;
- Adresa IP a sursei – adresa IP a nodului care a declanşat mesajul RREQ;
- Durata de viaţă – timp măsurat în milisecunde pentru care nodurile care ce primesc mesajul RREP percep calea validă;

3.2.2.1.4.3. Formatul pachetelor eroare RERR [36]

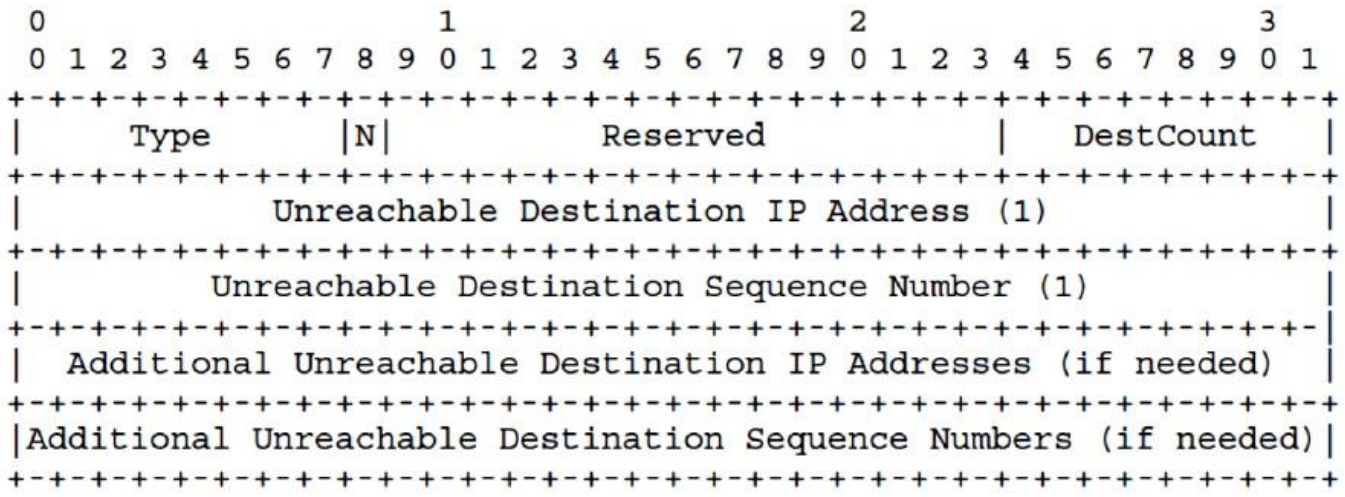


Figura 3.8 Antetul mesajelor eroare

Antetul mesajelor de eroare ilustrat mai sus este alcătuit din următoarele câmpuri:

- Tip – Mesajele Route Error, au valoarea 3;
- N – flagul No Delete, este setat când un nod a reușit o recuperare locală a unei rute și nodurile upstream sunt anunţate că nu trebuie să șteargă această cale;
- Rezervat – dacă e 0, ignorat la primire;
- Contor Destinaţii – numărul destinaţiilor conţinutului mesajului, minim 1;
- Destinaţie IP ce este indisponibilă – adresa IP ce a devenit indisponibilă din cauza pierderii unei legături;
- Număr secvenţă al destinaţiei ce este indisponibile – numărul de secvenţă specificată în câmpul anterior din tabela de rutare. Mesajul de eroare RERR este trimis atunci când întreruperea unei căi provoacă indisponibilitatea uneia sau mai multor destinaţii.

3.2.2.2.5. Îmbunătățiri

După cum am precizat și în cazul protocolului DSDV, rutarea asimetrică este una din probleme. Sunt frecvente cazurile în care presupunerea că legăturile sunt simetrice nu este adevărată. Rețelele ce folosesc protocolul AODV au un câștig prin folosirea extensiei pachetului Hello ce are lista vecinilor cunoscuți ai respectivului nod. Atunci când este recepționat mesajul Hello fără adresa IP a nodului ce primește acest mesaj, acesta consideră că legătura este unidirecțională și nu trebuie să trimită nimic nodului respectiv.

Întrucât protocolul ce stabilește legătura dintre adresele MAC și adresele IP, ARP (Address Resolution Protocol), face presupunerea existenței unei legături bidirecționale pentru rutarea asimetrică

trebuie utilizați algoritmi speciali pentru rezoluția adreselor. Protocolul AODV identifică rutele asimetrice ceea ce este un lucru foarte important.

Este posibilă utilizarea numai a legăturilor simetrice și dezactivarea celor asimetrice, însă pentru acest lucru este nevoie de o metoda de anunțare. După cum am precizat fiecare nod ce recepționează un pachet RREP va răspunde printr-un pachet RREP-ACK. Atunci când este recepționat un mesaj RREP nodul va ști că legătura este simetrică întrucât acest pachet este răspunsul cererii de rută. Mesajul RREP-ACK asigură simetria legăturii către următorul nod.

Securitatea, ce este oferită și de protocolul AODV, este o facilitate importantă a oricărui protocol. Nu orice persoană are acces la mediu de comunicație trebuie să aibă acces la rețeaua construită. Acest protocol prezintă puține puncte slabe din punctul de vedere al rutării. Metodele ce au fost folosite în cazul altor protocoale sunt suficiente și în cazul protocolului AODV. Acestea constau în metode de păstrare a confidențialității și prin autentificare.

3.2.2.3. Protocolul DSR

Protocolul DSR (Dynamic Source Route) sau în traducere rutare dinamică a sursei este un protocol simplu și eficient utilizat în rețelele wireless multi-salt și ii aparține lui *David B. Johnson.* [27]

Acest protocol a fost gândit pentru a fi utilizat în rețelele de senzori ce prezintă mobilitate ridicată cu restricția numărului maxim de noduri la 200. Protocolul DSR este un protocol reactiv, adică actualizează informațiile de rutare la cerere.

Obiectivele acestui protocol au fost adaptarea rapidă la modificările rețelei, evitarea supraîncărcării și oferirea garanției livrării pachetelor cu succes în ciuda dificultăților apărute în rețea datorate de mobilitățile nodurilor sau a modificărilor survenite. [27]

Rutele sunt descoperite în mod dinamic prin intermediul mai multor salturi spre orice posibilă destinație. Toate pachetele de date trimise conțin, în antet, o lista completă având nodurile intermediare dintre sursă și destinație. Astfel sunt evitate buclele infinite și necesitatea ca nodurilor de legătură să posede informații referitoare la rutare. Lista nodurilor ce trebuie traversate, adică ruta ce este salvată în toate pachetele de date ajută nodurile să își creeze propriile căi. [37]

Explorarea și mentenanța rutelor necesită utilizarea a două mecanisme ce cooperează și asigură implicit buna funcționare a protocolului DSR.

Procesul prin care un nod găsește calea dintre sursă și destinație prin care transferă pachete de date reprezintă metoda de descoperire a rutelor. Descoperirea rutelor se efectuează doar în cazul în care nodul sursă nu are la dispoziție o rută către destinație.

Mecanismul de întreținere a rutelor este procesul prin care sunt sesizate modificările topologice din rețea de către nodul sursă în timpul utilizării căii respective. Atunci când se produce o rupere a

legăturilor dacă nodul cunoaște o alta rută către destinație o va utiliza pe aceea, iar dacă nu, o nouă procedură de explorare a rutelor va fi apelată.

Cele două mecanisme, explorarea rutelor și mentenanța acestora, lucrează numai la nevoie. Protocolul DSR, spre deosebire de alte protocoale nu are nevoie de o actualizare periodică. Acest lucru face ca protocolul DSR să se comporte foarte bine din punctul de vedere al solicitării rețelei. Mesajele de control suplimentare pot fi chiar inexistente dacă în rețea nu s-au produs modificări. La apariția schimbărilor topologice sigurul lucru ce îngreunează funcționarea este descoperirea noilor rute.

Un nod intermediar ce răspunde unui apel al procedurii de explorare a rutelor poate reține în memorie multiple rute pe care să le folosească în caz de nevoie cu întârzieri mai mici.

Prin cele două proceduri, protocolul DSR face față cu succes legăturilor asimetrice și unidirecționale. Un alt punct în plus al acestui algoritm este acela că se poate conecta cu multiple rețele wireless de diferite tipuri.

3.2.2.3.1. Căutarea rutelor

Atunci când se dorește comunicarea între două noduri, nodul sursă atașează în antetul mesajului o cale prin care sunt anunțate o serie de salturi ce trebuie parcurse de către destinație. La început nodul sursă verifică în memorie dacă există vreo cale determinată anterior pentru a trimite pachetele la destinație. În cazul în care nici o cale spre destinația dorită nu este găsită atunci va fi apelat procesul de căutare a rutelor. [37]

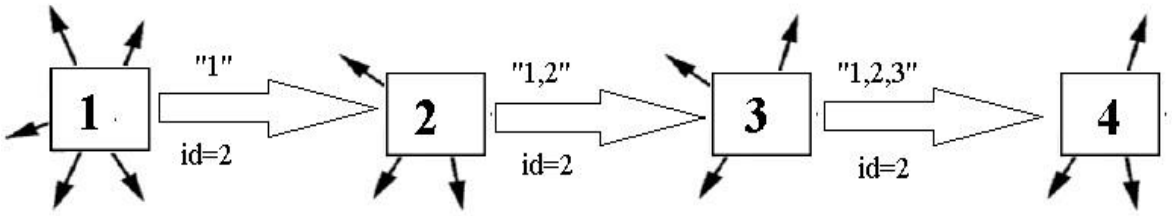


Figura 3.9 Descoperirea rutei în care nodul 1 este sursa, iar nodul 4 este destinația

Modul de căutare și descoperire a rutei este ilustrat în figura 3.9. Nodul 1, sursa, dorește să comunice cu nodul 4, destinația. Pentru acest lucru va fi generat un pachet RREQ, adică o cerere de rută ce va fi recepționat de toate nodurile ce se găsesc în aria de acoperire a nodului sursă. Orice pachet RREQ deține informații prin care este indicată sursa mesajului, destinația acestuia, dar și un unic identificator al pachetului. Mesajele RREQ dețin, în plus, o lista ce conține nodurile intermediare prin care a fost transmisă o copie a mesajului RREQ, iar inițializarea acestei liste este făcută de nodul sursă.

Atunci când mesajul RREQ este recepționat de către un nod căruia i se adresează, nodul în cauză trimite răspuns un pachet RREP nodului sursă al mesajului prin care se cere ruta. Acestui răspuns i se atașează o listă a nodurilor intermediare ce au fost parcurse de nodul RREQ. Acest pachet trimis ca răspuns, RREP, oferă rută cerută nodului sursă ce o va retine în memoria sa.

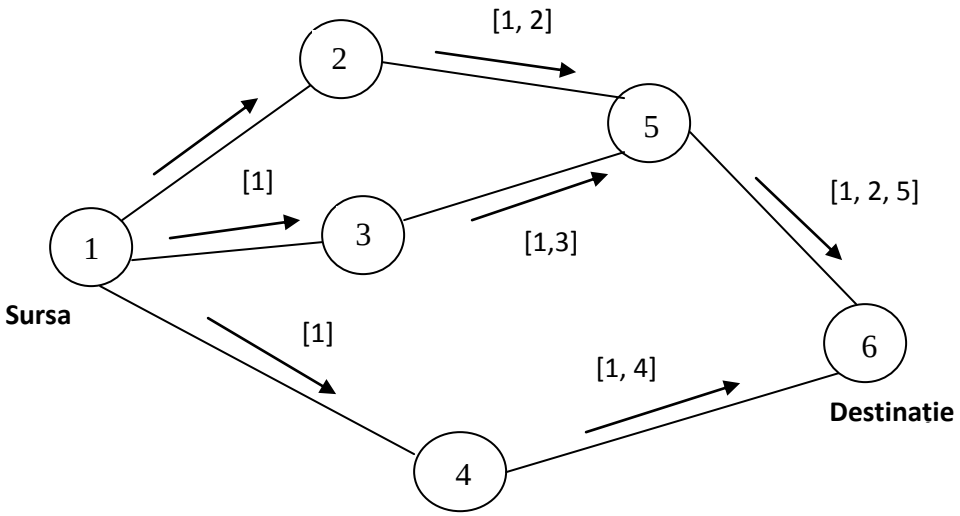


Figura 3.10 Propagare pachete RREQ

Dacă mesajul RREQ este recepționat de un nod căruia acest pachet nu i se adresează, acest nod va adăuga în lista mesajului RREQ adresa sa și va trimite mesajul mai departe. În acest mod protocolul evită formarea buclelor pentru că dacă mesajul RREQ este recepționat de un nod ce a mai primit de curând de la aceeași sursă cererea sau dacă adresa sa este deja în lista mesajului, nodul nu va lua în seamă pachetul. [37]

Nodul ce a primit o cerere de rută ce i se adresează trebuie să răspundă sursei mesajului. Pentru acest lucru este căutată în memorie nodului curent o ruta prin care să fie transmis răspunsul. Dacă este descoperită o cale în memorie atunci aceasta va fi folosită pentru trimiterea mesajului RREP. În caz contrar, pentru a evita întârzierile, mesajul RREP este trimis pe aceeași cale prin care a fost recepționat.[27]

Toate nodurile generatoare de cereri de căutare a rutelor memorează, într-un buffer, o copie a acestora. Mesajele de cerere pentru care nu s-a recepționat nici un pachet RREP se vor păstra pentru un timp în buffer. Această coada, pentru a nu se satura, respectă principiul FIFO, prin care primul venit este primul ce părăsește coada. După un timp în care nu s-a recepționat nici un răspuns, sursa va genera încă o data mesajul RREQ. Timpul de așteptare între două retransmiteri nu trebuie să fie prea scurt întrucât în acest mod rețeaua este supraîncărcată. Această așteptare ce intervine reprezintă o constrângere a vitezei de descoperire a rutelor. Ideea aceasta este asemănătoare cu modul în care este limitată rata cererilor ARP ce sunt trimise la aceeași adresa IP prin care sunt cerute noduri din Internet.

3.2.2.3.2. Păstrarea rutelor

In momentul în care un nod generează un mesaj recepționarea acestuia de către alt nod trebuie confirmată. Astfel fie că este vorba despre nodul destinație sau despre un nod intermediar, fiecare dintre acestea va confirma recepționarea pachetului și faptul că acesta a fost trimis mai departe, în cazul în care nodul nu este destinația. Pachetul parcurge rețeaua hop cu hop până la destinație. Spre exemplu în figura 3.11 nodul 1 este răspunzător pentru primirea mesajului de către nodul 2, nodul 2 este răspunzător pentru primirea mesajului de către nodul 3, nodul 3 este răspunzător pentru primirea

mesajului de către nodul 4, nodul 4 este răspunzător pentru primirea mesajului de către nodul 5. Aceste confirmări nu suprasolicită rețeaua. Confirmările pot fi făcute prin informări pasive (nodul 2 confirmă primirea pachetului de către nodul 3 atunci când sesizează trimiterea de către nodul 3 a pachetului spre nodul 4) sau printr-un flag la nivel legăturii de date. În cazul în care nici una din aceste variante se poate aplica, nodul sursă poate seta un bit în headerul mesajului pentru că fiecare nod receptor să trimită o confirmare. [37]

In cazul legăturilor unidirecționale confirmările sunt trimise prin intermediul altei căi.

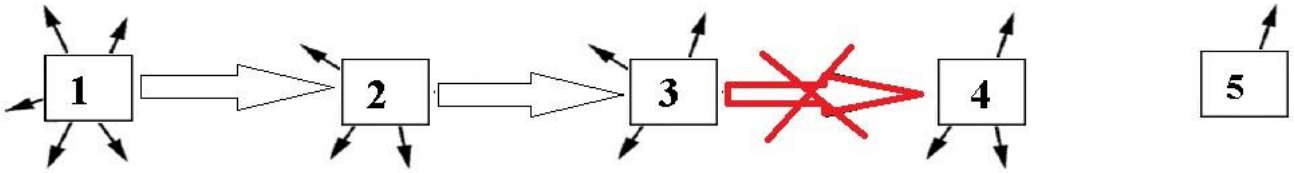


Figura 3.11 Nodul 4 și 5 nu pot recepționa pachete întrucât nodul 3 nu poate trimite mai departe

Dacă nici o confirmare nu a fost recepționată deși mesajul a fost retrimis de un număr maxim de ori, ultimul nod ce a recepționat mesajul va trimite un mesaj eroare de rută către nodul sursă, identificând astfel unde este întreruptă conexiunea. În exemplul nostru între nodul 3 și 4 conexiunea este ruptă. Ca efect al acestui lucru, pentru ca mesajul să ajungă la destinație, pachetul va trebui trimis direct nodului 4 evitând astfel nodul 3 în care pachetul se bloca. Nodul 3 va trimite o eroare nodului sursă și în acest mod se va cunoaște locul unde există o problemă. Apoi nodul sursă va șterge nodul 3 din ruta sa și va încerca prin pachete de tip RREQ să descopere ruta către nodul 5 prin intermediul altui nod, în cazul nostru nodul 4, evitând nodul 3.

3.2.2.3.3. Micșorare rute

Căile folosite pentru trimiterea pachetelor dintre nodul sursă și nodul destinație pot fi automat micșorate dacă între cele două există noduri redundante. Confirmările pasive folosesc o metodă asemănătoare cu cea folosită la micșorarea rutelor. Dacă un nod, în starea de primire aleatoare a pachetelor, poate recepționa un mesaj ce poartă o ruta, atunci acesta va examina sectorul căii nefolositor. Dacă nodul în cauză nu reprezintă următorul hop al mesajului pe care l-a recepționat, dar apare menționat în continuare în ruta mesajului atunci decide că segmentul dintre nodul ce a trimis ultimul pachetul și el este nefolositor.

In figura 3.12 este ilustrat un exemplu în care nodul 3 recepționează mesajul RREQ ce a fost trimis de nodul 1 către nodul 2 întrucât nodul 3 se află în raza de acoperire a nodului 1. Mesajul urmează să fie trimis nodului 3 conform rutei, iar acesta trimite un răspuns nodului sursă. Prin răspunsul oferit de nodul 3 nodul sursă dispune acum de calea cea mai scurtă.

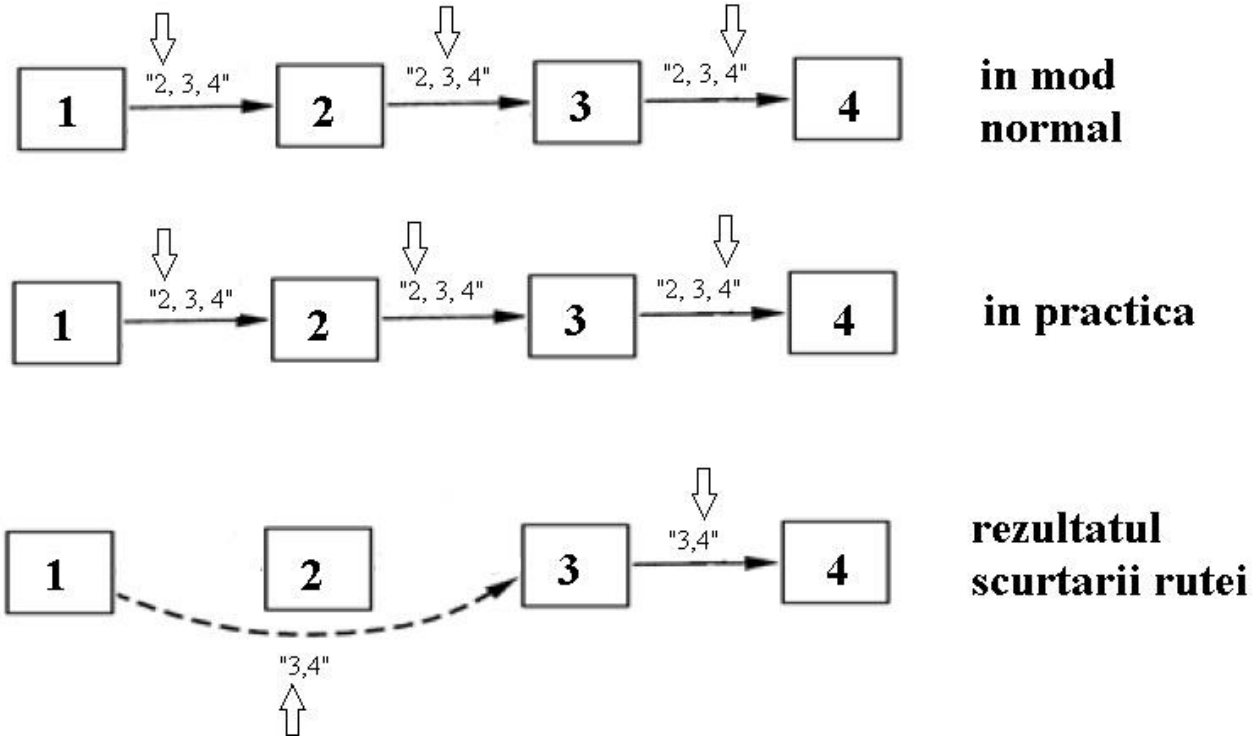


Figura 3.12 Exemplu de micșorare a rutei

3.2.2.3.4. Formatul pachetelor protocolului DSR

3.2.2.3.4.1. Antetul pachetelor cerere de ruta RREQ

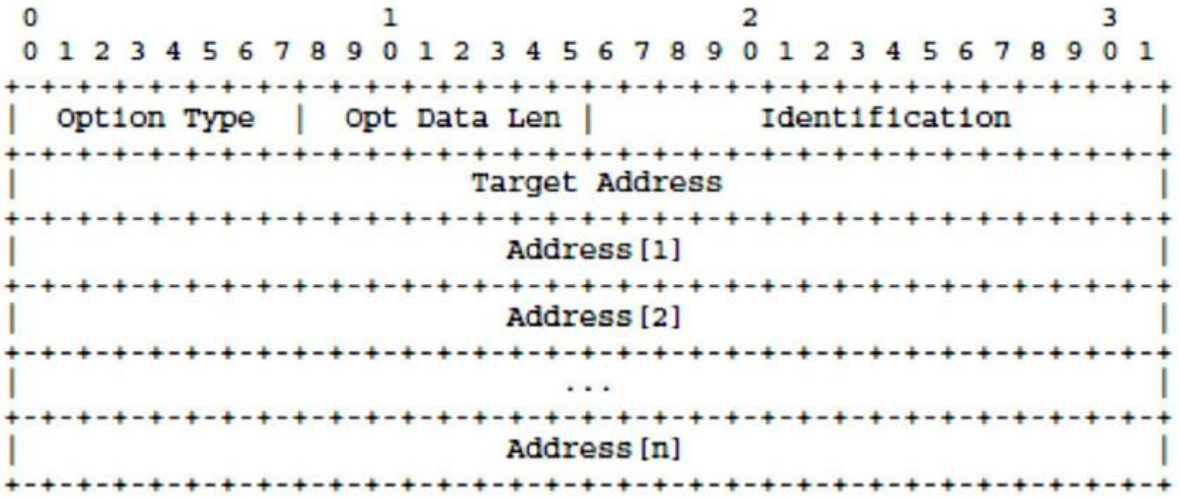


Figura 3.13 Formatul cererilor de ruta [37]

- Tipul de opţiune: Pachete Route Request au valoarea câmpului 1;
- Opt Data Len: mărimea opţiunii RREQ. Trebuie să aibă valoarea (4*n)+6, unde n este numărul adreselor din mesajul Route Request;
- Identificator: valoare unică generată de sursa cererii de cale. Fiecare mesaj Route Request are un unic identificator;
- Adresa Ţintă: adresa nodului destinaţie;
- Adresa [i]: adresa IPv4 a nodului i înscris în Route Request. [28]

3.2.2.3.4.2.Formatul pachetelor de raspuns RREP

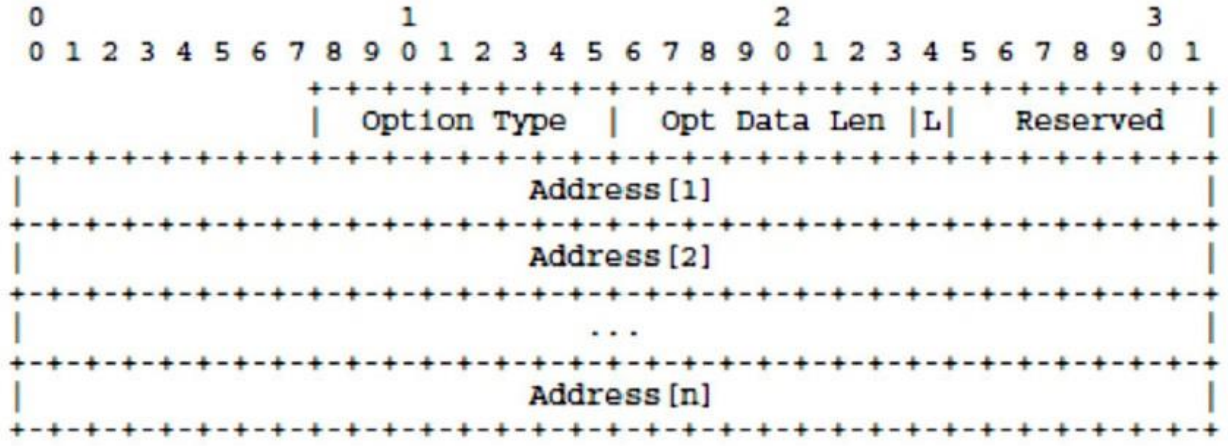


Figura 3.14 Formatul pachetelor RREP [37]

- Tipul opţiunii: Mesajele Route Reply au valoarea 2;
- Opt Data Len: mărimea opţiunii RREP. Trebuie să aibă valoarea (4*n)+6, unde n reprezintă numărul adreselor din mesajul Route Reply;
- L (Last Hop External): fanion care arată că ultima adresă din mesajul Route Reply este corespunzătoare unui nod ce leagă rețeaua cu una externă;
- Rezervat: este 0 şi ignorat la primire;
- Adresa [1,...,n]: adresele ce compun calea inversă spre sursa care a declanșat cererea de rută

3.2.2.3.4.3. Formatul pachetului eroare RERR

- Tipul opţiunii: Mesajele Route Error au valoarea 3;
- Opt Data Len: mărimea în octeţi a opţiunii RERR. Câmpul trebuie setat la valoarea 10 + dimensiunea specifică tipului de eroare;
- Tipul erorii: tip eroare;
- Rezervat: este 0 şi ignorat la primire;
- Adresa sursei de eroare: adresa nodului sursă ce a declanșat mesajul RERR;
- Adresa destinaţiei de eroare: adresa nodului destinație căruia îi este destinat mesajul RERR;
- Informaţia tipului de eroare: informații despre tipul de eroare.

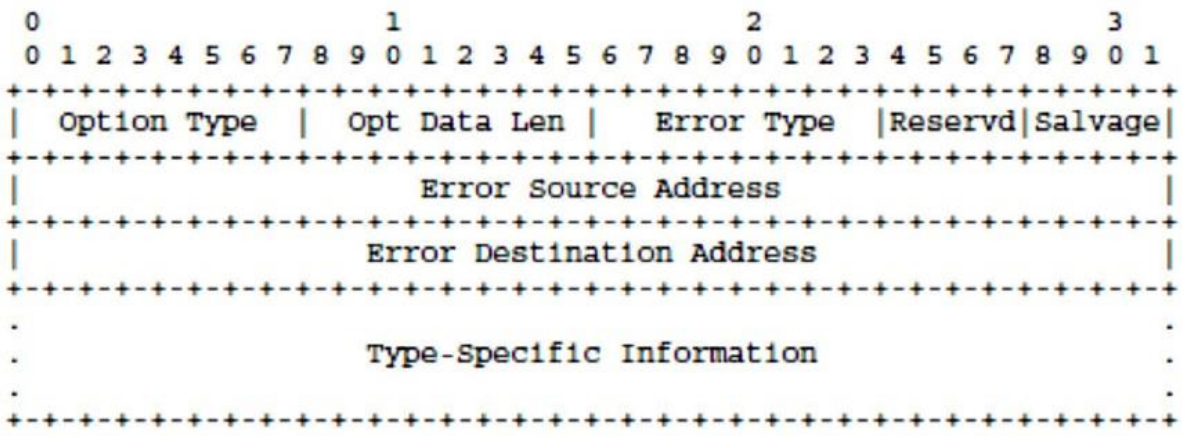


Figura 3.15 Formatul pachetelor eroare [37]

3.2.2.3.5. Problemele protocolului

Protocolul DSR deși este un protocol des utilizat este performantele sale sunt afectate de unele probleme ce îl expun și atacurilor. [29]

Nu este implementată o metodă prin care rutele să fie eliminate atunci când sunt prea vechi. Astfel memoria nodurilor va păstra rute ce sunt întrerupte, rute ineficiente ceea ce în rețelele de senzori nu poate fi acceptat. Utilizarea acestor rute determina irosirea de pachete și folosirea inadecvată a benzii rețelei. Performanța rețelei nu este mult îmbunătățită de faptul că nodurile intermediare dețin căi proprii în funcție de ce mesaje primesc de la nodurile vecine. Pentru acest lucru se pot utiliza rute multiple. Memorarea unor rute vechi face ca nodurile să trimită răspunsuri eronate cu rute ce nu pot fi

utilizate, iar acest lucru face ca ruta corectă, optimă, să fie găsită mai greu datorită intoxicării nodului ce are nevoie de o cale până la destinație. [29]

Răspunsurile trimise de nodurile situate între sursă și destinație sunt folosite într-un proces de învățare rapidă a căilor. Acest lucru este posibil întrucât nu e nevoie ca mesajele de cerere să ajungă până la destinație. Totuși, în lipsa unui sistem de eliminare a rutelor vechi este foarte posibilă intoxicarea cu date false a rețelei.

Atunci când o legătură este întreruptă și un nod depistează o altă cale pentru a ajunge la destinație prin utilizarea unei rute ce era salvată în memoria sa va încerca să trimită datele prin ruta găsită. Această abordare nu este valabilă în cazul unei mobilități crescute a rețelei întrucât noua rută ar putea fi defapt foarte veche și să nu mai existe. Acest lucru pe lângă faptul că nu ar ajuta lucrurile, ar îngreuna și mai mult situația.

Aceeași problemă, a mobilității crescute, este întâlnită și în cazul micșorării rutelor într-un mod automat. Dacă un nod interceptează un pachet ce nu ii este adresat, dar el se află pe lista nodurilor următoare din ruta pe care trebuie să o urmeze, acel pachet va încerca să scurteze calea prin trimiterea unui mesaj răspuns către sursa. [29]

In cazul unei rețele statice acest lucru funcționează perfect, dar dacă vorbim de rețele de senzori ce au o mobilitate crescută avantajele se diminuează, ba chiar apar dificultăți în ceea ce privește rutarea.

3.2.2.3.6. Imagine de ansamblu

Indicatori	DSDV	AODV	DSR
Complexitate timp identificare ruta	O(d)	O(2d)	O(2d)
Complexitate timp reidentificare ruta	O(d)	O(2d)	O(2d) sau 0 dacă este memorată o altă rută în cache
Complexitatea de comunicație la inițializare	O(N)	O(2N)	O(2N)
Complexitate de comunicație atunci când se pierde ruta	O(N)	O(2N)	O(2N)
d- diametrul rețelei N- numărul de noduri din rețea			

Tabel 3.1 Complexitatea protocoalelor de rutare

	DSDV	AODV	DSR
Tip protocol	Proactiv	Reactiv	Reactiv
Informații despre rețea	Imagine de ansamblu a rețelei	Noduri vecine	Ruta de la sursă la destinație
Mesaje de control	Mesaje de tip „Hello” și mesaje de update	Mesaje de tip „Hello” pentru localizare vecini, mesaje de căutare ruta: RREQ, RREP, RERR	Mesaje de căutare ruta: RREQ, RREP, RERR
Modalitate identificare ruta	Broadcast	Inundare	Inundare
Păstrare rute	Tabela de rutare	Tabela de rutare	Cache cu rute
Actualizare informație rutare	Periodic și când s-au produs modificări	La nevoie	La nevoie
Rutare fără bucle infinite	DA	DA	DA
Posibilitate de a găsi căi multiple	NU	DA	DA
Criteriu de identificare cea mai bună ruta	Cea mai scurtă și cea mai nouă	Cea mai nouă și cea mai scurtă	Cea mai scurtă
Posibilitate de multicast	NU	DA	NU
Avantaje	Rutare eficientă în rețele de dimensiuni mici, fără bucle infinite	Rutare eficiență în rețele cu dinamicitate crescută, rezolvând și problema căilor unidirecționale	Rutare foarte eficientă pentru rețele dinamice de dimensiuni mici, poate avea căi multiple, modalitate de scurtare rute, rezolvă problema căilor unidirecționale
Dezavantaje	Nu rezolva problema căilor unidirecționale, are eficiență scăzută în rețele de dimensiuni mari și nu asigură căi multiple	Pot apărea întârzieri și congestii datorită proceselor de identificare a rutei.	Este necesar un mecanism de actualizare a cache-ului. Pot apărea întârzieri și congestii datorită proceselor de identificare a rutei.

Tabel 3.2 Imagine de ansamblu asupra protocoalelor

4. Optimizarea rutării pe baza căilor multiple

Performanțele rețelelor de senzori ce folosesc ca protocoale de rutare cele prezentate în capitolul anterior sunt limitate din cauza diverselor impedimente.

Pe de o parte problemele principale ale protocoalelor de rutare pro-active sunt determinate chiar de principiul pe care se bazează aceste protocoale adică ideea de a păstra tabele de rutare pentru orice posibilă destinație. Acest lucru se reflectă în faptul că rețeaua este supraîncărcată cu mesaje de control ceea ce poate duce la congestionarea acesteia și că timpul necesar actualizării este foarte mare. O altă

problemă este reprezentată de prezența legăturilor unidirecționale. Aceste probleme constituie o risipă importantă și inacceptabilă a resurselor energetice și a capacității de calcul.

Pe de altă parte utilizarea protocoalelor de rutare reactive duce la creșterea performanțelor față de rețelele ce utilizează protocoale pro-active, însă și acestea prezintă unele probleme. Aceste protocoale descoperă rutele numai atunci când îi sunt necesare pentru a transmite date spre o destinație pentru care nu se cunosc informații de rutare. Procesul de explorare a rutelor se desfășoară prin inundarea rețelei cu mesaje prin care este cerută o rută. Efecte negative sunt vizibile și în cazul declanșării procedurilor în momentul pierderii unei rute, a unor modificări topologice, etc. Astfel frecvența proceselor de explorare a rutelor trebuie redusă cât mai mult, pentru a împiedica apariția întârzierilor sau a supraîncărcărilor din rețea.

Astfel soluția de optimizare ce a îmbunătățit performanța algoritmilor reactivi pentru reducerea frecvenței operațiilor de descoperire a rutelor a fost implementarea căutărilor de multiple căi. Această tehnică, după cum îi spune și numele, se bazează pe o idee simplă aceea că într-un proces de explorare a rutelor nu va fi memorată în tabela de rutare numai o cale ci vor fi memorate mai multe. Criteriile de alegere a rutelor este numărul de salturi, lungimea de bandă, costul rutelor. [30]

In capitolul precedent a fost prezentat protocolul, DSR, ce avea această funcție, de a memora mai multe căi pentru o destinație, însă acesta întâmpină dificultăți datorită rutelor prea vechi ce erau utilizate și care intoxicau sursa. Astfel mecanismul de rutare cu căi multiple, în loc să eficientizeze rețeaua, îi îngreuna existența.

Să presupunem că există un transfer de mesaje intre două noduri ale unei rețele de senzori ce se efectuează prin mai multe salturi. În acest timp din diverse motive una din legăturile ce fac posibilă conexiunea se oprește. Acest lucru poate fi datorat mobilității nodurilor sau pur și simplu un nod nu mai are suficientă energie. Transferul de date nu s-a terminat, așa încât nodul sursă are nevoie de o altă rută. Folosind un protocol de rutare reactiv, în acest moment o nouă procedură de explorare a rutelor va fi apelată. Prin acest protocol ce folosește căi multiple, nodul sursă are salvate mai multe rute și transmisia poate continua imediat pe una din căile memorate. De abia atunci când nici una din aceste rute nu mai este valabilă, se va apela procedura de explorare a rutelor. Această metodă are rezultate foarte bune, ne mai fiind înregistrate întârzieri mari în rețea atunci când se pierde o rută și nici nu va exista o suprasolicitare a rețelei prin mesaje de control. Reluarea rapidă a legăturii cu nodul destinație va eficientiza și folosirea protocolului TCP care nu va mai întâmpina scăderea ratei de transfer și întârzierile ce au fost prezentate în capitolul al doilea. De asemenea trebuie precizat încă un avantaj important al folosirii acestei tehnici în rețelele de senzori, acela că eficientizează consumul energetic, timpul de viață al rețelei crescând, prin micșorarea numărului de mesaje de control. [30]

4.1. Modelare matematică [31]

„Presupunând că există un nod sursă S ce dorește să transmită către un nod destinație D. Calea este compusă din k legături wireless ce traversează k-1 noduri. Legătura intermediară, L_i, având indicele i în cadrul rutei are durata de viață X_{Li}. Facem presupunerea că timpii de viață X_{Li} având i=1,2,...,k sunt independenți și sunt variabile aleatoare având o distribuție exponențială și o medie *l*.

Durata de viață a unei rute P, ce are în componența sa k legături, este o variabilă aleatoare X_P, întrucât o rută devine neutilizabilă atunci când oricare legătură din cele ce o compun se rupe. Aceasta poate fi exprimată astfel:

$$X_P=\min(X_{L1}, X_{L2}, ..., X_{Lk}) \quad (1)$$

Timpii dintre două descoperiri de rută succesive îi vom determină statistic, utilizând presupunerile făcute mai sus.

Presupunem că o sursă S are n rute către o destinație. Ruta primară este notată cu P_1, rutele secundare se vor nota cu P_2, P_3,....., P_N. Lungimea rutei P_i, este k_i. Timpul după care nici una din rute nu mai este utilă este o variabilă aleatoare T.

$$T=\max(X_{P1}, X_{P2}, ..., X_{PN}) \quad (2)$$

Astfel T este timpul dintre două descoperiri de rută succesive. Presupunând, în continuare că latența transmisiei capăt la capăt a pachetelor este mult mai mică în comparație cu intervalul dintre schimbările de ruta, timpii de propagare a mesajelor de cerere de rută sau a mesajelor de eroare pot fi neglijați în comparație cu T. Presupunerile sunt rezonabile, astfel rutele vor cădea în timpul procesului de descoperire ori de recuperare a acestora. Nici un protocol de rutare nu va avea performanțe în aceste condiții.

Formula funcției de densitate de probabilitate a variabilei aleatoare T este:

$$f_T(t)= \lambda_1 e^{-\lambda_1 t}(1- e^{-\lambda_2 t})(1- e^{-\lambda_3 t})....(1- e^{-\lambda_N t}) + \lambda_2 e^{-\lambda_2 t}\left(1- e^{-\lambda_2 t}\right)(1- e^{-\lambda_3 t})....(1- e^{-\lambda_N t})+....+ \lambda_N e^{-\lambda_2 t}\left(1- e^{-\lambda_2 t}\right)(1- e^{-\lambda_3 t})....(1- e^{-\lambda_N t}) \quad (3)$$

unde $\lambda_i= \frac{k_i}{l}$; $l= \frac{1}{X_{P_i}}$; $X_{P_i} = timpul\ de\ viata\ a\ rutei\ i.$

Demonstrație: Considerând N variabile aleatoare distribuite exponențial X_{P_1}, X_{P_2}, ..., X_{P_N}. Funcția densitate de probabilitate a unei astfel de variabile aleatoare este $f_{X_{P_i}}(t)= \lambda_i e^{-\lambda_i t}$, i=1,2,...,N. Cum variabilele aleatoare sunt independente, funcția de distribuție cumulativă a variabilei T este obținută prin: $F_T(t)=P[T\leq t]$

$$F_T(t)=P[\max(X_{P_1}, X_{P_2}, ..., X_{P_N}\leq t)]$$

$$F_T(t)=P[(X_{P_1} \leq t)\cap(X_{P_2} \leq t) \cap.....\cap(X_{P_N} \leq t)]$$

$$F_T(t)=\prod_{i=1}^{N} F_{X_{P_1}}(t) \quad (4)$$

unde $F_{X_{P_1}}(t)= (1- e^{-\lambda_i t})$.

Derivând formula (4) în raport cu t, se obține relația din formula (3).

Valoarea preconizată a timpului T poate fi dedusă din formula 3, cunoscând lungimile rutelor ca număr de hop-uri k_i=1,2,.....,N. De exemplu pentru N=2, valoarea preconizată a lui T, E[T] este:

$$E[T]=\frac{\lambda_1+\lambda_2+\lambda_1\lambda_2}{\lambda_1\lambda_2(\lambda_1+\lambda_2)} \quad (5)$$

In cazul în care ruta primară este compusă dintr-un număr de 3 noduri, iar ruta secundară dintr-un număr de 4 (k_1=3, k_2=4). $E[T]=\frac{9+16+12}{12*7}$ $l=\frac{37}{84}l$. Prin urmare timpul de asteptare dintre cele două descoperiri de rută succesive este $\frac{37}{84}l$. În cazul unui protocol reactiv ce nu folosește principiul căilor multiple E[T]=1/3. Astfel rutele multiple reduc cu aproximativ 25% frecvența proceselor de descoperire a rutelor față de rutele singulare.”

4.2. AOMDV

Protocolul AOMDV este varianta optimizată a protocolului AODV. Este ales pentru a fi optimizat acest protocol în detrimentul protocolului DSR întrucât în urma studiilor efectuate s-a constatat că protocolul AODV are performanțe mai bune și un potențial mai mare de dezvoltare.

AOMDV își propune recuperarea rapidă a conexiunii dintre sursă și destinație și toleranță mare la erori. Astfel acest protocol va căuta mai multe rute diferite fără bucle infinite, atunci când o operație de căutare de rută este lansată.

Caracteristica de bază a acestui protocol este că reușește să găsească mai multe rute valide între sursă și destinație la un singur apel al funcției de explorare a rutelor. Atunci când o rută este întreruptă, sau din alte motive devine neutilizabilă, AOMDV va comuta pe ruta de rezervă, considerată având o performanță inferioară. Astfel nu va mai fi necesar apelul procedurii de explorare a rutelor și trimiterea de mesaje de control. Această procedura va fi efectuată numai atunci când toate căile salvate devin neutilizabile. Din motive de eficiență numai rutele diferite vor fi memorate, astfel atunci când una devine invalidă acest lucru să nu afecteze și alte rute. Deci, modul de operare al acestui protocol nu este de a trimite în același timp pachete prin ambele căi ci de a folosi calea de rezervă când ruta principală devine invalidă și de a evita astfel suprasolicitarea și timpii de așteptare.

4.2.1. Modul de explorare a căilor

Deși pare asemănătoare modalitatea de explorare a rutelor este diferită față de cea a protocolului de la care s-a plecat, AODV. Oricare nod ce se află între sursă și destinație poate fi componentă a unei rute între cele două și având un număr mare de noduri rezultă că vor exista o sumedenie de căi.

Dacă vorbim despre protocolul AODV, răspunsul unui mesaj de cerere de rută parcurge același drum, dar în sens invers. În cazul protocolului AODV, răspunsurile multiple ale unei cereri de cale nu erau luate în seamă.

In cazul protocolului AOMDV toate mesajele de răspuns ale unei cereri de cale sunt testate, pentru a memora mai multe căi ce leagă sursa de nodul destinație. Dintre toate mesajele ce ajung ca

răspuns la o cerere de rută nu toate fac referire la căi diferite. Pentru a ilustra această situație figura de mai jos arată o posibilă situație. În această figură deși sunt recepționate trei mesaje de răspuns la cererea de rută numai două sunt considerate a fi căi total diferite, celelalte două rezultând dintr-un singur vecin al nodului sursă. Pentru a face diferența între aceste rute se adăugă în cadrul mesajului RREQ un câmp suplimentar unde vor fi înregistrare informații cu privire la vecinii nodului sursă și ai nodului destinație. [32]

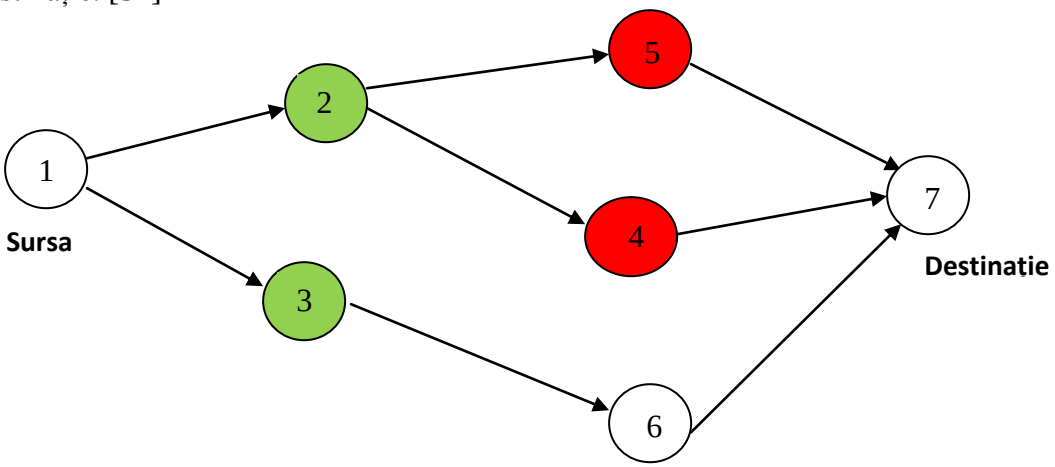


Figura 4.1 Rute multiple nedisjuncte

Primul salt al unei cereri de rută RREQ este reținut de fiecare nod într-o listă cu nodurile vecine. Calea în sens invers este acceptată când primul nod este unic. Căile din figura 4.2 sunt diferite întrucât au ca punct de start diferiți vecini ai nodului sursă.

Nodul destinația recepționează mesajele RREQ și va trimite un răspuns nodul sursei prin pachete RREP utilizând multiple căi inverse. Astfel nodul destinație va trimite mesaje RREP pe fiecare cale considerată a fi diferită. Numărul de multiplicări ale mesajului RREP sunt reduse la un număr x. Dacă există un număr mai mare de x rute distincte, nodul destinație nu le va accepta pe cele cu metrica mai dezavantajoasă și le va ignora. În acest fel se evită detonarea unui număr mare de mesaje RREP, dacă traseele căilor de întoarcere se întâlnesc într-un nod intermediar. [32] Acest caz este prezentat în figura 4.2.

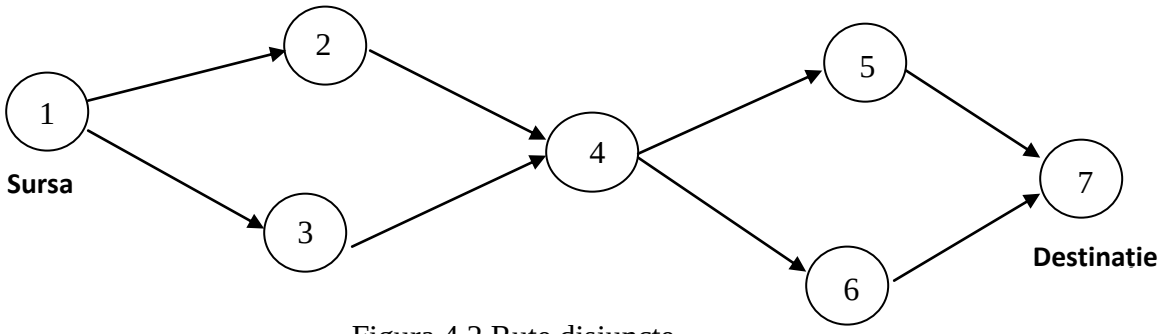


Figura 4.2 Rute disjuncte

4.2.2. Evitarea buclelor infinite

Metoda ce folosește numerele de secvență pentru evitarea buclelor infinite, ce este folosită în cazul protocolului AODV, este folosită și în cazul protocolului AOMDV. În cazul rutării cu căi multiple implementarea acestei tehnici este ceva mai dificilă și este nevoie de mai multă atenție. Spre deosebire de utilizarea unei singure rute, aici căile vor avea metrici distincte până la nodul destinație. Metrica reprezintă, după cum am explicat în capitolul 3, numărul de salturi pe care o rută le cuprinde între nodul sursă și nodul destinație. Trebuie să existe o mai mare atenție a nodurilor în a se informa care este ruta principală. Eventualele erori ce pot apărea datorită acestei probleme sunt eliminate prin folosirea unor reguli clare prin care se fac informările. Controlul acesteia este făcut de un câmp din tabela de rutare ce poartă numele de „announced number of jumps" – numărul de salturi anunțat. Câmpul se inițializează la fiecare actualizare a numărului de secvență a căii. [32]

4.2.3. Formatul tabelei de rutare

Protocolul AOMDV folosește o tabelă de rutare asemănătoare cu tabela de rutare a protocolului AODV. Diferențele dintre acestea sunt reprezentate de două câmpuri. Metrica înlocuită de metrica anunțată, iar saltul următor este înlocuit de lista căilor. Formatul tabelei de rutare este prezentat în tabelul 4.1.

Destinație
Număr de secvență
Metrica anunțată
Timp de viață
Lista căilor ruta1={salt următor, metrica;.....salt următor, metrica} ruta2={salt următor, metrica;.....salt următor, metrica}

Tabel 4.1 Formatul tabelei de rutare

Metrica anunțată înseamnă numărul de salturi maxim ce o rută din multiplele căi le poate avea.
Actualizarea rutelor se face pe baza algoritmului din figura 4.3.

Acest algoritm prezintă regulile ce trebuiesc respectate atunci când un nod primește o informație nouă referitoare la o anumită rută ce leagă un nod j de un nod destinație d. Numărul de secvență al nodului i către d este reprezentat de variabila $seqnum^d_i$, $advertised\_hopcount^d_i$ reprezintă numărul anunțat de salturi pe care o are rută dintre i și d, iar $route\_list^d_i$ este lista de căi de rezervă către d. Atunci când un nod primește un răspuns la rută, cerere de rută având forma de mai jos, va modifica dacă e cazul numărul anunțat de salturi. [32]

$$advertised\_hopcount^d_i := \max_k\{hopcount_k \mid (nexthop_k, hopcount_k) \in route\_list^d_i\}$$

In cazul în care nici o rută din cele ce există în tabelă nu sunt folosite în timpul specificat de câmpul timp de viață, atunci toate aceste căi vor fi considerate nefolositoare și vor fi șterse.

Oricare două noduri succesive i, j ce sunt cuprinse de o rută trebuie să respecte condiția:

$$(seqnum^d_i, advertised\_hopcount^d_i, i) > (seqnum^d_j, advertised\_hopcount^d_j, j)$$ [32]

1: **if** $(seq\_num_i^d < seq\_num_j^d)$ **then** {/* enforces the sequence number rule */}
2: $seq\_num_i^d := seq\_num_j^d$;
3: $advertised\_hop\_count_i^d := \infty$;
4: $route\_list_i^d := NULL$;
5: **if** $(j = d)$ **then** {/* neighbor is the destination */}
6: insert $(j, i, 1)$ into $route\_list_i^d$;
7: **else**
8: insert $(j, last\_hop_{jk}^d, advertised\_hop\_count_j^d + 1)$ into $route\_list_i^d$;
9: **end if**
10: **else if** $((seq\_num_i^d = seq\_num_j^d)$ **and** $(advertised\_hop\_count_i^d > advertised\_hop\_count_j^d))$ **then** {/* enforces the route acceptance rule */}
11: **if** $(j = d)$ **then** {/* neighbor is the destination */}
12: **if** $((\nexists k_1 : (next\_hop_{ik_1}^d = j))$ **and** $(\nexists k_2 : (last\_hop_{ik_2}^d = i)))$ **then** {/* establishes uniqueness of next and last hops */}
13: insert $(j, i, 1)$ into $route\_list_i^d$;
14: **end if**
15: **else if** $((\nexists k_3 : (next\_hop_{ik_3}^d = j))$ **and** $(\nexists k_4 : (last\_hop_{ik_4}^d = last\_hop_{jk}^d))$ **then** {/* establishes uniqueness of next and last hops */}
16: insert $(j, last\_hop_{jk}^d, advertised\_hop\_count_j^d + 1)$ into $route\_list_i^d$;
17: **end if**
18: **end if**

Figura 4.3 Algoritm actualizare rută [32]

4.2.4. Menținerea căilor

Spre deosebire de procesul de menținere a rutelor din cadrul protocolului AODV, la acest protocol legătura dintre sursă și destinație este întreruptă atunci când toate rutele sunt compromise. Numărul de secvență al nodului destinație trebuie modificat la valoarea maximă atunci când toate rutele sunt compromise, în funcție de numărul de secvență primit în pachetele RERR. [32]

5. Implementare și evaluare experimentală

In acest capitol este descrisă desfășurarea studiului de evaluare a performanțelor a celor mai reprezentative protocoale de rutare utilizate în rețelele de senzori. Capitolul cuprinde modului de evaluare a performanțelor, scenariile pe care se bazează simulările, parametri inițiali și exemple de aplicare a rețelelor de senzori. De asemenea după obținerea rezultatelor simulărilor este prezentată analiza și interpretarea rezultatelor, dar și concluziile finale.

5.1. Simulatorul

Evaluarea performanțelor unei rețele de senzori este importantă pentru a verifica dacă așteptările teoretice se verifică și în practică. Pentru a evita irosirea inutilă de resurse acest lucru se realizează pe baza soft-urilor de simulare dezvoltate în acest scop. Cele mai utilizate simulatoare sunt NS2, NS3, OMNET, OPNET, etc.

Network simulator 2 (NS2) a fost conceput la Universitatea din Berkley și este un soft utilizat în domeniul rețelelor pentru evenimente discrete. Acesta are capabilitatea de a simula diverse topologii, scenarii ce utilizează o gamă largă de algoritmi de dirijare, tipuri de trafic și poate fi util atât în rețelele wireless cât și în cele clasice, cablate. Versiunea utilizată este cea mai recenta, 2.35.

Network simulator 2 are la bază două limbaje de programare. Apare necesitatea de a utiliza două limbaje dat fiind faptul că pentru a fi modelate protocoalele este nevoie de un limbaj întrucât se lucrează cu seturi mari de date, iar cel de al doilea este folosit pentru dezvoltarea rețelelor, crearea diferitelor scenarii și modificarea parametrilor. Simulatorul este scris în C++, iar pentru execuția instrucțiunilor utilizatorului interpretorul OTcl, extensie a limbajului Tcl orientată pe obiecte. Biblioteca acestui simulator cuprinde o gamă vastă de tipuri de rețele, protocoale și elemente de rețea. Pentru a crea o simulare utilizatorul va construi obiecte folosind OTcl și apoi acestea vor fi instanțiate. Obiectele create au un corespondent în ierarhia de clase a limbajului C++.

Pentru a vizualiza și a analiza simulările create utilizatorul are la îndemână două utilitare: NAM și XGRAPH. Primul dintre acestea creează o interfață vizuală în care se pot observa obiectele create și evoluția în timp a rețelei. Se pot vizualiza mișcările nodurilor, schimburile de pachete dar și nivelul energetic al fiecărui nod. Cel de al doilea utilitar are scopul de a reprezenta grafic rezultatele simulărilor.

5.2. Parametri de performanța

Performanța și eficiența unei rețele de senzori se pot evalua în funcție de anumite mărimi ce au titulatura de indicatori de performanță. Prin introducerea diferitelor valori ai parametrilor se pot observa modificări în comportarea rețelei. Aceste modificări se pot observa analizând spre exemplu întârzierile end-to-end sau throughput. În continuare vor fi definiți astfel de indicatori și în funcție de acești indicatori vom trage concluzia dacă un protocol de rutare este performant și care din protocoalele analizate se comportă mai bine într-o anumită situație.

Acesta lucrare își propune analiza a mai mulți indicatori cum ar fi: timpul necesar găsirii rutei, durata de viață a rețelei, procentul de recepție al pachetelor de date, throughput-ul, întârzierile end-to-end și încărcătura normalizată.

5.2.1. Timpul necesar găsirii rutei

Acest indicator reprezintă, după cum îi spune și numele, timpul necesar unei rețele de senzori pentru a găsi calea dintre sursî și destinație din momentul în care se dorește trimiterea unui pachet de date și momentul în care acesta reușește să fie trimis. Acest indicator trebuie să fie cât mai mic în rețele

de senzori întrucât, în majoritatea cazurilor, nu trebuie să existe întârzieri în trimiterea informațiilor receptate de senzori.

Formula matematicî al timpului necesar găsirii rutei este:

Ts = moment de timp trimitere pachet de date - moment de timp al apariției dorinței de trimitere a pachetului

5.2.2. Durata de viață a rețelei

Acest indicator este destul de important în rețelele de senzori reflectând gestiunea consumului energetic a nodurilor unei rețele de senzori. Fiecare nod are inițial un anumit nivel energetic de care dispune. Această energie se consumă la trimiterea pachetelor, la recepția pachetelor, pentru a ține nodul pornit și pentru a face trecerea dintre starea de așteptare și starea de sleep a unui nod. În funcție de nivelul pachetelor de control ce trebuiesc trimise valorile energetice de care dispun nodurile scad mai mult sau mai puțin. Dacă un nod nu este solicitat să trimită sau să recepționeze un pachet, dar algoritmul de rutare îi cere să trimită pachete de control pentru a actualiza tabela de rutare, atunci consumul energetic efectuat este inutil. Este de dorit ca rețeaua să aibă o durată cât mai mare de viață și chiar dacă nodurile ce se regăseau în rutele cele mai scurte nu mai au energie, pachetele de date să poată fi trimise prin alte noduri, chiar folosindu-se rute ocolitoare, către destinație.

Formula matematică al timpului de viață este:

Tv= momentul în care s-a trimis ultimul pachet de date − momentul când s-a trimis primul pachet de date

5.2.3. Procentul de recepție al pachetelor de date

Procentul de recepție al pachetelor de date ne oferă o imagine despre cât de bine funcționează un anumit protocol. Acesta se definește ca fiind procentul de pachete de date trimise de sursă ce ajung la destinație. Acest indicator este de dorit să fie cât mai mare întrucât aceasta înseamnă atât siguranța de transmitere a datelor, rapiditate, cât și consum eficient de energie pentru că un pachet pierdut trebuie retrimis.

Formula matematică a acestui indicator este:

$$p = \frac{număr\ pachete\ de\ date\ receptionate * 100}{număr\ pachete\ de\ date\ trimise}$$

5.2.4. Throughput

Throughput-ul reprezintă rata medie de transfer al pachetelor de date ce ajung corect la destinație. Unitatea de măsură a acestui indicator este biți/secundă. În toate rețelele, nu numai în cele de senzori, această caracteristică este una importantă și se dorește ca valoarea acestui parametru să fie cât mai mare.

Formula matematică a thrughput-ului este:

$$T = \frac{număr\ de\ biti}{timpul\ de\ observatie}$$

5.2.5. End-to-end delay

Întârzierea capăt la capăt reprezintă timpul necesar unui pachet de date să ajungă de la sursă la destinație. Acest indicator evaluează performantele protocoalelor în asigurarea căii optime pentru un transfer rapid al datelor. Acest lucru este important pentru că ne dorim ca rețelele de senzori să aibă o funcționare în timp real. Trebuie precizat că sunt luate în calcul numai pachetele de date ce ajung la destinație. Este de dorit să avem întârzieri cât mai mici pentru ca rețeaua să funcționeze în timp real.

Formula matematică a acestui indicator este:

$$D = \sum (\text{moment de timp recepție - moment de timp trimitere})$$

5.2.6. Încărcătura de rutare normalizată

Acest parametru măsoară numărul de pachete de control trimise către nodurile rețelei supra numărul de pachete recepționate la destinație. Cu alte cuvinte este o măsura a nivelului de încărcare a rețelei cu pachete de control comparativ cu numărul de pachete de date trimise. Este de dorit să avem cât mai puține pachete de control pentru a putea folosi energia în scopul trimiterii unor pachete de informație.

Formula matematică a acestui indicator este:

$$rd = \frac{\Sigma\ \text{pachete de rutare}}{\Sigma\ \text{pachete de date}}$$

5.3. Modelarea rețelelor de senzori

In această secțiune sunt prezentate componentele ce ajută la conectarea rețelelor de senzori.

Traficul de date ce va fi utilizat în simulările prezentate ulterior va fi de tip FTP, dar se poate folosi și un alt tip de trafic, CBR, trafic ce are rata de bit constantă. Aceste tipuri de trafic sunt folosite având ca protocol de transport în primul caz TCP, iar în al doilea caz UDP. După cum am spus ca și nivel de transport pot fi folosite protocoalele UDP și TCP care sunt simulați prin atașarea sursei unui agent corespunzător. La nodul destinație trebuie să atașam un agent NULL pentru UDP sau un agent TCPsink. [35]

La nivel rețea vom folosi pe rând protocoalele DSDV, AODV, DSR și AOMDV, ce sunt precizate în diverse publicații ca fiind protocoalele cu rezultatele cele mai bune în rețelele de senzori. Aceste protocoale sunt utilizate pentru a dirija pachetele de date de la sursă către destinație în rețele multi-salt ce se auto-organizează. [35]

La nivelul legăturilor de date sunt utilizate protocoale specifice ce au ca responsabilitate fragmentarea și reasamblarea datelor. Protocolul ARP, ce este utilizat, creează legături în adrese IP și adrese MAC. [35]

Nivelul MAC utilizează protocolul 802.11 IEEE ce folosește mesaje de tipul RTS/CTS și DATA/ACK.

Există și o componentă coada, IFQ, ce specifică câte pachete și tipul cozii de așteptare folosit. Cel mai frecvent utilizată este coada de tip PriQueue ce acordă prioritate pachetelor de rutare și poate elimina pachetele către o adresă specificată. O coadă FIFO este setată atunci când este utilizată Queue/DropTail/PriQueue. Aceasta funcționează pentru AODV, DSDV și AOMDV însă pentru DSR trebuie folosită o coadă CMU PriQueue. [35]

Ca model radio este utilizată atenuarea TwoRayGround pentru distantele mari. Date fiind puterea de transmisie, lungimea de undă și o anumită distanță se poate lua decizia dacă pachetele pot sau nu să fie recepționate de un anumit nod mobil. Acest model de reflexie detectează atât calea directă cât și calea inversă, fiind specificat în multe articole faptul că oferă preconizări mai exacte decât modelul spațiului liber pentru distanțe mari. Puterea de recepție are formula [35]:

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L}$$

unde, P_t – puterea de transmisie
G_t, G_r – câștigurile antenei receptoare/transmițătoare
h_t, h_r – înălțimea antenelor
L – pierdere

Antena folosită este de tip Omni Direcțional adică radiază în toate direcțiile din plan.

5.4. Scenariile

Cel mai important lucru într-un studiu ce cuprinde simulări este alegere scenariilor potrivite. Topologia rețelei, mobilitatea nodurilor dar și modul cum interacționează nodurile sunt aspecte importante și pot influența mult rezultatele. Întotdeauna trebuie alese scenariile care scot în evidență mărimea pe care dorim să o comparăm astfel încât interpretarea rezultatelor să fie clară. Astfel vom avea protocoale ce se vor adapta mai bine la condițiile respective și protocoale care nu vor face față și vor obține rezultate slabe. Într-un alt scenariu se poate întâmpla ca protocoalele ce au avut o comportare bună în scenariul precedent să nu reușească să facă față condițiilor din noul scenariu și invers. Astfel este clar că un singur scenariu nu este suficient pentru a decide ce algoritm de rutare este cel mai bun ci doar putem afirma că pentru acele condiții are rezultate foarte bune.

Știind faptul că rutarea nu este independentă și depinde destul de mult și de celelalte protocoale din nivelele adiacente, nivelul legătură de date și transport, trebuie să se țină cont și de acestea la definirea scenariilor. Există câteva probleme din cele expuse în capitolul 2 de care trebuie, de

asemenea, să ținem seamă. Una dintre acestea ar fi problema legăturilor unidirecționale precum și problema terminalelor ascunse/expuse. Modul de comportare al protocolului TCP are de asemenea un impact important pentru rutare. Astfel chiar și cel mai performant protocol de rutare poate întâmpina dificultăți în obținerea unor rezultate bune. Un alt factor al rețelelor de senzori ce influențează puternic comportarea rețelei este legat de mobilitatea crescută a nodurilor din unele aplicații. Congestia poate afecta rețelele de senzori și trebuie să ținem seamă și de aceasta, iar o altă problemă este numărul de noduri ce compun rețeaua. Cu cât rețeaua are dimensiuni mai mari cu atât performanțele acesteia vor scădea.

In cele ce urmează au fost alese patru scenarii ce își propun să evidențieze lucruri diferite în aplicații diferite ale rețelelor de senzori. Aceste scenarii au fost concepute luând în calcul toți factorii de care am vorbit în paragraful anterior. Se dorește realizarea unei comparații între cele patru protocoale, dar și evidențierea protocolului cu rezultatele cele mai bune în orice fel de aplicație. De bază este evidențierea capacitaților rutării cu căi multiple a protocolului AOMDV, întrucât din analiza matematică a rezultat creșterea cu 25% a performantelor. Scenariile au fost alese astfel încât să fie simulate aplicații reale ale rețelelor de senzori.

5.4.1. Scenariul cu variația numărului de noduri

5.4.1.1. Implementare

In primul scenariu este prezentată o rețea de senzori de tip mesh (plasa). Această rețea este formată la început din 25 de noduri, apoi din 64 de noduri, 100 și în final 196 de noduri. Practic acest scenariu evidențiază comportamentul protocoalelor de rutare atunci când dimensiunea rețelei creste. Sub forma acestui scenariu poate fi modelată una dintre cele mai dese aplicații rețelelor de senzori, asistență în caz de incendiu. Pe o suprafață de diferite dimensiuni există senzori de temperatură ce pot fi localizați. În momentul în care izbucnește un incendiu această rețea ne va putea transmite date referitoare la suprafața și la zona afectată. Practic este de dorit un algoritm de rutare ce are comportarea cea mai bună la creșterea dimensiunii rețelei. În figurile următoare sunt prezentate cele patru cazuri ale rețelei.

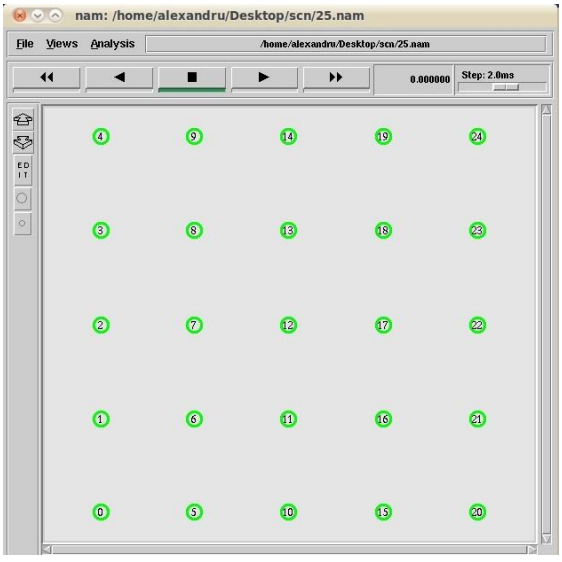


Figura 5.1 Rețea mesh cu 25 de noduri

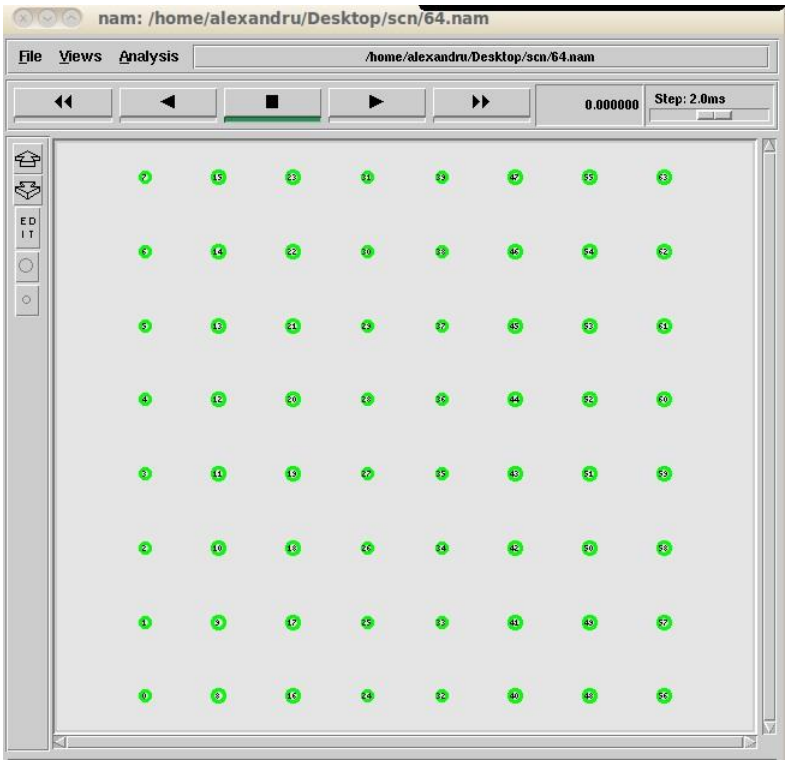


Figura 5.2 Rețea mesh cu 64 de noduri

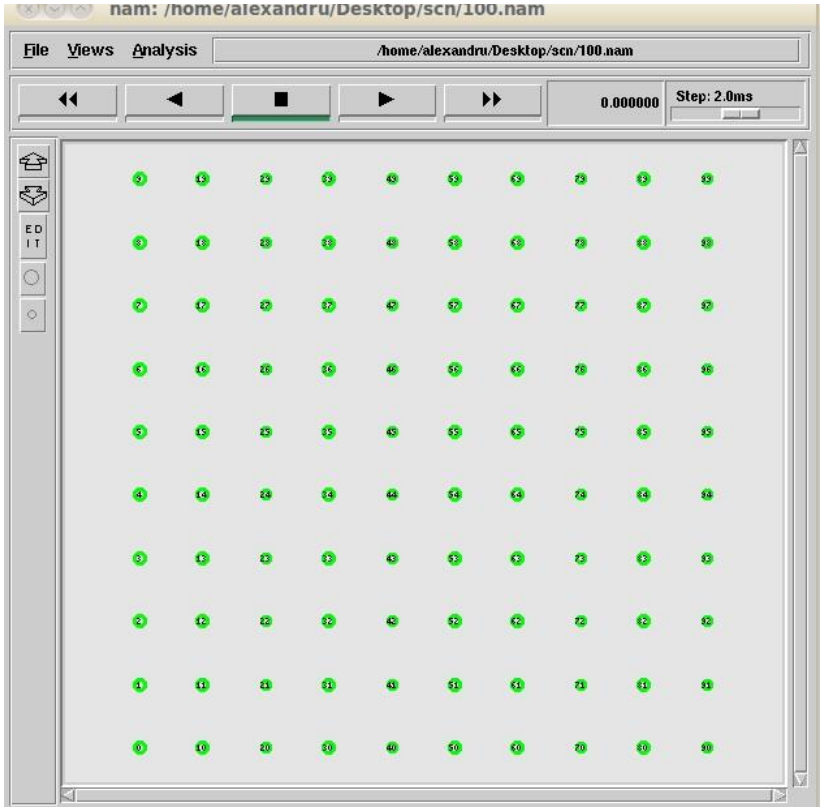


Figura 5.3 Rețea mesh cu 100 de noduri

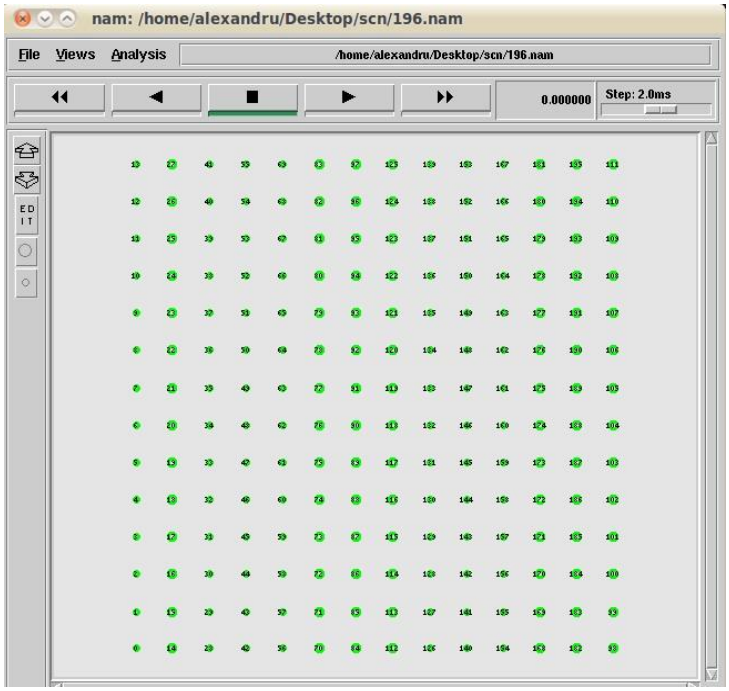


Figura 5.4 Rețea mesh cu 196 de noduri

Traficul este reprezentat de schimbul de informații pe diagonala secundară cu scopul de a simula cele mai lungi rute posibile din acest tip de rețea. Având în vedere faptul că nodurile sunt amplasate pe o suprafață care este practic supravegheată de acești senzori nu există noduri mobile.

Acest scenariu are ca scop evidențierea comportamentului algoritmilor de rutare la diferite dimensiuni ale rețelei de senzori.

In tabelul de mai jos sunt prezentați parametri de start ai rețelei.

Tipul canalului	Channel/WirelessChannel
Modelul de propagare radio	Propagation/TwoRayGround
Tipul de interfața de rețea	Phy/WirelessPhy
Protocolul Legătura de Date	Mac/802.11
Tipul cozii de așteptare	Queue/DropTail/PriQueue (CMUPriQueue pentru DSR)
Modelul de antena	Antenna/OmniAntenna
Numărul maxim de pachete în coada	10
Numărul de noduri	25/64/100/196
Protocolul de rutare	DSDV, AODV, DSR, AOMDV
Dimensiunea topologiei (X)	1000
Dimensiunea topologiei (Y)	1000
Durata simulării	800
Raza de transmisie	142
Lărgimea de banda a canalului	10 Mb/s
Tipul de trafic	TCP
Tipul de aplicație	FTP
Dimensiunea pachetelor	1500 bytes
Energia inițială a nodurilor	500

Tabel 5.1 Configurația inițială scenariul 1

5.4.1.2. Rezultate

25 de noduri	DSDV	AODV	DSR	AOMDV
Start	95.95	10.1	10.21	10.15
Stop	471.32	501.43	489.47	432.1
Timp necesar găsirii rutelor inițiale	85.95	0.1	0.21	0.15
Timp de viața rețea	375.37	491.33	479.26	421.95
Număr total de pachete trimise	258920	256883	279752	305163
Număr total de pachete primite	309166	348262	315496	427759
Număr total de pachete pierdute	9201	10268	9430	11776
Număr total de pachete de **date** trimise	4382	4267	4195	4749
Număr total de pachete de **date** recepționate	4277	4232	4188	4641
Procent recepție [%]	97.60	99.17	99.83	97.72
Throughput [Kbps]	91.32	70.54	71.57	92.08
Întârzierile capăt la capăt [ms]	414.885	337.982	1011.92	408.621
Încărcătura de rutare normalizata	0.385	2.721	0.334	2.936

Tabel 5.2 Rezultate obținute pentru 25 de noduri

64 de noduri	DSDV	AODV	DSR	AOMDV
Start	156.84	10.2	10.32	10.21
Stop	594.23	581.87	524.29	562.54
Timp necesar găsirii rutelor inițiale	146.84	0.2	0.32	0.21
Timp de viața rețea	437.39	571.67	513.97	552.33
Număr total de pachete trimise	336170	437451	516275	475582
Număr total de pachete primite	447256	610848	599726	745070
Număr total de pachete pierdute	16140	20633	28262	33659
Număr total de pachete de **date** trimise	2934	3771	3799	3748
Număr total de pachete de **date** recepționate	2743	3373	3784	3585
Procent recepție [%]	93.49	89.44	99.60	92.98
Throughput [Kbps]	51.36	48.32	60.3	51.67
Întârzierile capăt la capăt [ms]	369.864	339.355	737.302	321.712
Încărcătura de rutare normalizata	3.607	6.332	2.852	10.845

Tabel 5.3 Rezultate obținute pentru 64 de noduri

100 de noduri	DSDV	AODV	DSR	AOMDV
Start	236.75	10.26	-	10.29
Stop	651.15	631.03	-	583.94
Timp necesar găsirii rutelor inițiale	226.75	0.26	-	0.29
Timp de viața rețea	414.4	620.77	-	573.65
Număr total de pachete trimise	387818	521389	4514	616780
Număr total de pachete primite	610743	757278	35295	1046219
Număr total de pachete pierdute	22861	24105	1272	42336
Număr total de pachete de **date** trimise	2343	3352	17	3504
Număr total de pachete de **date** recepționate	2208	3024	-	3308
Procent recepție [%]	94.23	90.21	-	94.40
Throughput [Kbps]	43.61	39.89	-	47.23
Întârzierile capăt la capăt [ms]	454.115	414.332	-	395.338
Încărcătura de rutare normalizata	12.965	10.164	-	18.814

Tabel 5.4 Rezultate obținute pentru 100 de noduri

196 de noduri	DSDV	AODV	DSR	AOMDV
Start	303.08	10.39	-	10.47
Stop	631.53	686.43	-	625.87
Timp necesar găsirii rutelor inițiale	293.08	0.39	-	0.47
Timp de viața rețea	328.45	676.04	-	615.4
Număr total de pachete trimise	465404	648875	7768	934507
Număr total de pachete primite	1063844	960448	62281	1831824
Număr total de pachete pierdute	39158	29169	2625	62310
Număr total de pachete de **date** trimise	1352	2877	21	3201
Număr total de pachete de **date** recepționate	1258	2582	-	3055
Procent recepție [%]	93.04733728	89.74626347	-	95.43892534
Throughput [Kbps]	31.35	31.28	-	40.65
Întârzierile capăt la capăt [ms]	627.832	500.631	-	519.603
Încărcătura de rutare normalizata	75.622	15.63	-	44.861

Tabel 5.5 Rezultate obținute pentru 196 de noduri

5.4.2. Scenariul cu variația numărului de clustere

5.4.2.1. Implementare

In cel de-al doilea scenariu este modelată o rețea ce cuprinde pe rând 1, 2, 3, 4 clustere. Acest scenariu evidențiază comportarea unei rețele de senzori în funcție de cate clustere sunt atașate unui nod colector aflat în centru. Clusterele sunt formate din noduri omogene.

Acest scenariu poate modela o altă posibilă aplicație a rețelelor de senzori. Aceasta se referă la plasarea de noduri senzor în zone greu accesibile ale mașinilor, utilajelor, ale unei fabrici, astfel încât ele să poată detecta posibile evenimente cum ar fi erori de funcționare, starea mecanismelor, etc. Avantajul primordial pe care îl au rețelele de senzori în acest caz este independența de cabluri. Acest lucru face ca instalarea să fie ieftină, întreținerea de asemenea și face ca un astfel de sistem să poată fi instalat după ce utilajul a fost dat în folosința. Pe lângă parametri clasici ai unei rețele, throughput, end-to-end delay, încărcătura de rutare normalizată, aici ne interesează și un consum eficient al energiei bateriilor cu care sunt dotate nodurile. Acest lucru duce la creșterea timpului de utilizare al rețelei întrucât schimbarea bateriilor este foarte dificil sau chiar imposibil de efectuat.

Traficul din acest scenariu va fi mai intens întrucât față de primul scenariu în fiecare cluster există trei surse de trafic, nodurile extreme. Toate aceste noduri folosesc un trafic de tip FTP ce utilizează protocolul TCP și trimit pachete de date către nodul colector aflat în centrul topologiei. Intrucât nodurile sunt amplasate pe utilaje am considerat că nodurile rețelelor sunt fixe. În figurile următoare sunt prezentate cele patru cazuri ale rețelei.

Nodurile sunt dispuse la distanțe egale de 100m, iar distanța de transmisie a fiecărui nod este de 142m.

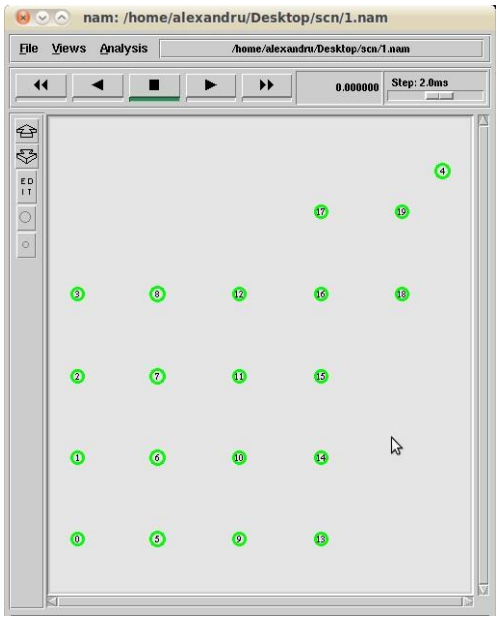


Figura 5.5 Rețea având un cluster

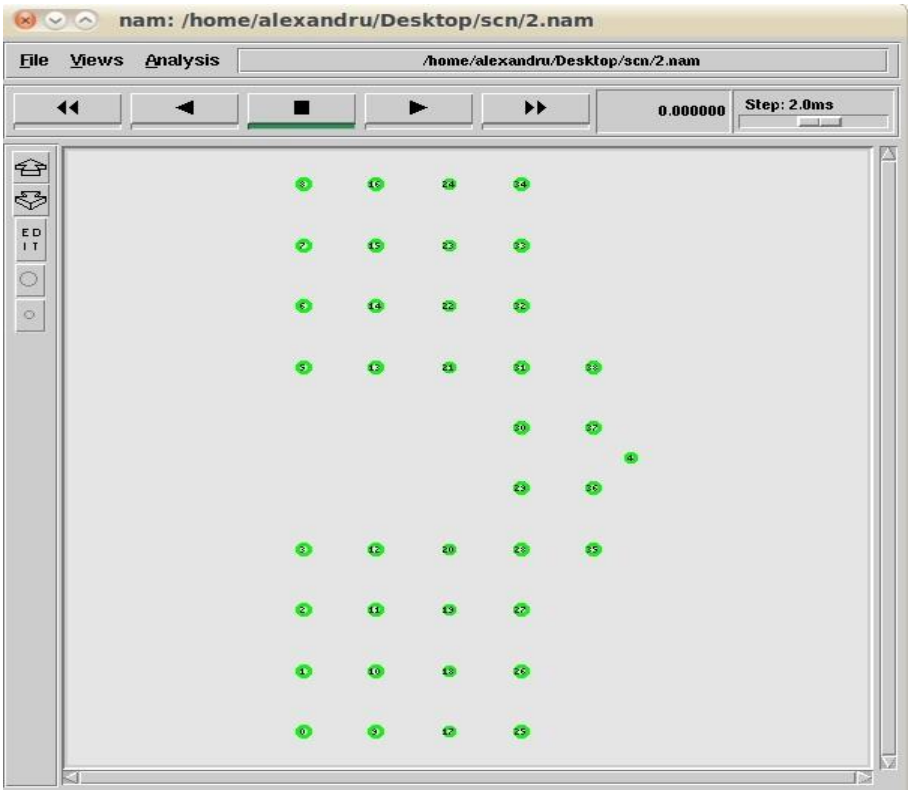


Figura 5.6 Rețea având doua clustere

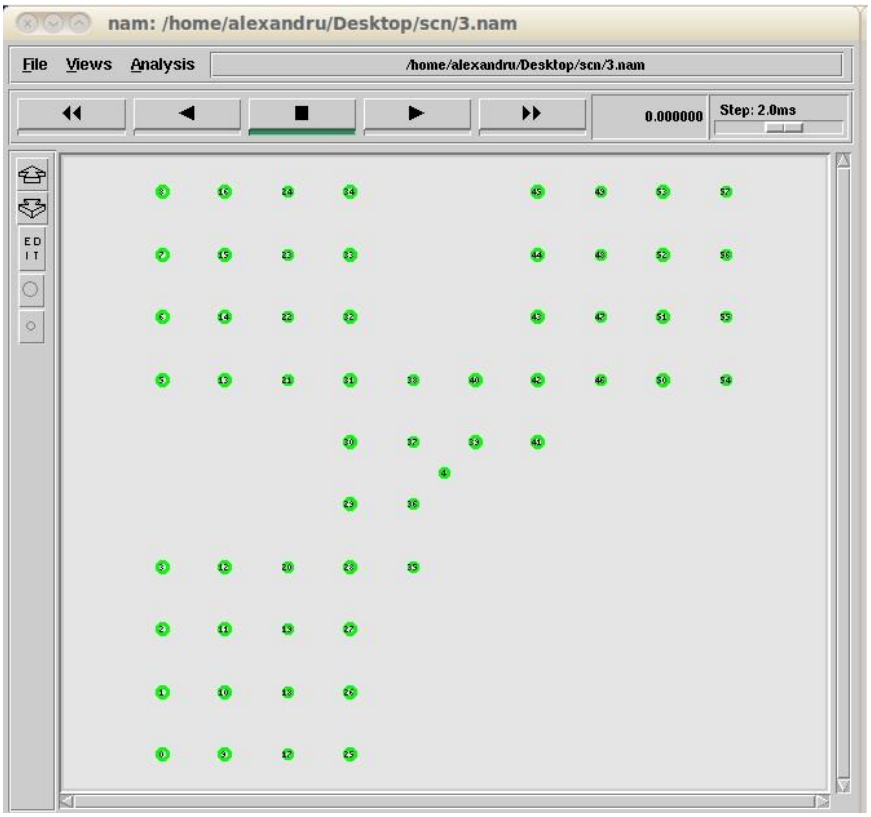


Figura 5.7 Rețea având trei clustere

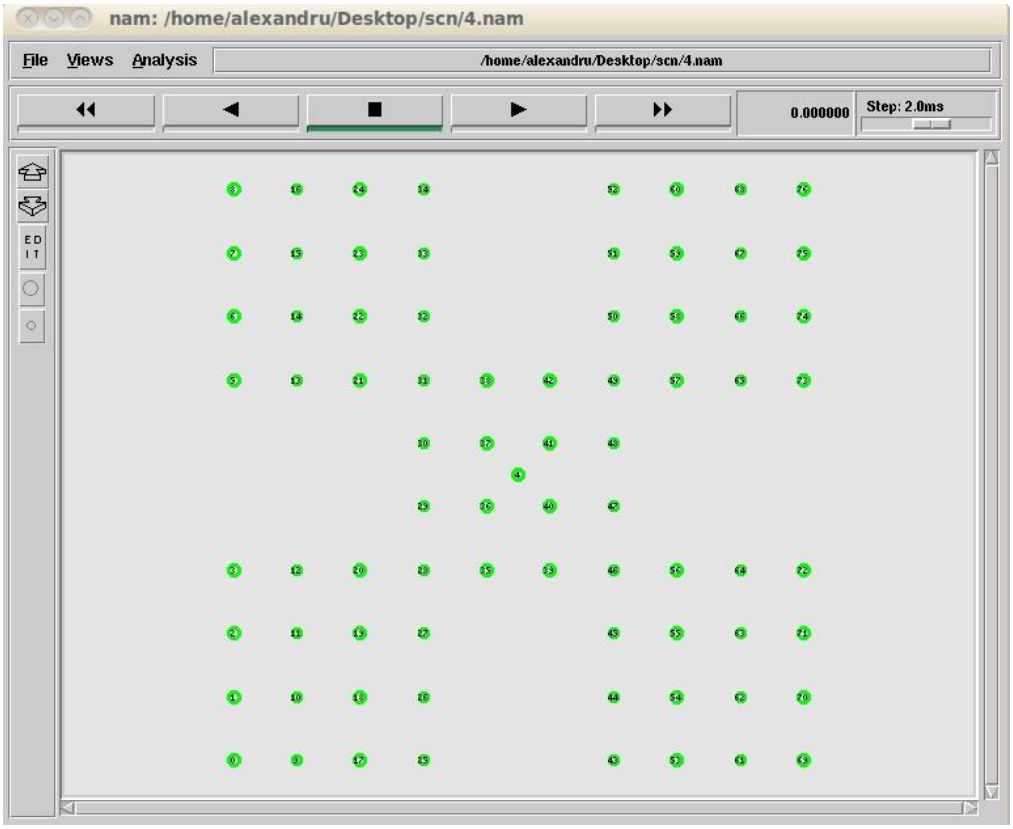


Figura 5.8 Rețea având patru clustere

Parametri inițiali sunt prezentați în tabelul de mai jos.

Tipul canalului	Channel/WirelessChannel
Modelul de propagare radio	Propagation/TwoRayGround
Tipul de interfața de rețea	Phy/WirelessPhy
Protocolul Legătura de Date	Mac/802.11
Tipul cozii de așteptare	Queue/DropTail/PriQueue (CMUPriQueue pentru DSR)
Modelul de antena	Antenna/OmniAntenna
Numărul maxim de pachete în coada	10
Numărul de noduri	18/cluster
Protocolul de rutare	DSDV, AODV, DSR, AOMDV
Dimensiunea topologiei (X)	1000
Dimensiunea topologiei (Y)	1000
Durata simulării	800
Raza de transmisie	142
Lărgimea de banda a canalului	10 Mb/s
Tipul de trafic	TCP
Tipul de aplicație	FTP
Dimensiunea pachetelor	1500 bytes
Energia inițială a nodurilor	500

Tabel 5.6 Configurația inițială scenariul 2

5.4.2.2. Rezultate

1 cluster	DSDV	AODV	DSR	AOMDV
Start	132.03	10.14	10.24	10.15
Stop	466.73	391.24	394.72	400.3
Timp necesar găsirii rutelor inițiale	122.03	0.14	0.24	0.15
Timp de viața rețea	334.7	381.1	384.48	390.15
Număr total de pachete trimise	238692	278410	298345	290486
Număr total de pachete primite	287808	333890	347438	398377
Număr total de pachete pierdute	1613	1692	5037	4885
Număr total de pachete de **date** trimise	4568	5378	5037	4885
Număr total de pachete de **date** recepționate	4430	5202	4993	4723
Procent recepție [%]	96.97	96.72	99.12	96.68
Throughput [Kbps]	108.35	111.76	106.3	99.17
Întârzierile capăt la capăt [ms]	691.178	701.029	2083.91	679.22
Încărcătura de rutare normalizata	0.29	0.201	0.31	2.435

Tabel 5.7 Rezultate obținute pentru 1 cluster

2 clustere	DSDV	AODV	DSR	AOMDV
Start	72.13	10.29	10.64	10.24
Stop	482.08	450.22	471.62	486.86
Timp necesar găsirii rutelor inițiale	62.13	0.29	0.64	0.24
Timp de viața rețea	409.95	439.93	460.98	476.62
Număr total de pachete trimise	294641	324440	380518	383312
Număr total de pachete primite	359153	394002	446632	555063
Număr total de pachete pierdute	3954	4216	15159	16788
Număr total de pachete de **date** trimise	5546	5996	5830	5721
Număr total de pachete de **date** recepționate	5341	5743	5694	5476
Procent recepție [%]	96.30	95.78	97.66	95.71
Throughput [Kbps]	106.61	106.81	101.08	94
Întârzierile capăt la capăt [ms]	822.74	856.635	1612.26	605.089
Încărcătura de rutare normalizata	0.551	0.597	1.338	4.601

Tabel 5.8 Rezultate obținute pentru 2 clustere

3 clustere	DSDV	AODV	DSR	AOMDV
Start	61.33	10.31	11.28	10.35
Stop	498.04	478.1	481.2	477.36
Timp necesar găsirii rutelor inițiale	51.33	0.31	1.28	0.35
Timp de viața rețea	436.71	467.79	469.92	467.01
Număr total de pachete trimise	334219	381531	485617	442946
Număr total de pachete primite	409910	490767	567668	687024
Număr total de pachete pierdute	9104	10437	32485	26810
Număr total de pachete de **date** trimise	6348	6680	5721	6228
Număr total de pachete de **date** recepționate	5777	6172	5462	5603
Procent receptie [%]	91.00	92.39	95.47	89.96
Throughput [Kbps]	108.14	107.89	95.01	98.09
Întârzierile capăt la capăt [ms]	717.453	812.814	2452.87	652.492
Încărcătura de rutare normalizata	0.885	2.049	3.291	7.836

Tabel 5.9 Rezultate obținute pentru 3 clustere

4 clustere	DSDV	AODV	DSR	AOMDV
Start	55.03	10.48	11.59	10.52
Stop	507.55	488.87	489.9	494.82
Timp necesar găsirii rutelor inițiale	45.03	0.48	1.59	0.52
Timp de viața rețea	452.52	478.39	478.31	484.3
Număr total de pachete trimise	357302	406090	514799	478319
Număr total de pachete primite	441384	542975	604854	794948
Număr total de pachete pierdute	12419	15547	39371	37085
Număr total de pachete de **date** trimise	8278	8343	6313	6946
Număr total de pachete de **date** recepționate	7520	7545	6101	6225
Procent recepție [%]	90.84	90.43	96.64	89.61
Throughput [Kbps]	135.92	128.89	104.17	105.03
Întârzierile capăt la capăt [ms]	679.284	662.749	3169.64	628.532
Încărcătura de rutare normalizata	1.004	2.59	3.654	9.706

Tabel 5.10 Rezultate obținute pentru 4 clustere

5.4.3. Scenariul cu variația vitezei de mișcare

5.4.3.1. Implementare

Scenariul acesta a fost gândit pentru a simula o rețea de senzori formată din 40 de noduri ce se deplasează cu diferite viteze pe direcții aleatorii. Acest scenariu vrea să evidențieze comportarea celor patru protocoale de rutare atunci când nodurile au o mobilitate continuă. Presupunând că se dorește monitorizarea unor fenomene ale naturii sau a unor animale în sălbăticie sau se dorește urmărirea intrușilor într-o zona de securitate, rețelele de senzori trebuie să facă față cu brio la mișcările nodurilor.

Traficul de pachete este unul intens având 30 de surse de trafic de tip FTP ce folosesc ca protocol de transport TCP.

Mobilitatea nodurilor este și ea destul de mare, nodurile mișcându-se pe rând cu viteza de 1 m/s, 5 m/s, 10 m/s, 20 m/s.

In imaginile de mai jos este prezentată desfășurarea rețelei pe parcursul simulării.

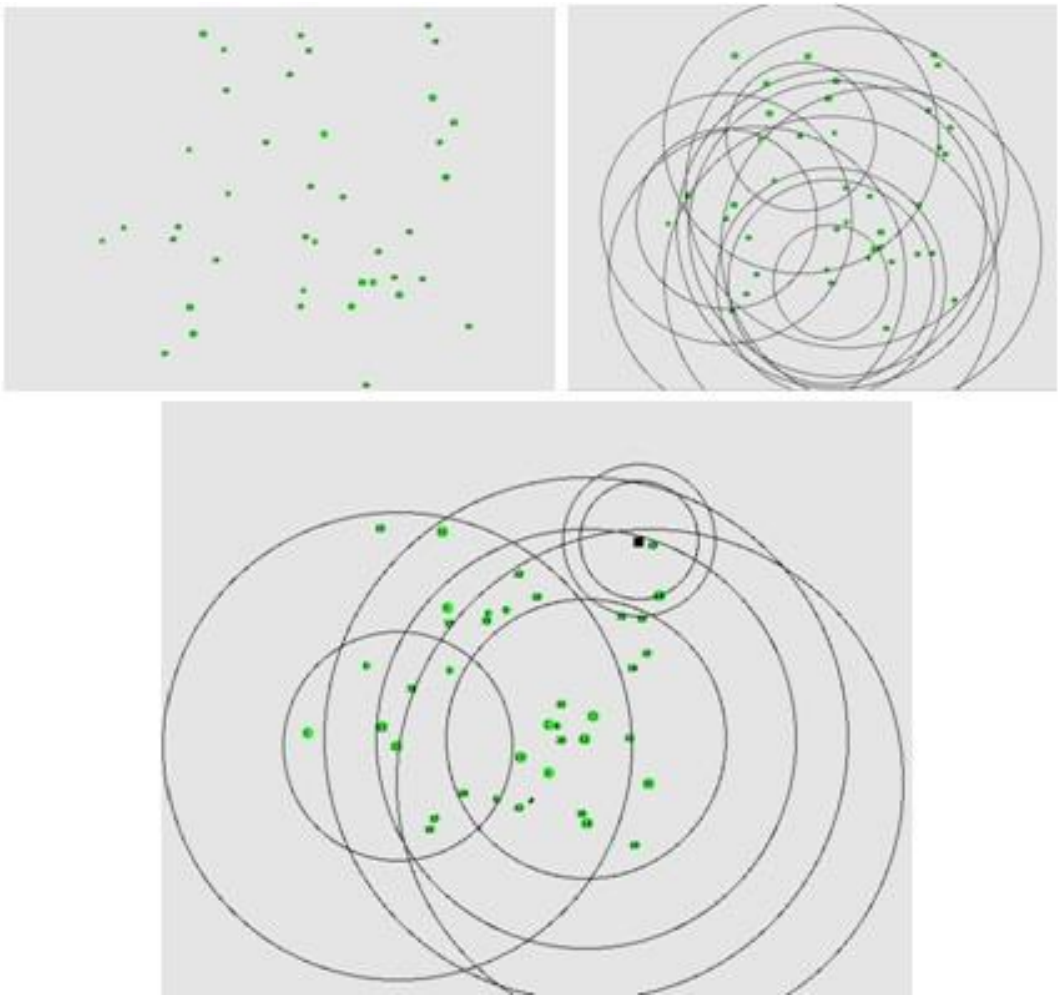


Figura 5.9 Inițializare rețelei

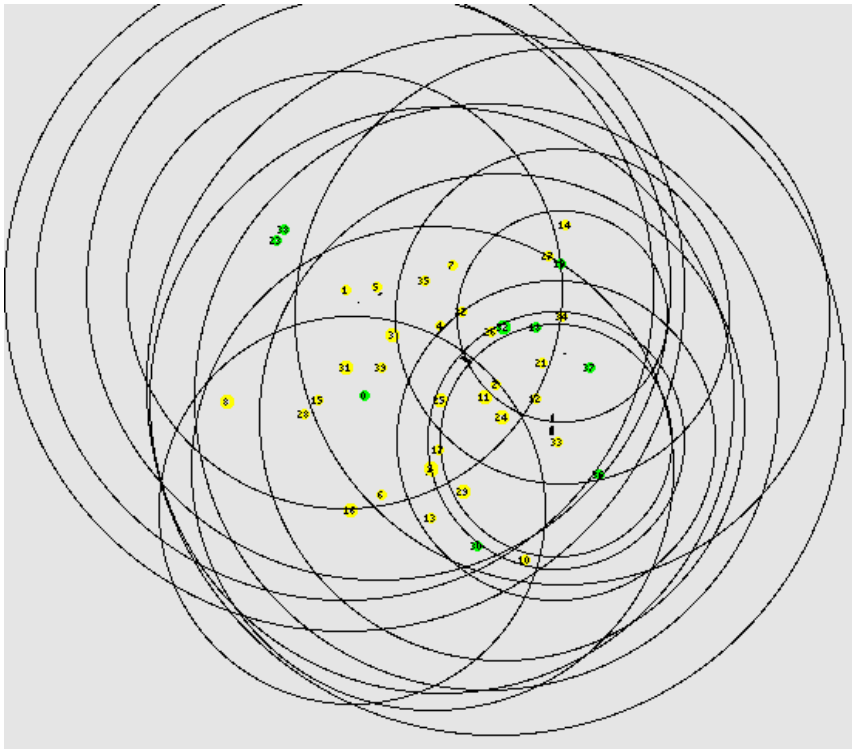


Figura 5.10 Nodurile rețelei cu galben au un nivel de energie mediu

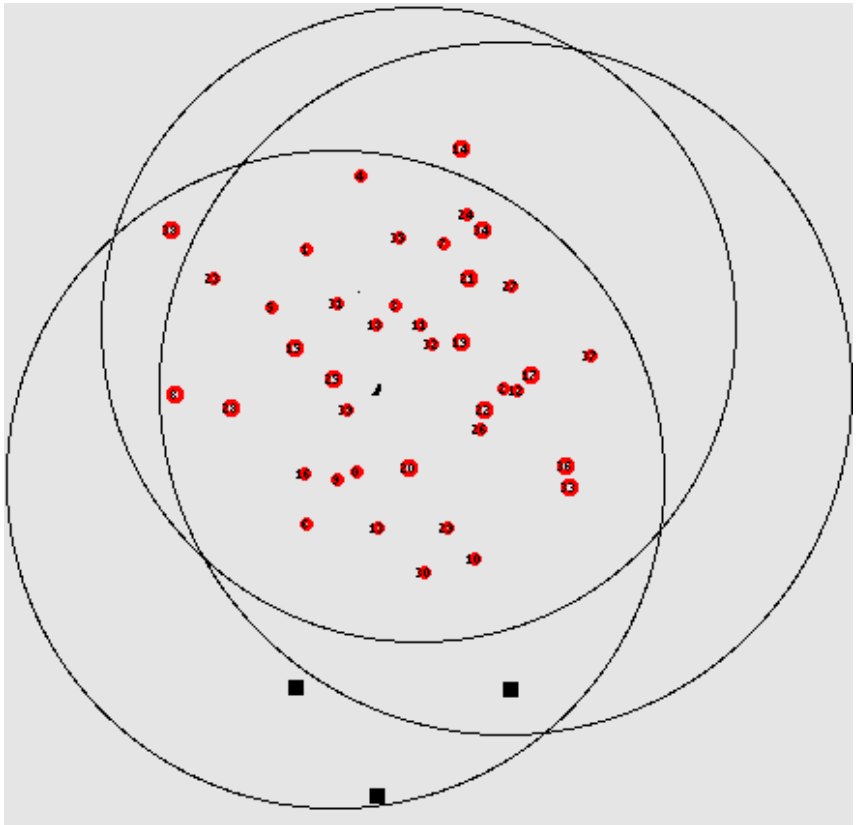


Figura 5.11 Nodurile rețelei cu roșu au un nivel de energie foarte scăzut

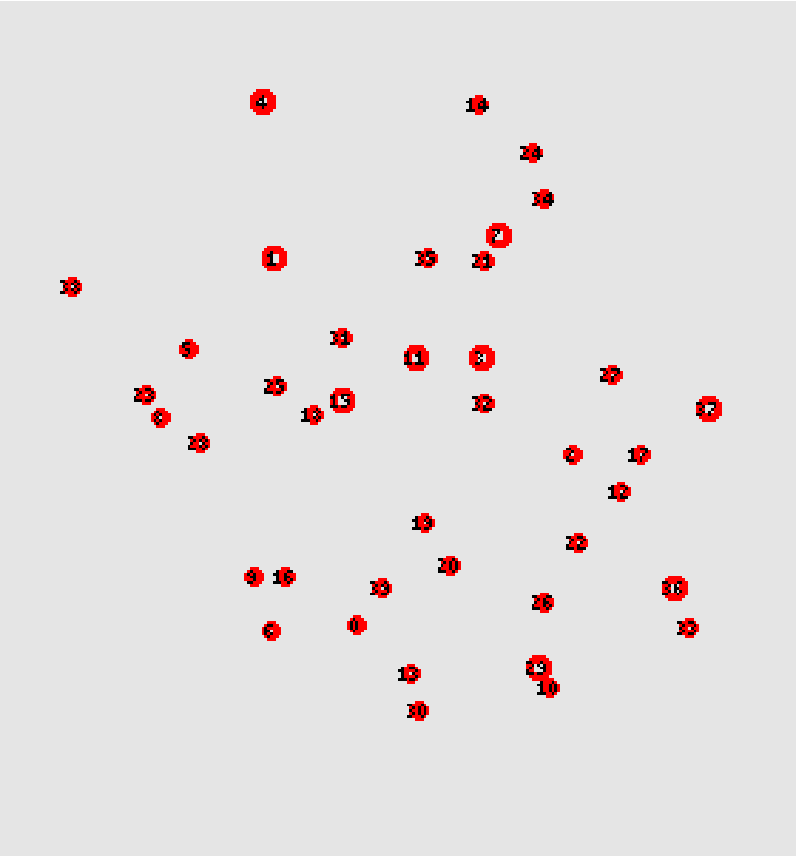


Figura 5.12 Nodurile rețelei cu roșu nu mai au energie si nu mai transmit

Parametri inițiali sunt prezentați în tabelul de mai jos:

Tipul canalului	Channel/WirelessChannel
Modelul de propagare radio	Propagation/TwoRayGround
Tipul de interfața de rețea	Phy/WirelessPhy
Protocolul Legătura de Date	Mac/802.11
Tipul cozii de așteptare	Queue/DropTail/PriQueue (CMUPriQueue pentru DSR)
Modelul de antena	Antenna/OmniAntenna
Numărul maxim de pachete în coada	10
Numărul de noduri	40
Protocolul de rutare	DSDV, AODV, DSR, AOMDV
Dimensiunea topologiei (X)	500
Dimensiunea topologiei (Y)	500
Durata simulării	800
Raza de transmisie	142
Lărgimea de banda a canalului	10 Mb/s
Tipul de trafic	TCP
Tipul de aplicație	FTP
Dimensiunea pachetelor	1500 bytes
Energia inițială a nodurilor	500

Tabel 5.11 Configurația inițială scenariul 3

5.4.3.2. Rezultate

viteza 1 m/s	AODV	DSR	AOMDV
Start	10.27	10.04	10.32
Stop	498.01	508.45	506.94
Timp necesar găsirii rutelor inițiale	0.27	0.04	0.32
Timp de viața rețea	487.74	498.41	496.62
Număr total de pachete trimise	501384	569170	530568
Număr total de pachete primite	1048874	589876	1423255
Număr total de pachete pierdute	151401	31492	164803
Număr total de pachete de **date** trimise	32765	28201	33679
Număr total de pachete de **date** recepționate	30025	27748	31053
Procent recepție [%]	91.63	98.39	92.20
Throughput [Kbps]	503.66	455.11	528.93
Întârzierile capăt la capăt [ms]	598.113	4191.53	562.284
Încărcătura de rutare normalizata	0.667	0.494	1.243

Tabel 5.12 Rezultate obținute pentru viteza de 1 m/s

viteza 5 m/s	AODV	DSR	AOMDV
Start	10.51	10.04	10.53
Stop	499.94	505.44	501.33
Timp necesar găsirii rutelor inițiale	0.51	0.04	0.53
Timp de viața rețea	489.43	495.4	490.8
Număr total de pachete trimise	514384	597317	544717
Număr total de pachete primite	1384083	613810	1529052
Număr total de pachete pierdute	213293	32358	162136
Număr total de pachete de **date** trimise	32525	28519	33542
Număr total de pachete de **date** recepționate	29161	28122	30716
Procent recepție [%]	89.65	98.60	91.57
Throughput [Kbps]	476.74	464.07	512.02
Întârzierile capăt la capăt [ms]	494.634	3693.76	467.819
Încărcătura de rutare normalizata	1.044	0.701	1.31

Tabel 5.13 Rezultate obținute pentru viteza de 5 m/s

viteza 10 m/s	AODV	DSR	AOMDV
Start	10.61	10.04	10.66
Stop	502.97	511.75	504.39
Timp necesar găsirii rutelor inițiale	0.61	0.04	0.66
Timp de viața rețea	492.36	501.71	493.73
Număr total de pachete trimise	515899	607798	544117
Număr total de pachete primite	1367813	639053	1443985
Număr total de pachete pierdute	179884	33986	129890
Număr total de pachete de **date** trimise	32023	25413	33313
Număr total de pachete de **date** recepționate	28659	25075	30659
Procent recepție [%]	89.49	98.66	92.03
Throughput [Kbps]	476.22	408.26	507.97
Întârzierile capăt la capăt [ms]	445.788	4074.95	380.885
Încărcătura de rutare normalizata	1.156	1.015	1.318

Tabel 5.14 Rezultate obținute pentru viteza de 10 m/s

viteza 15 m/s	AODV	DSR	AOMDV
Start	10.6	10.05	10.68
Stop	501.88	512.87	506.11
Timp necesar găsirii rutelor inițiale	0.6	0.05	0.68
Timp de viața rețea	491.28	502.82	495.43
Număr total de pachete trimise	509911	653651	535951
Număr total de pachete primite	1324788	722219	1377615
Număr total de pachete pierdute	151995	41062	100503
Număr total de pachete de **date** trimise	30768	24648	33656
Număr total de pachete de **date** recepționate	27738	24263	30726
Procent recepție [%]	90.15	98.43	91.29
Throughput [Kbps]	461.78	394.22	507.4
Întârzierile capăt la capăt [ms]	373.069	3969.09	314.572
Încărcătura de rutare normalizata	1.301	1.767	1.344

Tabel 5.15 Rezultate obținute pentru viteza de 15 m/s

viteza 20 m/s	AODV	DSR	AOMDV
Start	10.75	10.05	10.81
Stop	502.26	530.76	505.62
Timp necesar găsirii rutelor inițiale	0.75	0.05	0.81
Timp de viața rețea	491.51	520.71	494.81
Număr total de pachete trimise	514046	761607	530221
Număr total de pachete primite	1321934	841492	1303032
Număr total de pachete pierdute	133411	60288	87285
Număr total de pachete de **date** trimise	30209	18223	32877
Număr total de pachete de **date** recepționate	27222	17759	30150
Procent recepție [%]	90.11	97.45	91.70
Throughput [Kbps]	452.97	278.4	501.51
Întârzierile capăt la capăt [ms]	316.107	5824.35	243.355
Încărcătura de rutare normalizata	1.464	5.487	1.412

Tabel 5.16 Rezultate obținute pentru viteza de 20 m/s

5.4.4. Scenariul cu variația frecvenței de mișcare

5.4.4.1. Implementare

Acest scenariu a fost gândit astfel încât să evidențieze comportamentul protocoalelor de rutare în cazul în care nodurile rețelei de senzori au mișcări simultane la o anumită frecvență. Acest lucru se poate transpune în realitate în cerințe de urmărire a unor animale sălbatice sau a unor fenomene ale naturii ce au loc cu diferite frecvențe și solicită rețeaua diferit în funcție de cât de des se produc aceste evenimente.

Traficul de pachete este unul intens având 30 de surse de trafic de tip FTP ce folosesc ca protocol de transport TCP.

Mobilitatea nodurilor este și ea destul de mare, nodurile mișcându-se pe rând cu viteza de 1/10 Hz, 1/50 Hz, 1/100 Hz, 1/150 Hz, 1/200 Hz.

In imaginile de mai jos este prezentată desfășurarea rețelei pe parcursul simulării.

Tipul canalului	Channel/WirelessChannel
Modelul de propagare radio	Propagation/TwoRayGround
Tipul de interfața de rețea	Phy/WirelessPhy
Protocolul Legătura de Date	Mac/802.11
Tipul cozii de așteptare	Queue/DropTail/PriQueue (CMUPriQueue pentru DSR)
Modelul de antena	Antenna/OmniAntenna
Numărul maxim de pachete în coada	10
Numărul de noduri	40
Protocolul de rutare	DSDV, AODV, DSR, AOMDV
Dimensiunea topologiei (X)	500
Dimensiunea topologiei (Y)	500
Durata simulării	800
Raza de transmisie	142
Lărgimea de banda a canalului	10 Mb/s
Tipul de trafic	TCP
Tipul de aplicație	FTP
Dimensiunea pachetelor	1500 bytes
Energia inițială a nodurilor	500

Tabel 5.17 Configurația inițială pentru scenariul 4

5.4.4.2. Rezultate

frecvență 1/200 Hz	AODV	DSR	AOMDV
Start	10.71	10.06	10.7
Stop	500.39	509.09	511.36
Timp necesar găsirii rutelor inițiale	0.71	0.06	0.7
Timp de viața rețea	489.68	499.03	500.66
Număr total de pachete trimise	472624	541896	492608
Număr total de pachete primite	654843	567638	900379
Număr total de pachete pierdute	36423	21471	58847
Număr total de pachete de **date** trimise	27129	24944	26380
Număr total de pachete de **date** recepționate	25171	24485	25085
Procent recepție [%]	92.78	98.15	95.09
Throughput [Kbps]	420.45	401.16	436.73
Întârzierile capăt la capăt [ms]	662.958	4079.9	568.274
Încărcătura de rutare normalizata	0.609	0.719	1.576

Tabel 5.18 Rezultate obținute pentru frecvența de 1/200 Hz

frecvență 1/150 Hz	AODV	DSR	AOMDV
Start	10.71	10.06	10.7
Stop	506.63	507.11	506.42
Timp necesar găsirii rutelor inițiale	0.71	0.06	0.7
Timp de viața rețea	495.92	497.05	495.72
Număr total de pachete trimise	461795	552749	494802
Număr total de pachete primite	641585	572314	897407
Număr total de pachete pierdute	35525	21615	54403
Număr total de pachete de **date** trimise	26663	26523	26725
Număr total de pachete de **date** recepționate	24755	25969	25172
Procent recepție [%]	92.84	97.91	94.18
Throughput [Kbps]	408.28	427.21	416.08
Întârzierile capăt la capăt [ms]	645.163	3365.11	541.868
Încărcătura de rutare normalizata	0.629	0.634	1.516

Tabel 5.19 Rezultate obținute pentru frecvența de 1/150 Hz

frecvență 1/100 Hz	AODV	DSR	AOMDV
Start	10.35	10.05	10.81
Stop	489.9	500.83	514.16
Timp necesar găsirii rutelor inițiale	0.35	0.05	0.81
Timp de viața rețea	479.55	490.78	503.35
Număr total de pachete trimise	451750	569960	493800
Număr total de pachete primite	661877	596154	902160
Număr total de pachete pierdute	34695	21166	54455
Număr total de pachete de **date** trimise	24742	26297	25498
Număr total de pachete de **date** recepționate	22857	25766	23903
Procent recepție [%]	92.38	97.98	93.74
Throughput [Kbps]	389.73	428.86	399.23
Întârzierile capăt la capăt [ms]	598.265	3045.44	495.862
Încărcătura de rutare normalizata	0.775	0.944	1.73

Tabel 5.20 Rezultate obținute pentru frecvența de 1/100 Hz

frecvență 1/50 Hz	AODV	DSR	AOMDV
Start	10.35	10.05	10.81
Stop	494.29	520.04	500.42
Timp necesar găsirii rutelor inițiale	0.35	0.05	0.81
Timp de viața rețea	483.94	509.99	489.61
Număr total de pachete trimise	474631	637813	497620
Număr total de pachete primite	787124	663715	974878
Număr total de pachete pierdute	46111	34628	53341
Număr total de pachete de **date** trimise	26678	23225	27598
Număr total de pachete de **date** recepționate	24549	22638	25526
Procent recepție [%]	92.01	97.47	92.49
Throughput [Kbps]	414.85	362.52	435.42
Întârzierile capăt la capăt [ms]	454.061	2883.98	359.072
Încărcătura de rutare normalizata	1.033	2.35	1.702

Tabel 5.21 Rezultate obținute pentru frecvența de 1/50 Hz

frecvență 1/10 Hz	AODV	DSR	AOMDV
Start	10.35	10.05	10.81
Stop	501.86	530.76	505.62
Timp necesar găsirii rutelor inițiale	0.35	0.05	0.81
Timp de viața rețea	491.51	520.71	494.81
Număr total de pachete trimise	514046	761607	530221
Număr total de pachete primite	1321934	841492	1303032
Număr total de pachete pierdute	133411	60288	87285
Număr total de pachete de **date** trimise	30209	18223	32877
Număr total de pachete de **date** recepționate	27222	17759	30150
Procent recepție [%]	90.11	97.45	91.70
Throughput [Kbps]	452.97	278.4	501.51
Întârzierile capăt la capăt [ms]	316.107	5824.35	243.355
Încarcatura de rutare normalizata	1.464	5.487	1.412

Tabel 5.22 Rezultate obținute pentru frecvența de 1/10 Hz

5.5. Analiza și interpretare rezultate

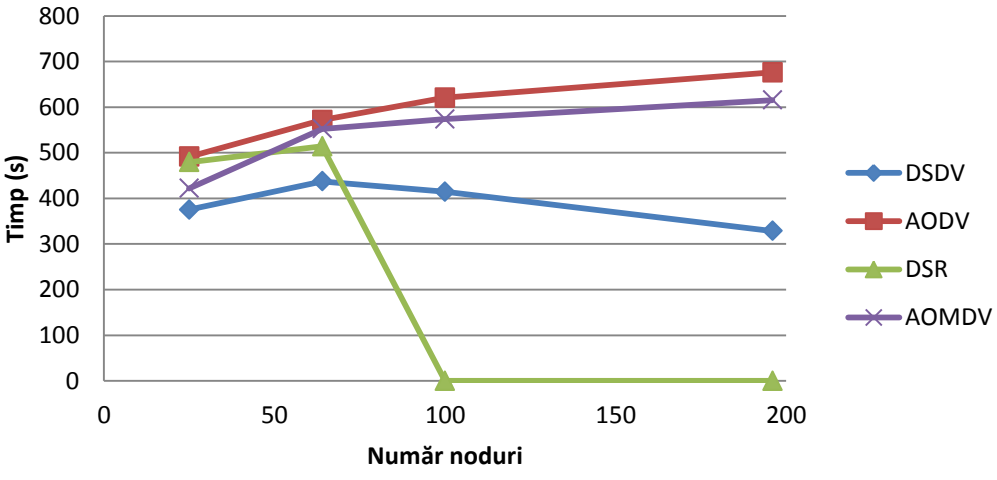
5.5.1. Scenariul cu variația numărului de noduri

Acest scenariu a avut drept scop evidențierea celui mai bun protocol de rutare pentru rețele de senzori cu dimensiuni din ce în ce mai mari. Cu alte cuvinte se dorește identificarea unui protocol scalabil. Un alt obiectiv a fost acela de a pune în evidență faptul că un număr mare de noduri pune în dificultate și protocoalele reactive. Acesta este motivul pentru care am și ales ca traficul să se desfășoare între două noduri aflate pe diagonală, adică la distanța cea mai mare din topologia în formă de pătrat.

Din punctul de vedere al timpului de viață al rețelei:

- se observă cum în cazul algoritmilor AODV și AOMDV timpul de viață al rețelei crește întrucât rutele cele mai scurte ce conțin nodurile ce își pierd energia rapid pot fi înlocuite ușor cu alte rute care deși sunt ocolitoare pot conduce în continuare pachetele de date de la sursă la destinație;
- algoritmul DSDV oferă o scădere a timpului de viață a rețelei întrucât un protocol pro-activ. Modul cum funcționează acest algoritm face ca energia să fie consumată inutil prin încercarea sa de a afla rutele către toate destinațiile posibile fără a fi necesar ;
- protocolul DSR nu funcționează pentru mai mult de 64 de noduri lucru confirmat de teorie. Acesta este un algoritm gândit pentru o mobilitate ridicată a nodurilor dar nu pentru a fi scalabil la rețele de dimensiuni mari;
- cel mai performant protocol este AODV, în acest caz dovedindu-se că identificarea de rute multiple a protocolului AOMDV este inutilă și consumatoare de energie.

Figura 5.13 Timpul de viață al retelei



Din punctul de vedere a timpului de inițializare a rutelor:

- cel mai slab protocol din acest punct de vedere este DSDV care are un timp de începere a transmisiei cu mult mai mare decât a celorlalte protocoale indiferent de dimensiunea rețelei fiind un protocol pro-activ;
- cel mai bun protocol și din acest punct de vedere este AODV care asigură cel mai scurt timp până la începerea transmisiei datelor urmat din apropiere de AOMDV ce obține rezultate apropiate și cu mult mai bune decât a celorlalte două protocoale;

Figura 5.14aTimpul de initializare rute

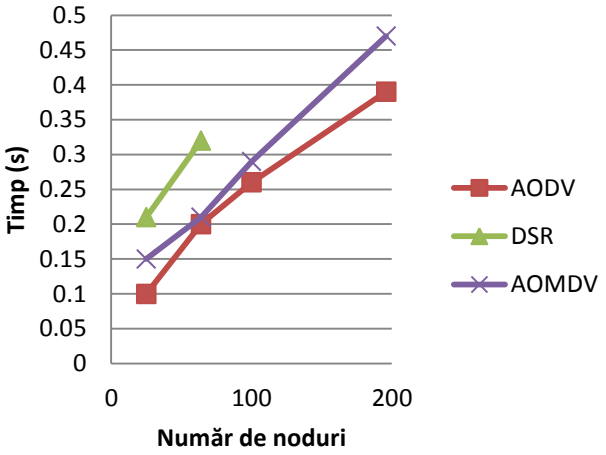
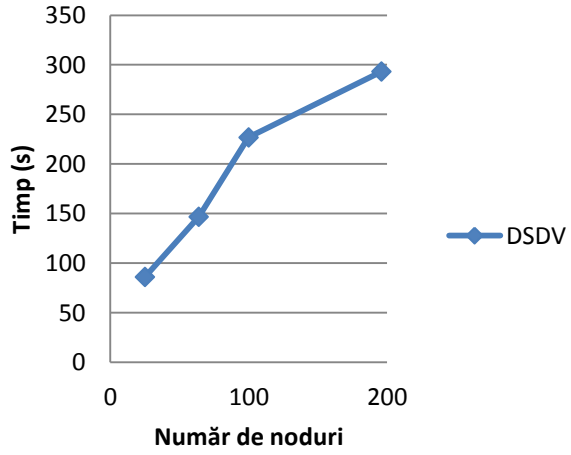


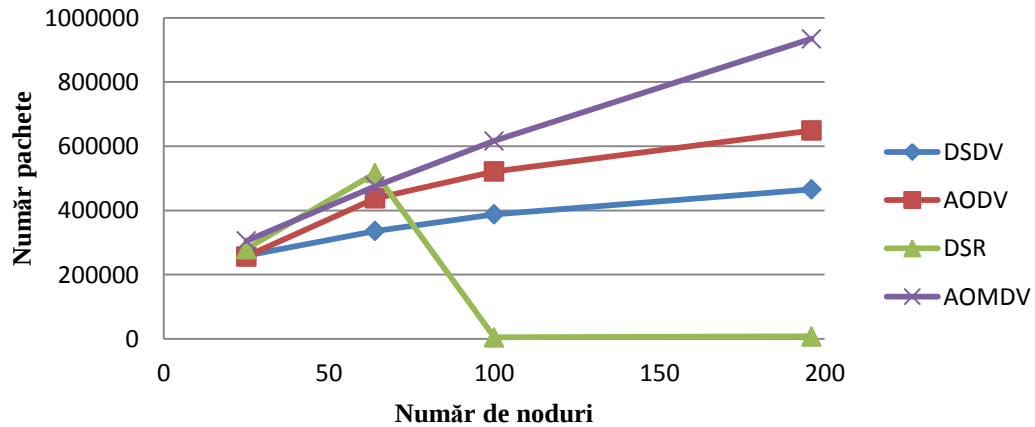
Figura 5.14bTimpul de initializare rute



Din punctul de vedere al numărului total de pachete trimise:

- era de așteptat ca cele mai multe pachete să fie trimise de protocolul AOMDV dat fiind faptul că aceasta încearcă să găsească rute multiple;

Figura 5.15 Număr total de pachete trimise



Din punctul de vedere al procentului de recepție al pachetelor de date:

- cele mai bune rezultate sunt înregistrate de protocolul AOMDV ce este depășit în cazul rețelelor de 25 și 64 de noduri de protocolul DSR, dar acesta pierde întrucât nu mai poate dirija pachete în rețele de dimensiuni mai mari.

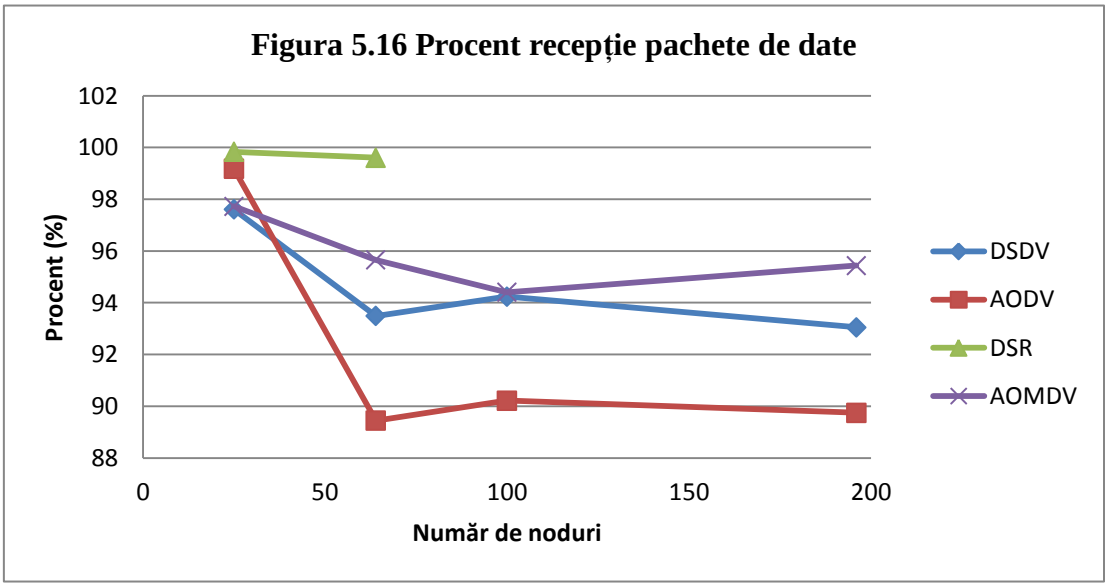


Figura 5.16 Procent recepție pachete de date

Din punctul de vedere al numărului de pachete de date trimise/recepționate

- cel mai bun algoritm este AOMDV ce reușește să trimită cu succes cele mai multe pachete de date indiferent de dimensiunea rețelei. Acesta este ajutat de procentul mare de pachete de date pierdute de protocolul AODV în cazul rețelei de 64 de noduri;
- protocolul AODV este protocolul ce reușește să țină pasul cu AOMDV, dar totdeauna înregistrează rezultate mai slabe din acest punct de vedere.

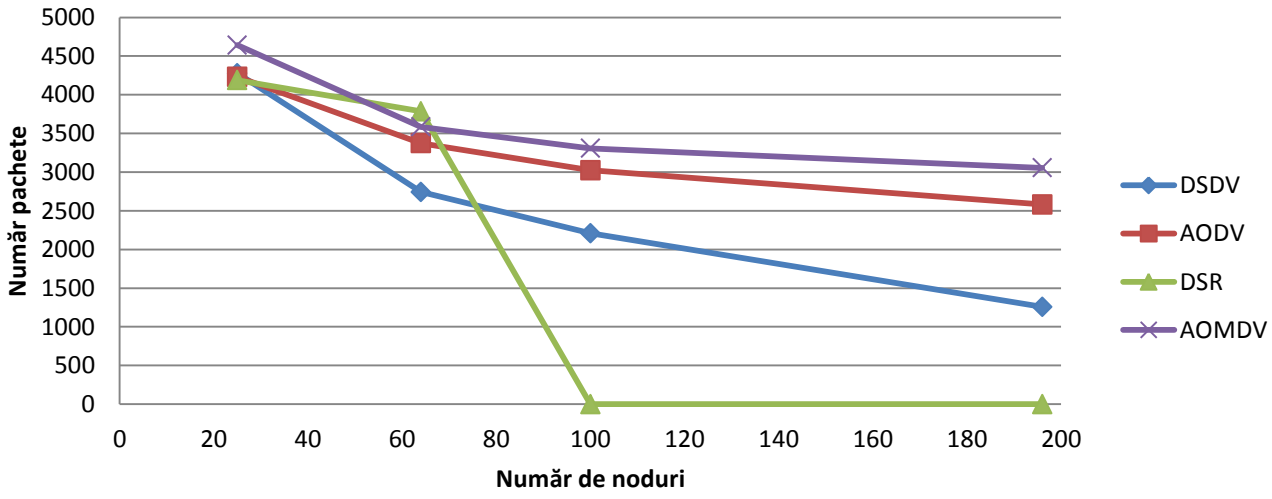
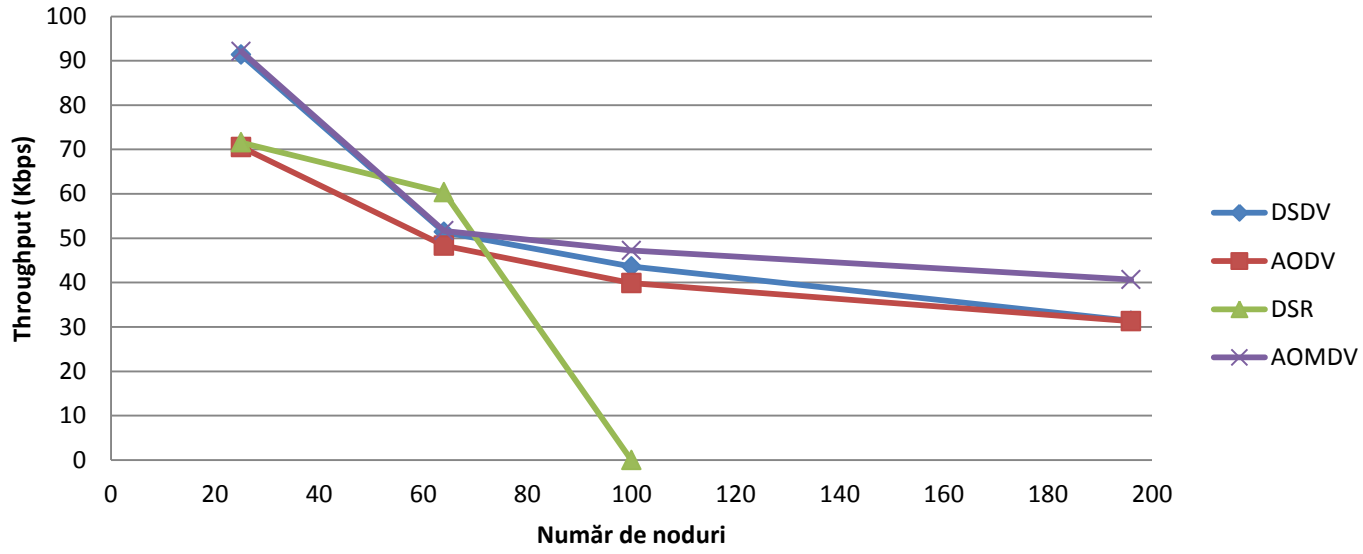


Figura 5.17 Numărul de pachete de date recepționate

Din punctul de vedere al throughput-ului :

- cele mai bune rezultate sunt înregistrate de protocolul AOMDV;
- protocolul DSDV are rezultate asemănătoare sau chiar mai bune decât AOMDV însă acesta înregistrează valori mai scăzute pentru rețele de dimensiuni mai mari.
- cele mai slabe rezultate sunt ale protocolului AODV.

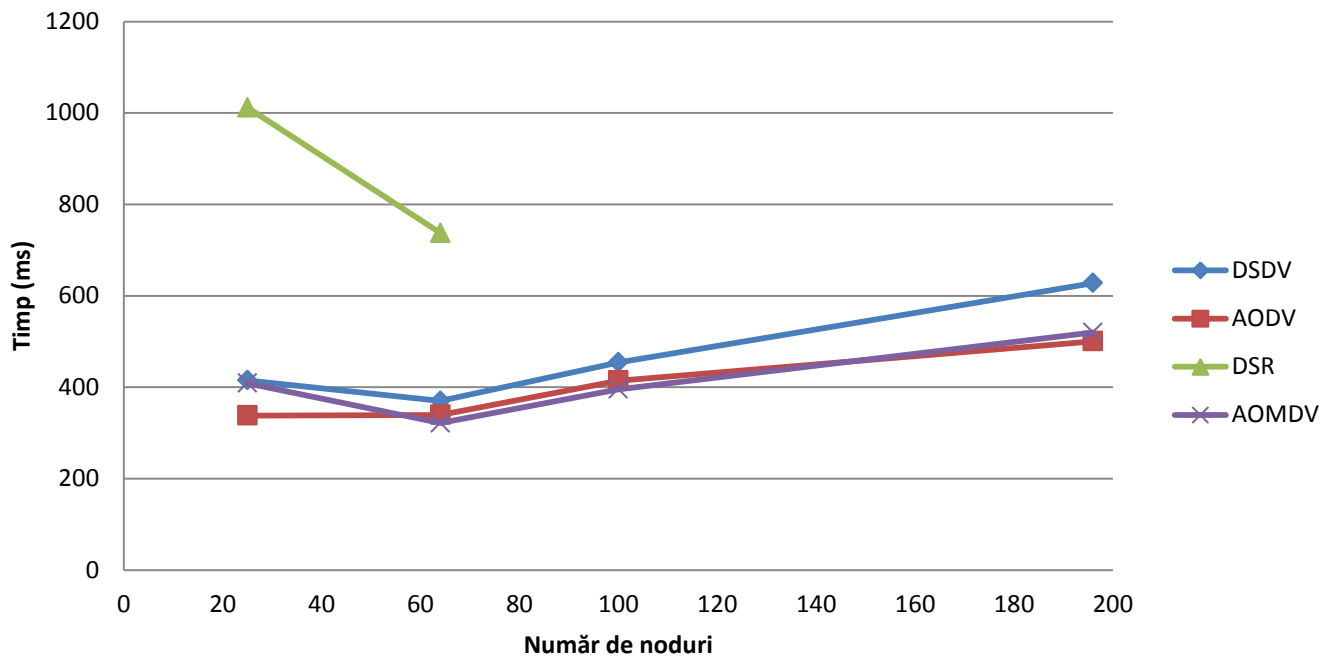
Figura 5.18 Throughput



Din punctul de vedere al întârzierilor capăt-la-capăt:

- cele mai bune rezultate sunt obținute atât de protocolul AODV cât și de protocolul AOMDV . Protocolul AODV obține cele mai bune rezultate pentru rețele foarte mici, care dacă sunt și statice nu e nevoie de căutarea de rute multiple și în cazul rețelelor de dimensiuni foarte mari. Protocolul AOMDV obține cele mai bune rezultate în cazul rețelelor de dimensiuni medii, dar apropiate de protocolul AODV;
- protocolul DSR are cele mai mari întârzieri, iar explicația este timpul îndelungat de procesare. Fiecare nod este nevoit să se adauge pe sine în tabela de rutare, iar acest lucru introduce întârzieri mari.

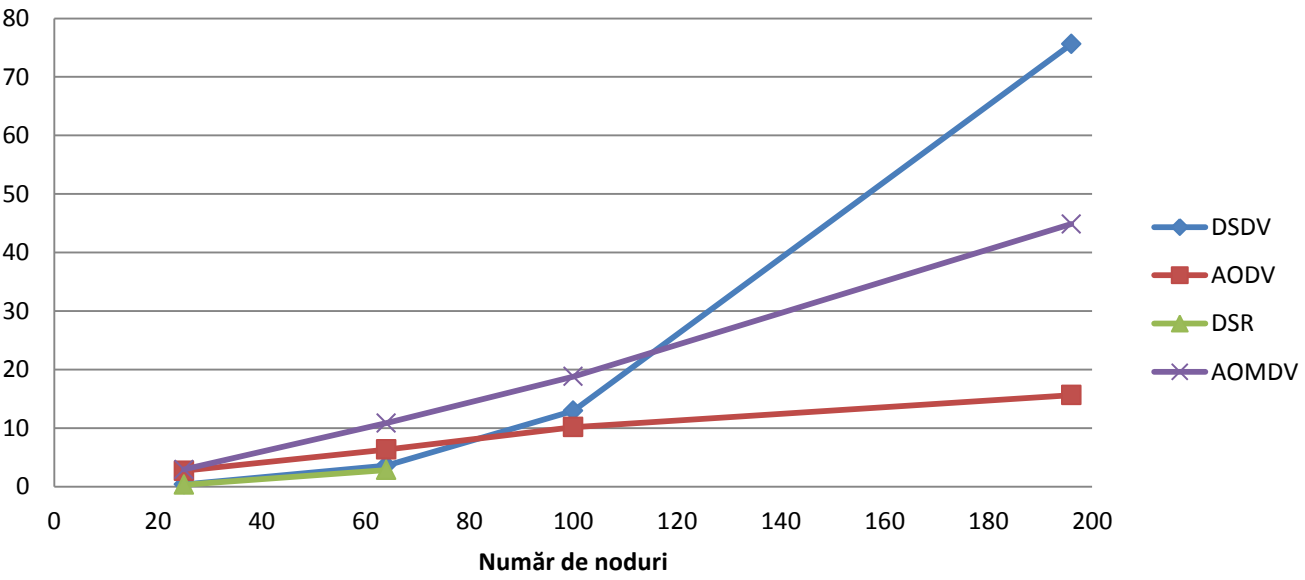
Figura 5.19 Întârzierile capăt la capăt



Din punctul de vedere al încărcării de rutare normalizate:

- Cele mai bune rezultate sunt înregistrate de protocolul DSR, însă acesta nu funcționează pentru rețele de dimensiuni mari ;
- pentru rețele de dimensiuni mari protocolul AODV este cel ce are rezultatele cele mai bune întrucât nici nu folosește pachete de rutare inutil ca în cazul DSDV și nici nu încearcă descoperirea de rute multiple ca în cazul AOMDV.

Figura 5.20 Încărcătura de rutare normalizată



In acest scenariu protocolul de rutare AOMDV obține rezultate foarte bune la mai multe categorii decât protocolul AODV. Protocolul DSR este cel mai slab din punctul de vedere al utilizării în rețele de dimensiuni mari, acesta nu găsește ruta pentru rețele de 100 și 196 de noduri.

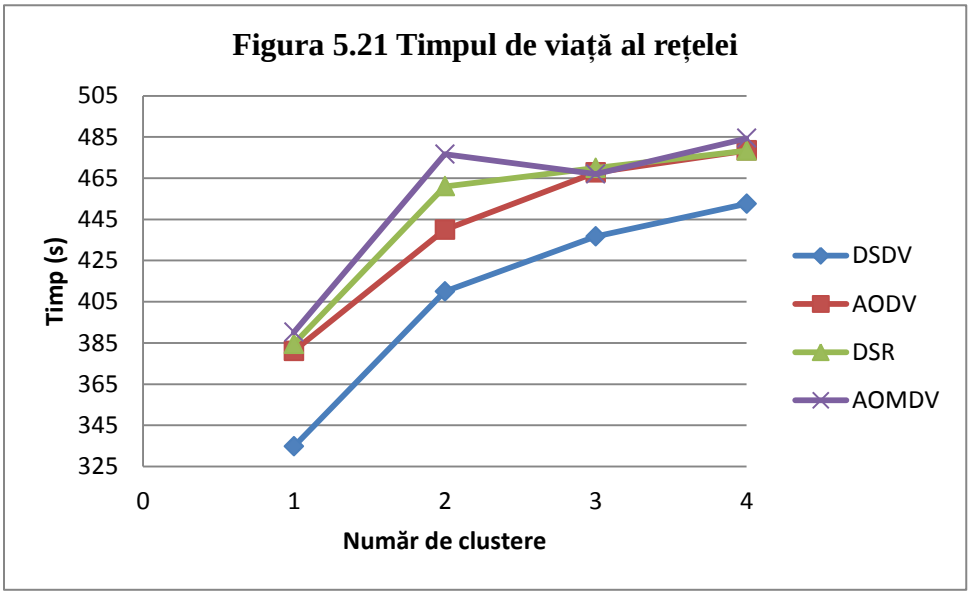
5.5.2. Scenariul cu variația numărului de clustere

Acest scenariu are drept scop testarea performanțelor atunci când există din ce în ce mai multe clustere fiecare având mai multe surse de trafic ce trimit pachete de date către un nod colector aflat în centrul topologiei.

Din punctul de vedere al timpului de viața al rețelei:

- fiind o rețea statică în care în fiecare moment de timp există aceleași surse de trafic și aceleași destinații este clar că identificarea mai multor rute prelungește durata de viață a rețelei. Acest lucru este realizat de AOMDV ce înregistrează și cele mai lungi durate de viață ale rețelelor;
- la polul opus este protocolul DSDV întrucât acesta caută de fiecare dată rute pentru toate destinațiile posibile din rețea și astfel irosește energie.

Figura 5.21 Timpul de viață al rețelei



Din punctul de vedere al timpului de inițializare a rutelor:

- cel mai mare timp de inițializare a rutelor este înregistrat ca și în primul scenariu de protocolul DSDV;
- cel mai bun protocol din acest punct de vedere este AODV care reușește să identifice cea mai bună rută între sursă și destinație, iar protocolul AOMDV înregistrează timpi suficienți de mici chiar într-un caz depășește AODV chiar dacă încearcă să identifice rute multiple.

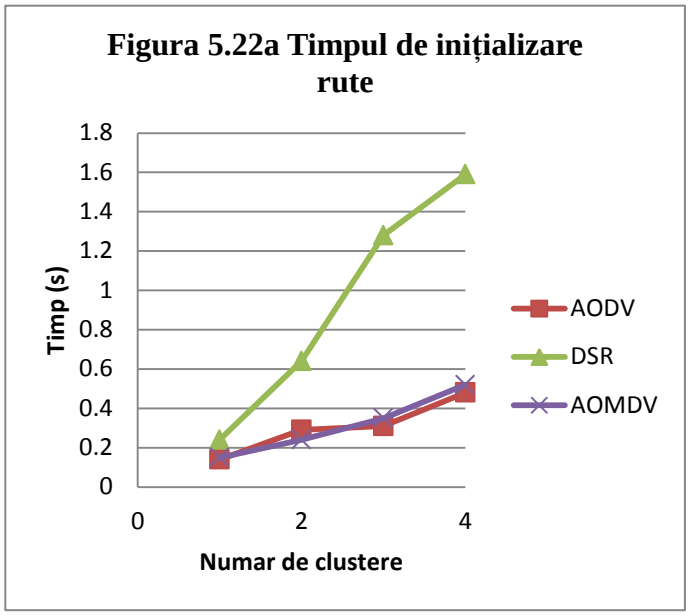


Figura 5.22a Timpul de inițializare rute

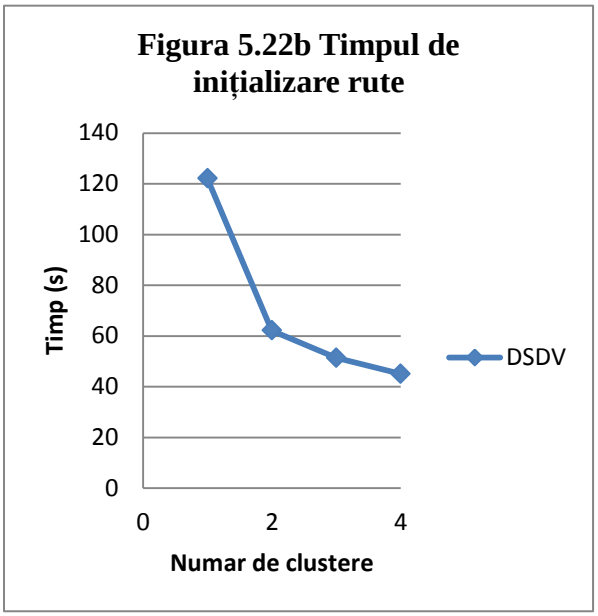


Figura 5.22b Timpul de inițializare rute

Din punctul de vedere al procentului de recepție al pachetelor de date:

- procentul de recepție al pachetelor este ridicat în cazul protocolului DSR atingând cote impresionante 99,12%;
- cel mai mic procent este obținut de protocolul AOMDV, 89,62%.

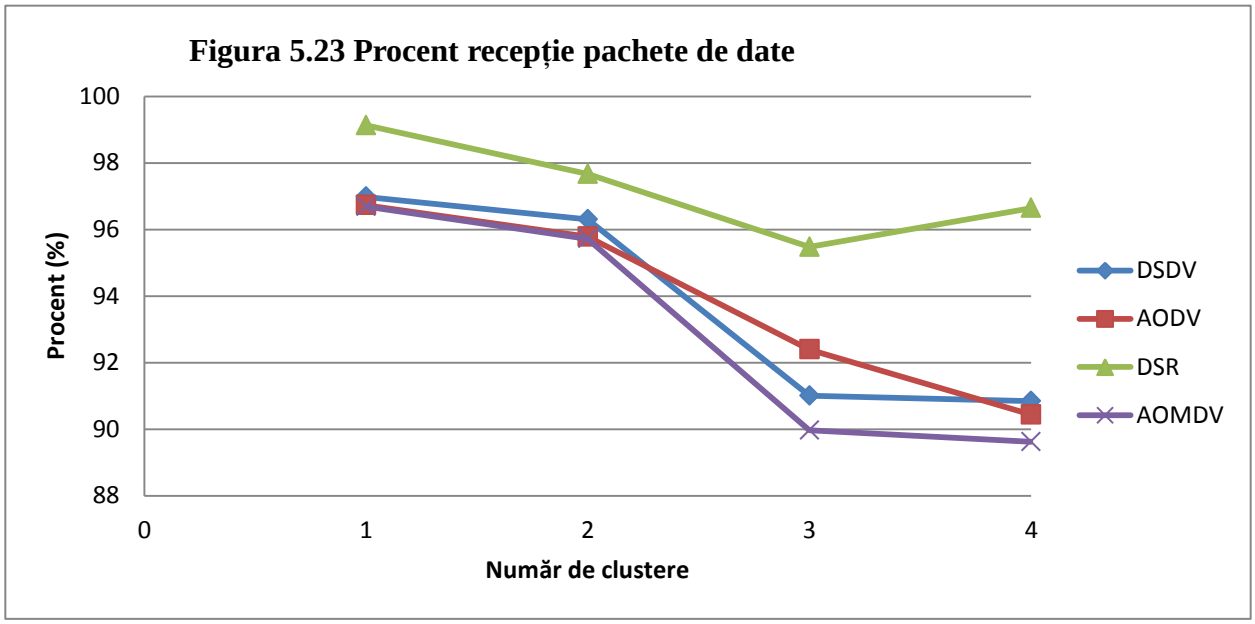


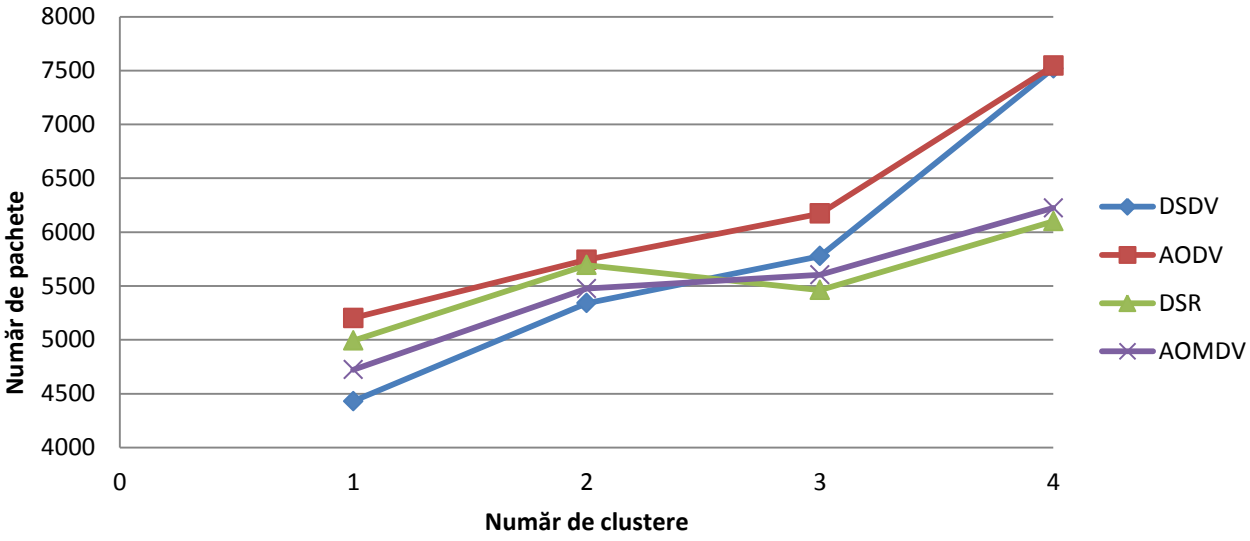
Figura 5.23 Procent recepție pachete de date

Din punctul de vedere al numărului de pachete de date trimise/recepționate

- Protocolul AODV reușește să trimită și să recepționeze cel mai mare număr de pachete de date chiar dacă durata de viață a rețelei în cazul protocolului AODV nu e cea mai mare. Acest lucru se verifică și prin throughput-ul mare pe care îl are rețeaua în cazul folosirii protocolului AODV;

- trebuie remarcat faptul că protocolul DSDV, care pentru rețeaua cu un singur cluster transmite cele mai puține pachete, reușește să atingă numărul de pachete al protocolului AODV pentru rețeaua cu patru clustere.

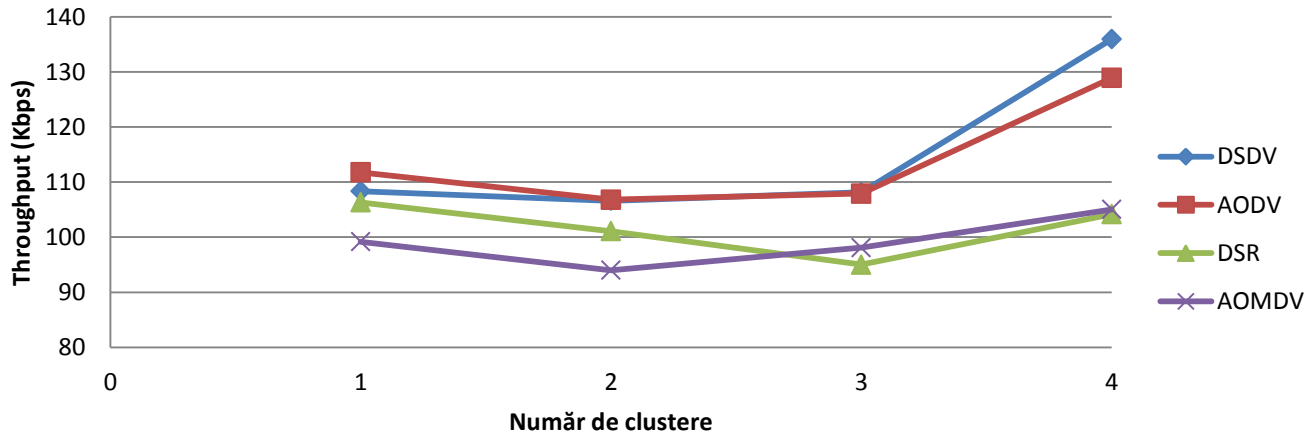
Figura 5.24 Numărul de pachete de date recepționate



Din punctul de vedere al throughput-ului :

- după cum am anticipat, după numărul de pachete de date cel mai bun protocol din acest punct de vedere este protocolul DSDV, iar cu rezultate foarte apropiate este protocolul AODV.
- cea mai scăzută performanță este atinsă de protocolul AOMDV, iar o idee pentru a mări performanță ar fi folosirea căilor multiple de care dispune acest protocol pentru a transmite pachete de date prin toate rutele găsite.

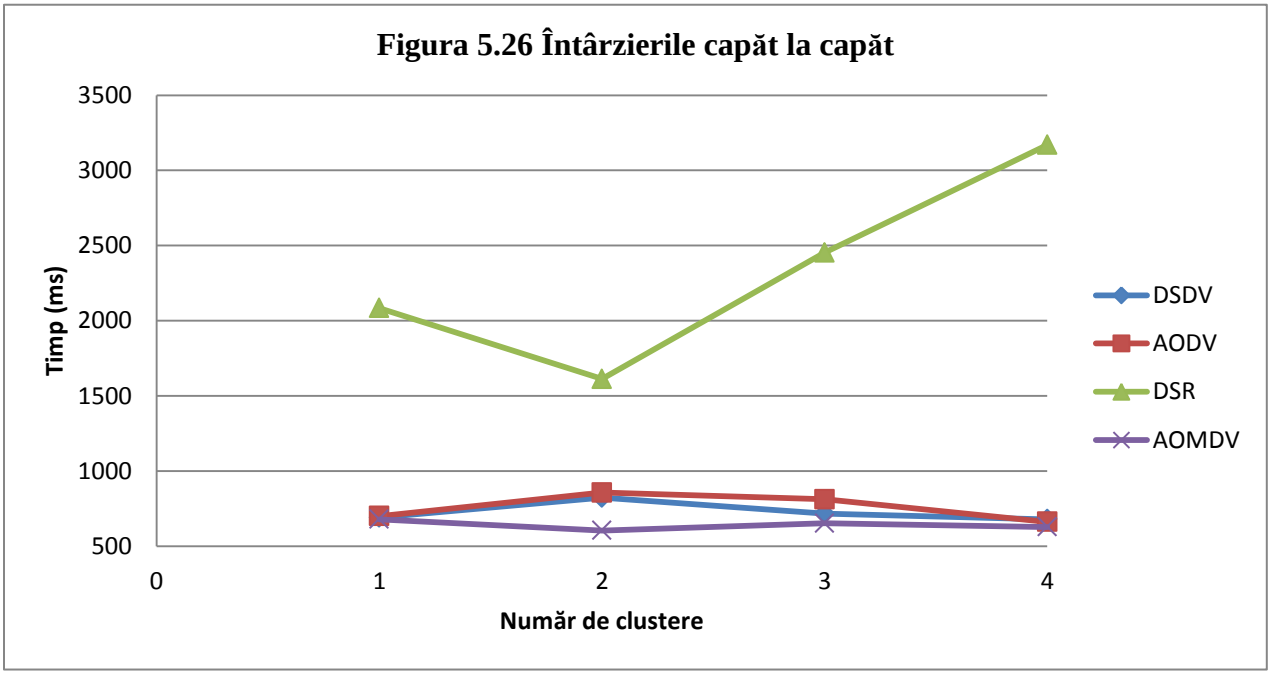
Figura 5.25 Throughput



Din punctul de vedere al întârzierilor capăt-la-capăt:

- protocolul DSR are cele mai mari întârzieri, iar acestea cresc pe măsură ce în rețea sunt introduse mai multe clustere. Explicația este timpul îndelungat de procesare. Fiecare nod este nevoit să se adauge pe sine în tabela de rutare, iar acest lucru introduce întârzieri mari;
- protocolul ce introduce cele mai mici întârzieri este AOMDV obținând în orice situație cele mai bune rezultate.

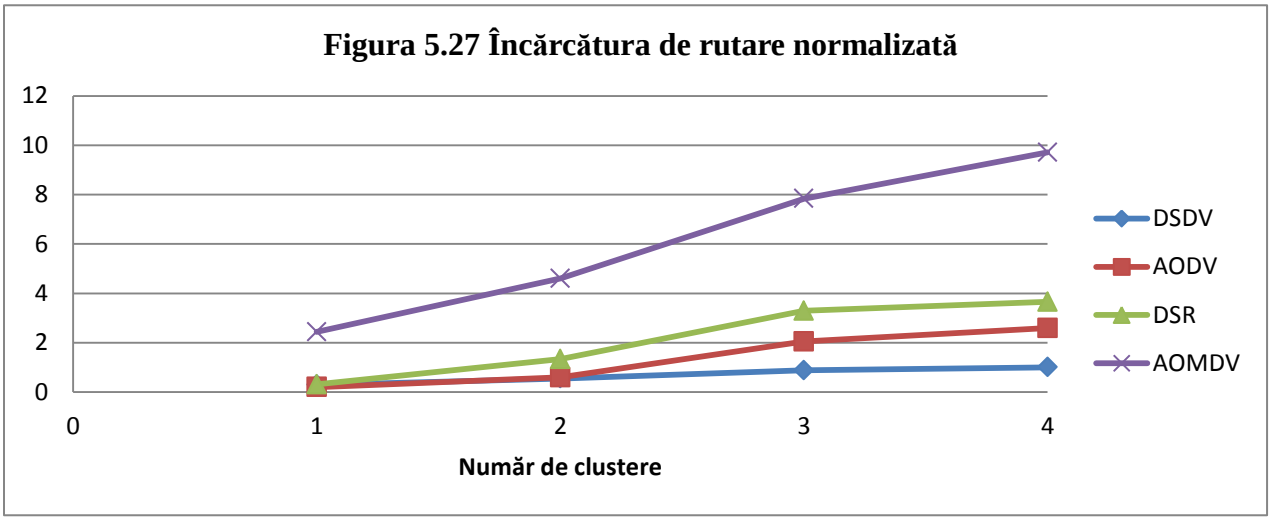
Figura 5.26 Întârzierile capăt la capăt



Din punctul de vedere al încărcării de rutare normalizate:

- în acest caz **cel mai dezavantajos protocol este protocolul AOMDV** ce are o încărcătură de rutare foarte mare spre deosebire de celelalte.
- de remarcat **că cele mai bune rezultate sunt obținute de protocolul DSDV**, dar acest lucru este normal datorită celorlalte rezultate.

Figura 5.27 Încărcătura de rutare normalizată



Concluzia acestui scenariu este că protocolul AOMDV oferă un timp ridicat de existență al rețelei, are timpi de inițializare a rutelor apropiați și în unele cazuri chiar mai buni decât cei ai protocolului AODV, are cele mai mici întârzieri end-to-end și chiar poate oferi performante bune utilizând îmbunătățirea adusă prin folosirea concomitentă a tuturor căilor găsite în materie de throughput. Fiind un scenariu static nici protocolul AODV nu are rezultate pe care să le trecem cu vederea oferind timpi de inițializare a rutelor apropiați cu cei ai protocolului AODV, un număr mare de pachete de date transferate cu un procent de recepție bun, dar și un throughput bun. Protocolul DSDV excelează în materie de încărcarea de rutare normalizată, dar are timpi de inițializare a rutelor foarte mari și o durată mică de viață a rețelei.

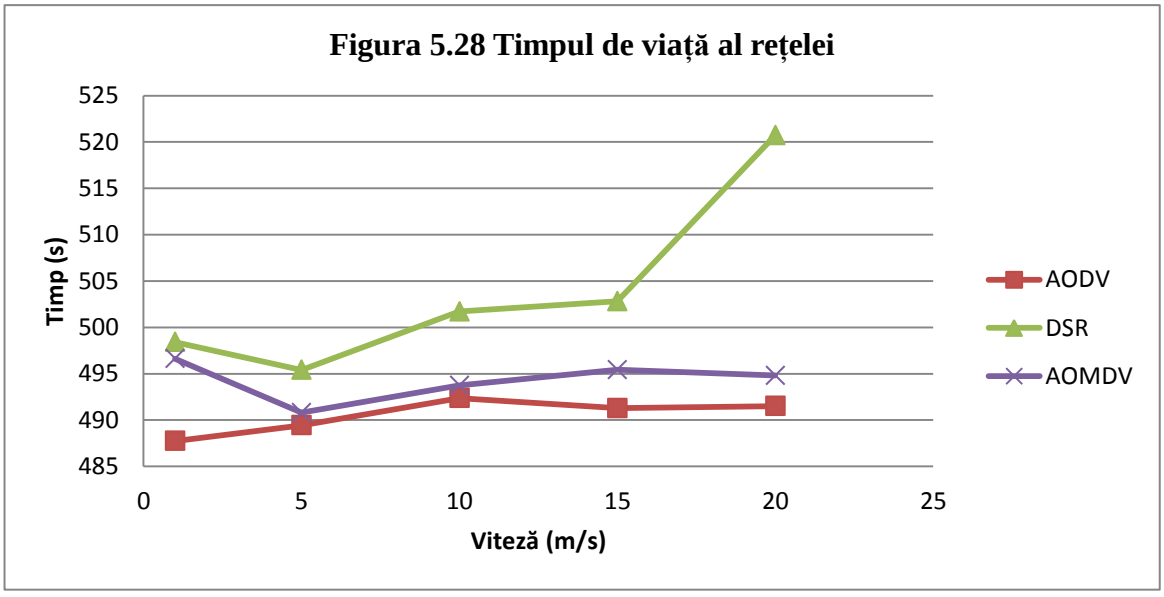
5.5.3. Scenariul cu variația vitezei de mișcare

Acest scenariu a fost proiectat pentru a pune în evidentă comportarea protocoalelor atunci când se creste progresiv viteza de mișcare a nodurilor în rețea. Distanța dintre sursă de trafic și destinație, ca număr de hopuri, poate fi chiar de lungime 1 datorită mișcării aleatoare a nodurilor.

Din punctul de vedere al timpului de viață al rețelei:

- **protocolul DSR își evidențiază calitățile**, confirmând teoria, dar și ideea de la care s-a plecat în proiectarea acestui protocol: construirea unui protocol pentru rețele cu mobilitate crescută cu un număr nu foarte mare de noduri. Totuși analizând rezultatele în continuare acesta nu va mai avea aceeași imagine bună.
- trebuie remarcat că protocolul AOMDV are performanțe apropiate fată de protocolul DSR și mai bune decât protocolul AODV.

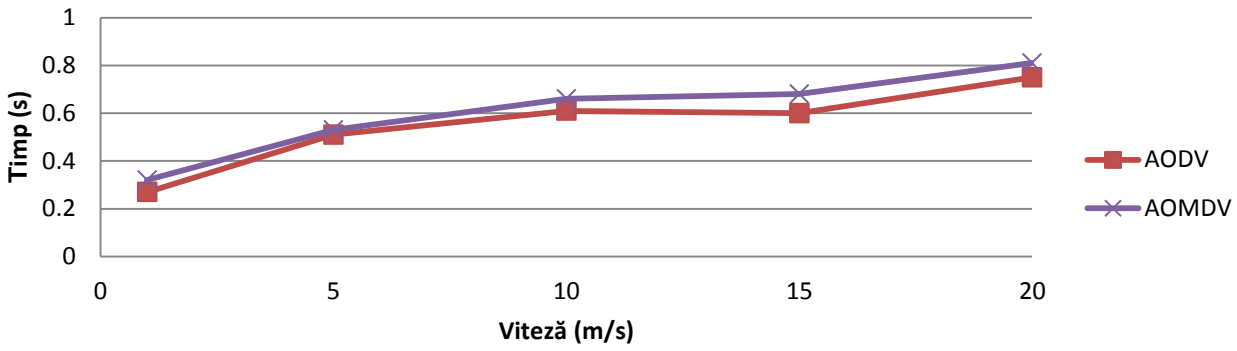
Figura 5.28 Timpul de viață al rețelei



Din punctul de vedere al timpului de inițializare a rutelor:

- protocolul de rutare AODV înregistrează cei mai mici timpi de inițializare a rețelei, iar aceștia cresc de la un caz la altul.
- protocolul AOMDV deși caută rute multiple obține rezultate apropiate de protocolul AODV diferența dintre cele doua nefiind foarte mare.

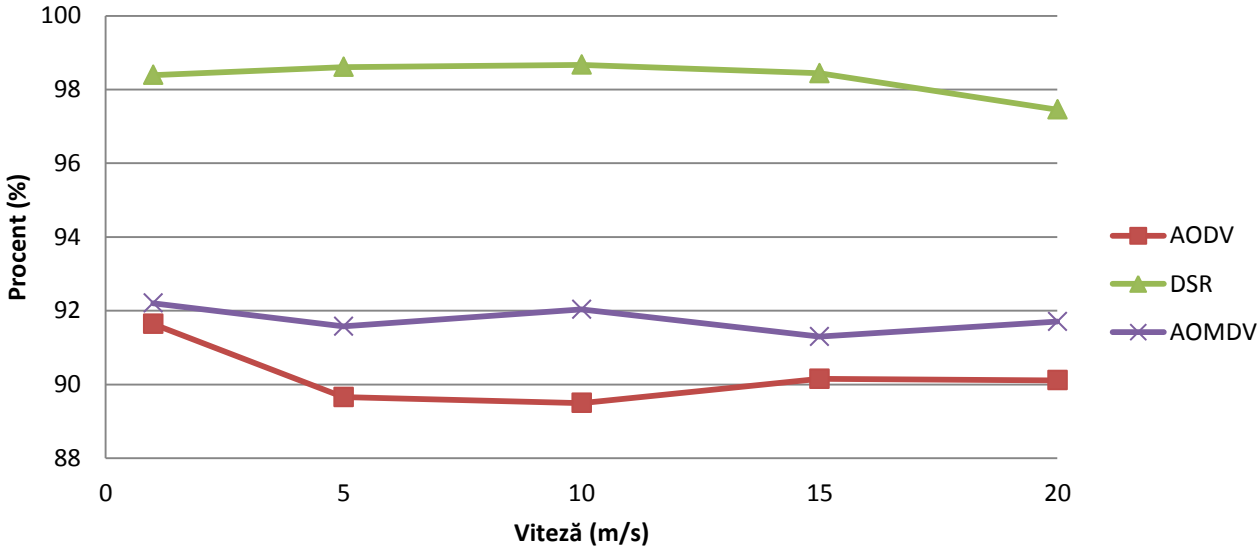
Figura 5.29 Timpul de inițializare rute



Din punctul de vedere al procentului de recepție al pachetelor de date:

- procentul cel mai mare de recepție al pachetelor este obținut de protocolul DSR care obține procente impresionante 98,67 %, dar care e bazat pe principiul „nu muncești nu greșești” întrucât are cel mai mic număr de pachete recepționate care mai este și în scădere pe măsură ce viteza crește.
- protocolul DSR dovedește faptul că este un protocol ce asigură siguranță prin acest procent mare de recepție al pachetelor
- protocolul AOMDV trebuie remarcat pentru procentul său aproape constant situat în jurul valorii de 91,5%, iar acest lucru corelat cu numărul cel mai mare de pachete este un plus.

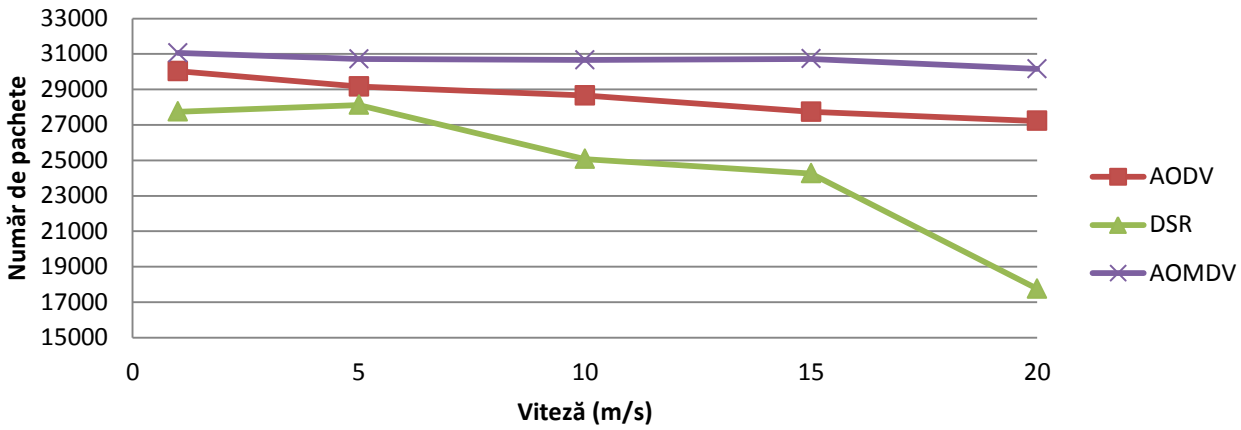
Figura 5.30 Procent recepție pachete de date



Din punctul de vedere al numărului de pachete de date trimise/recepționate

- protocolul AOMDV are cel mai mare număr de pachete recepționate;
- la polul opus se situează protocolul DSR, însă acesta demonstrează că aproape tot din ceea ce transportă ajunge și la destinație.

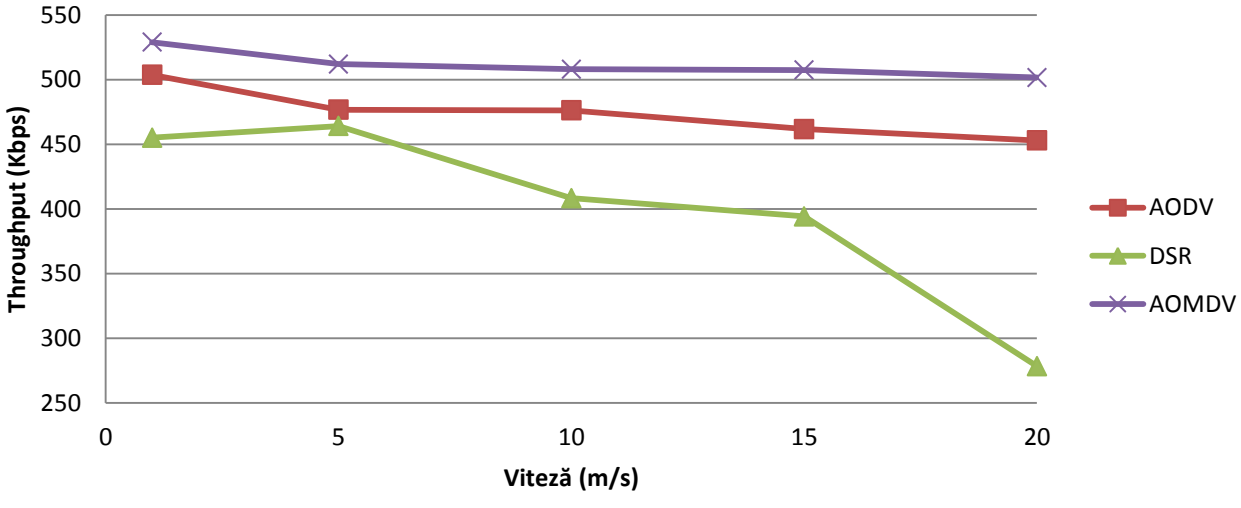
Figura 5.31 Numărul de pachete de date recepționate



Din punctul de vedere al throughput-ului :

- având un număr mare de pachete trimise/recepționate, o durată de viață mare a rețelei este clar faptul că și la acest capitol protocolul AOMDV este cel ce are cele mai bune rezultate.
- la polul opus se află protocolul DSR care deși are o durată de viață mare și un procent mare de recepție, are un număr mic de pachete trimise și cel mai mic throughput.

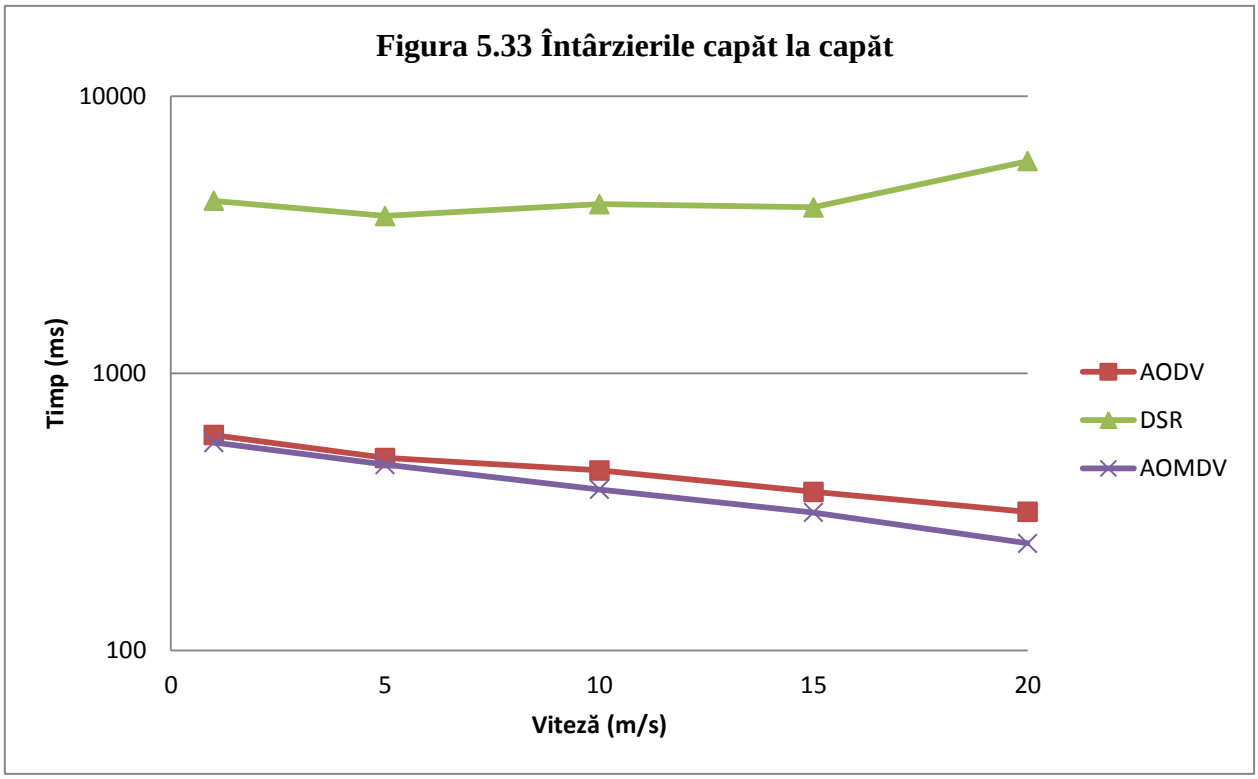
Figura 5.32 Throughput



Din punctul de vedere al întârzierilor capăt-la-capăt:

- ca și în celelalte scenarii cele mai mari întârzieri sunt produse de protocolul DSR, iar acestea cresc pe măsură ce viteza nodurilor în rețea creste. Explicația este timpul îndelungat de procesare. Fiecare nod este nevoit să se adauge pe sine în tabela de rutare, iar acest lucru introduce întârzieri mari;
- cele mai mici întârzieri sunt produse de protocolul AOMDV și astfel devine cel mai bun protocol din acest punct de vedere.

Figura 5.33 Întârzierile capăt la capăt



Din punctul de vedere al încărcării de rutare normalizate:

- cele mai bune rezultate pentru acest capitol sunt înregistrate de protocolul DSR, dar numai pentru viteze de până la 10 m/s. Pentru celelalte cele mai bune rezultate sunt obținute, pe rând, de AODV și AOMDV.

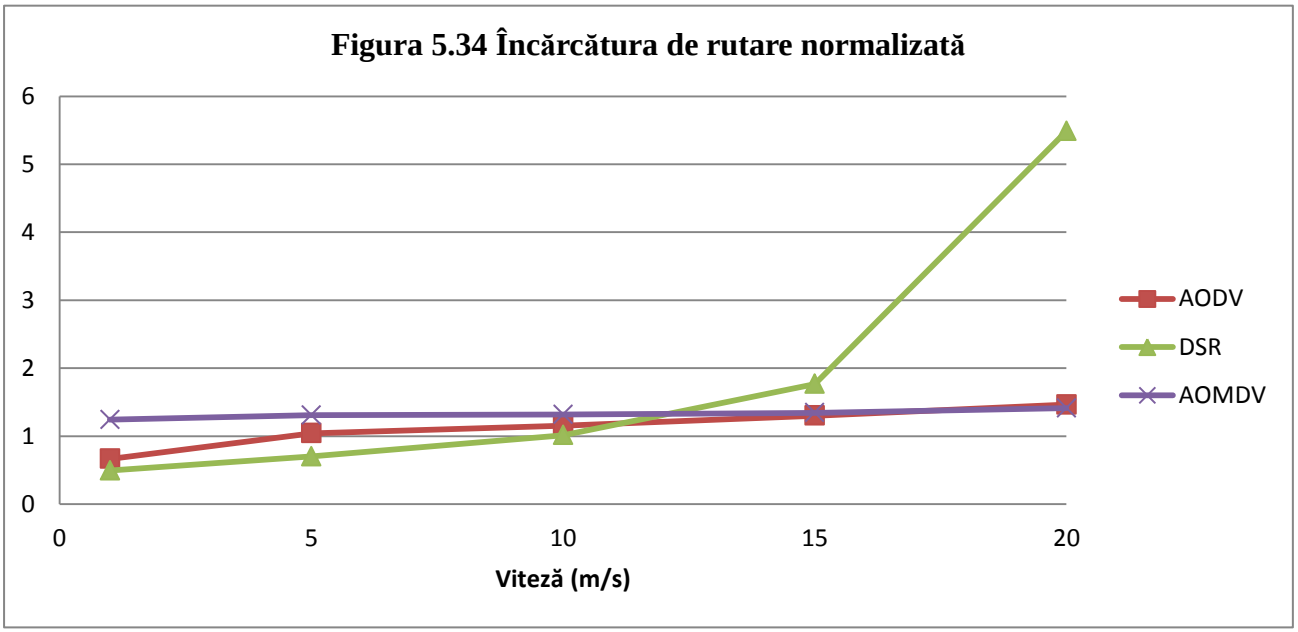


Figura 5.34 Încărcătura de rutare normalizată

Deși protocolul DSR are unele rezultate bune cum ar fi timpul de viață, un procent mare de recepție al pachetelor restul rezultatelor fac ca acesta să fie cel mai slab protocol din acest scenariu.

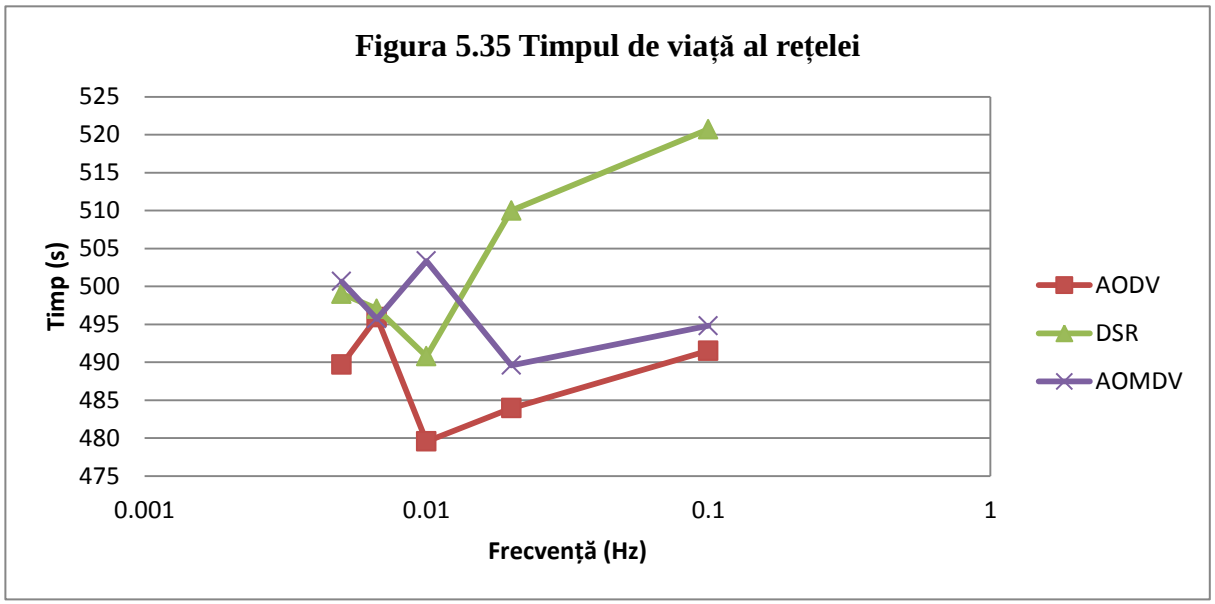
Protocolul AODVM asigură cel mai mare throughput, un număr mare de pachete recepționate cu un procent bun de recepție, cele mai mici întârzieri. Pe lângă acestea asigură o durată bună de viață a rețelei prin utilizarea rutelor multiple, însă o încărcătură de rutare mai mare pentru viteze mici lucru decontat prin comportarea bună la viteze mari.

5.5.4. Scenariul cu variația frecvenței de mișcare

Acest scenariu a fost gândit pentru a evidenția comportamentul protocoalelor de rutare în cazul în care nodurile își modifică coordonatele cu o frecvență ce este variată treptat de la 1/200 Hz până la 0.1 Hz.

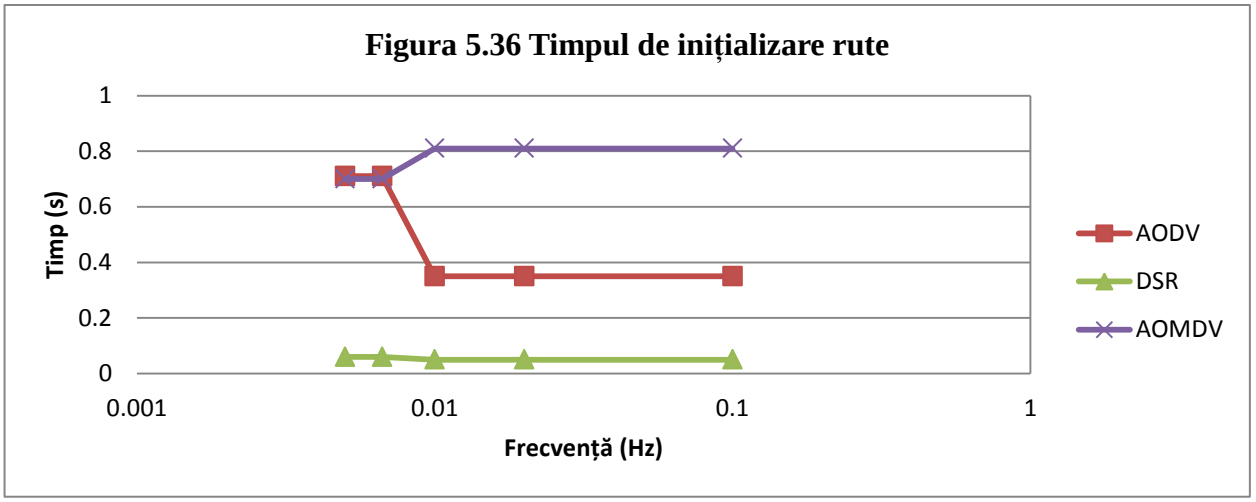
Din punctul de vedere al timpului de viață al rețelei:

- protocolul AOMDV obține cele mai bune rezultate pentru primele trei cazuri, iar în ultimele 2 protocolul DSR îl devansează însă nu din cauză că asigură performanțe mai mari ci tocmai din cauza neperformantei ce îl face să consume mai putină energie.
- cel mai slab protocol este AODV ce reușește doar în cazul al doilea să obțină rezultatul cel mai bun, însă în rest are rezultatele cele mai mici.



Din punctul de vedere al timpului de inițializare a rutelor:

- protocolul DSR obține cei mai mici timpi de inițializare a rutelor, dar acesta pare să fie pentru el un proces continuu întrucât în urma rezultatelor următoare vom vedea că defapt este un protocol leneș.
- protocolul AOMDV obține un rezultat așteptat din acest punct de vedere având cel mai mare timp de inițializare a rutei datorită căilor sale multiple pe care încearcă să și le formeze;



Din punctul de vedere al procentului de recepție al pachetelor de date:

- protocolul DSR este cel ce obține cele mai bune rezultate având chiar și 98,15% însă dacă corelăm această informație cu faptul că numărul de pachete recepționate este cel mai mic nu mai punem acest protocol într-o lumină la fel de buna. Totuși ceea ce se poate afirma este faptul că protocolul DSR este unul sigur, aproape toate pachetele trimise fiind și recepționate.

- protocolul ce se clasează pe poziția următoare este protocolul AOMDV ce are și un număr mare de pachete recepționate.
- cel mai slab protocol este AODV ce înregistrează cele mai scăzute procente de recepție a pachetelor.

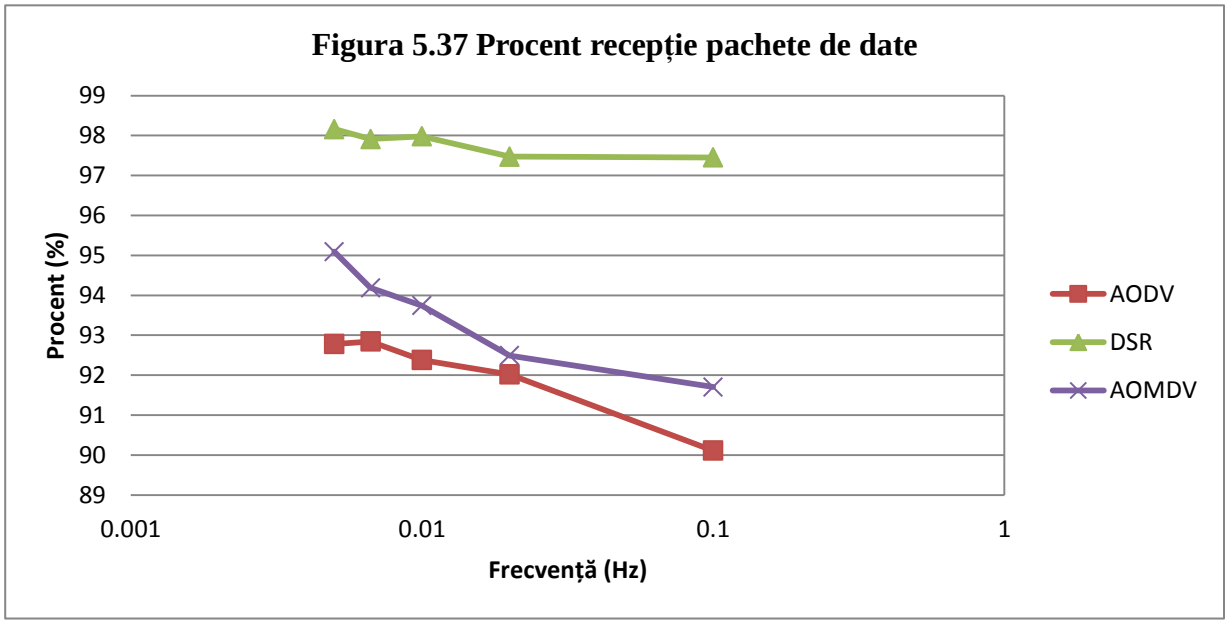


Figura 5.37 Procent recepție pachete de date

Din punctul de vedere al numărului de pachete de date trimise/recepționate

- cum am spus și mai sus protocolul AOMDV este cel ce are cele mai multe pachete recepționate, dar și cele mai multe pachete trimise ceea ce îl face să fie cel mai bun la acest domeniu.
- la polul opus se situează protocolul DSR, însă acesta demonstrează că aproape tot din ceea ce transportă ajunge la destinație.

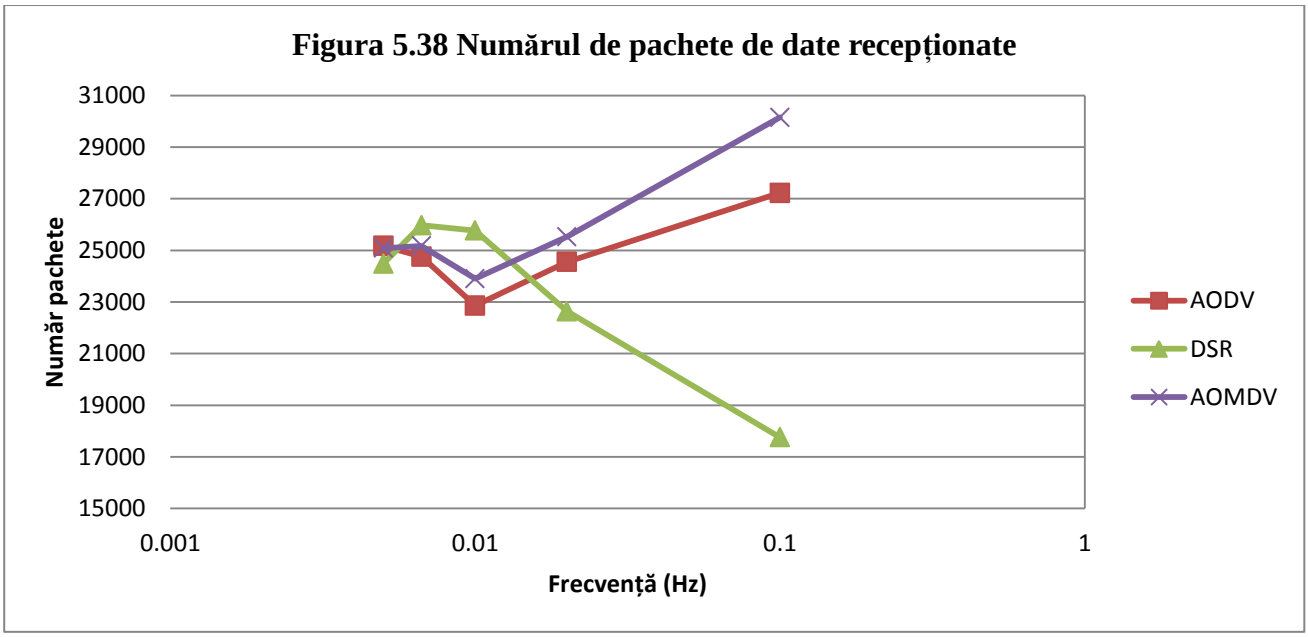
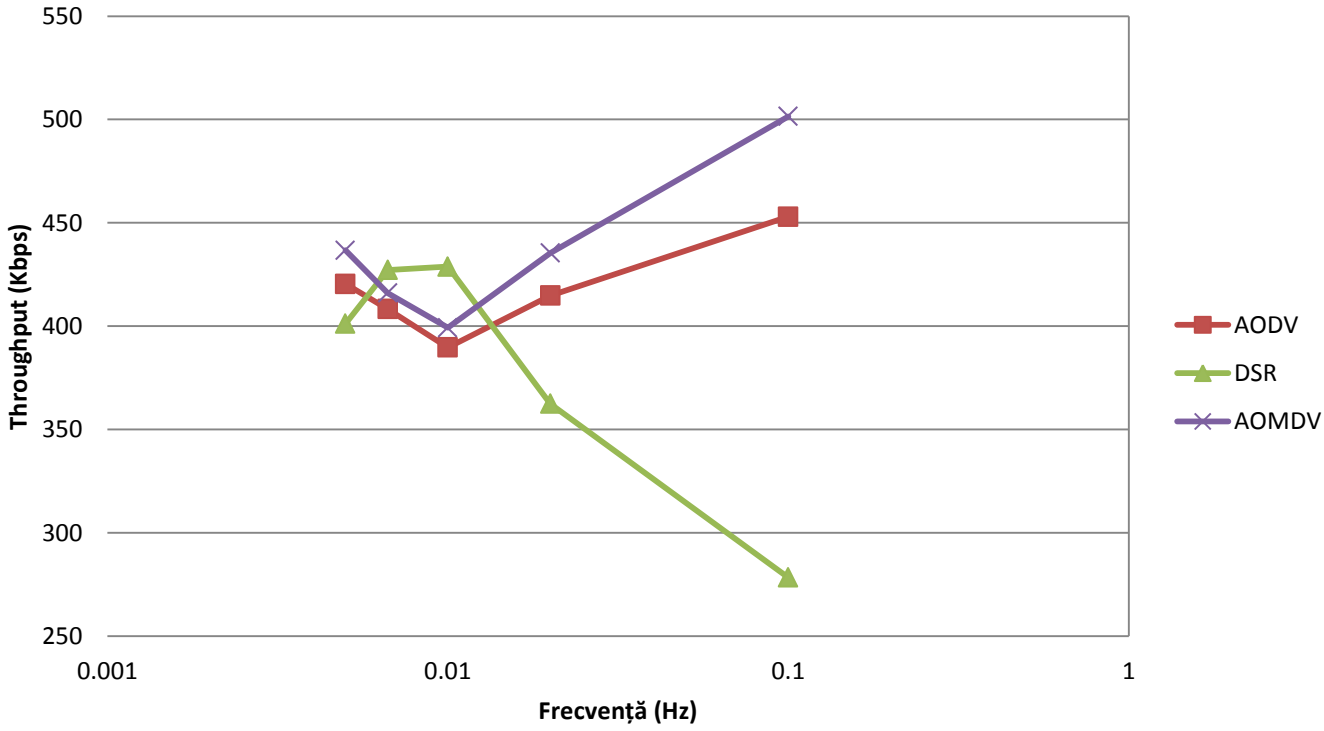


Figura 5.38 Numărul de pachete de date recepționate

Din punctul de vedere al throughput-ului :

- din punctul acesta de vedere protocolul AOMDV este cel ce are cele mai bune rezultate în 3 din 5 cazuri, iar în celelalte 2 fiind al doilea;
- cel mai slab protocol este DSR care deși obține cele mai bune rezultate pentru primele două cazuri, în rest obține rezultate cu mult mai proaste decât a celorlalte protocoale și care scad pe măsură ce frecvența mișcărilor crește.

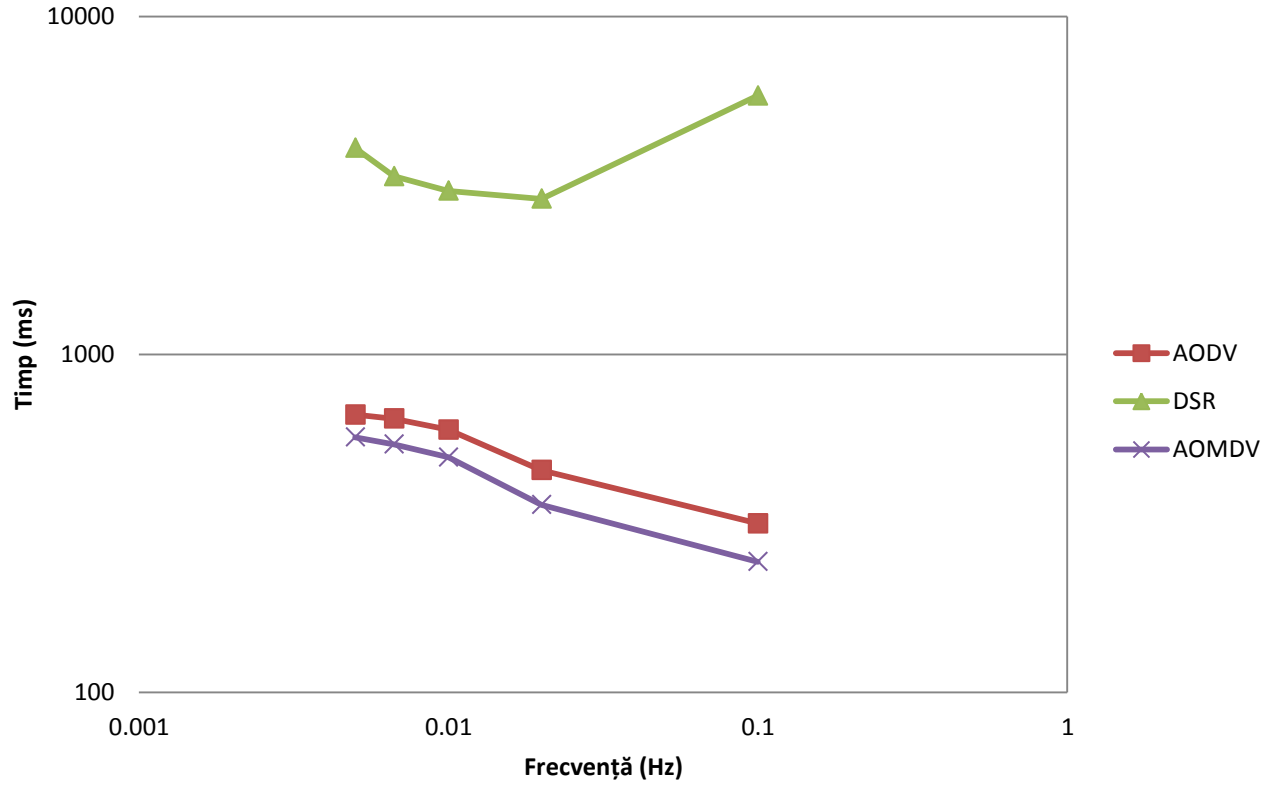
Figura 5.39 Throughput



Din punctul de vedere al întârzierilor capăt-la-capăt:

- ca în toate celelalte cazuri protocolul DSR este cel ce are cele mai mari întârzieri, cu mult mai mari decât cele ale celorlalte protocoale.
- cele mai bune rezultate sunt obținute de protocolul AOMDV ce înregistrează cele mai mici întârzieri end-to-end în fiecare caz și devansează protocolul AODV.

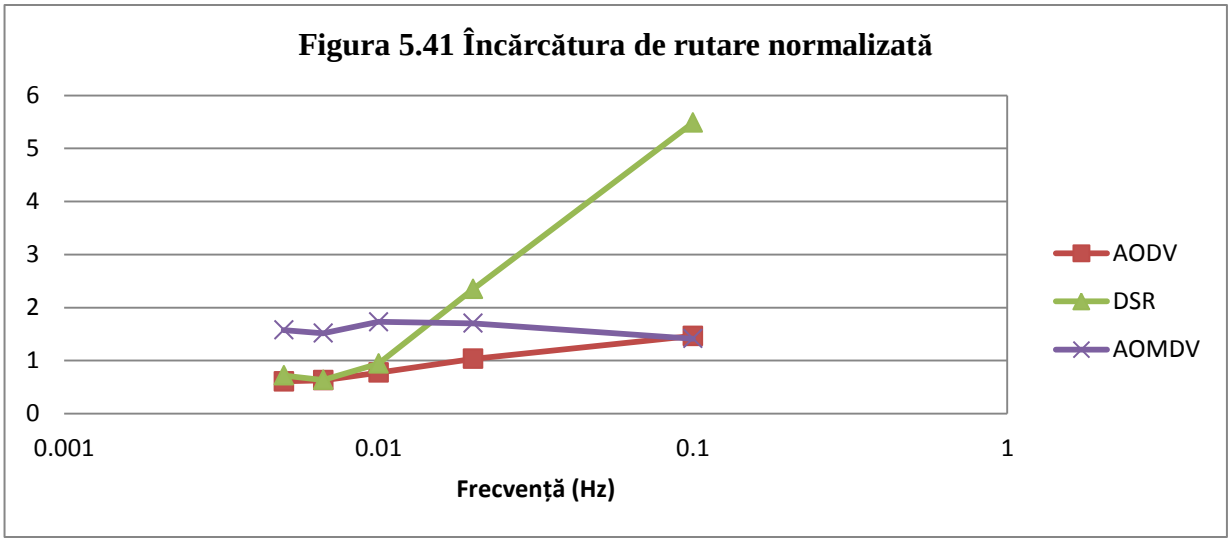
Figura 5.40 Întârzierile capăt la capăt



Din punctul de vedere al încărcării de rutare normalizate:

- protocolul DSR este cel ce înregistrează valori cu mult mai mari ale încărcăturii normalizate pentru frecvențe mari de deplasare față de celelalte;
- dacă este vorba de frecvențe mici de mișcare protocolul AODV are cele mai bune rezultate însă trebuie remarcat că acestea cresc, iar pentru frecvențe mari de mișcare cel mai recomandat este protocolul AOMDV.

Figura 5.41 Încărcătura de rutare normalizată



Protocolul DSR are rezultate foarte bune în ceea ce privește timpul de viață al rețelei și timpul de inițializare al rețelei, însă acestea nu sunt susținute de numărul mic de pachete trimise/recepționate, throughput, întârzierile end-to-end, încărcătura de rutare normalizată și astfel acesta poate fi notat ca fiind protocolul cel mai slab din acest scenariu. Trebuie totuși remarcat că protocolul DSR este totuși cel mai sigur protocol având cel mai mare procent de recepție al pachetelor.

Protocolul AODV are rezultate bune în ceea ce privește timpul de inițializare a rutelor, încărcătura de rutare normalizată, însă procentele mici de recepție al pachetelor, numărul mic de pachete trimise/recepționate, timpul mic de viața al rețelei fac ca acesta să nu exceleze.

Protocolul AOMDV prezintă rezultatele cele mai bune chiar dacă nu la toate capitolele. Are un timp de viață al rețelei bun, susținut de cel mai mare număr de pachete trimise/recepționate și de un procent de recepție bun, de un throughput ce are valorile cele mai mari pentru acest protocol și de întârzierile cele mai mici. Ca puncte slabe ale acestui protocol putem nota încărcătura de rutare normalizată și timpul de inițializare a rutelor mare aceste rezultate fiind obținute tocmai din motivele pentru care are ceilalți parametri buni.

Durata de viață a rețelei este influențată, în studiul pe care l-am făcut, de eficiența fiecărui protocol în gestionarea energiei nodurilor. Acest aspect este unul foarte important în rețelele de senzori wireless. Majoritatea implementărilor implică utilizarea bateriilor ca sursă de alimentare pentru nodurile acestor rețele. În multe aplicații aceste baterii ori sunt foarte dificil de înlocuit, ori nodurile sunt gândite a fi de unică folosință. Varianta optimizată a protocolului AODV, AOMDV obține rezultate bune din acest punct de vedere, dar nu excelează. Modul de funcționare al acestui protocol, prin căutarea de rute multiple, face ca uneori energia nodurilor să nu fie gestionată bine. Soluția ar putea veni prin uniformizarea consumului energetic al nodurilor rețelei.

Să presupunem că există mai multe fluxuri de date. Pentru unul din acestea ruta compusă din nodurile x, y și z este unică pentru a ajunge la destinație. Pentru celelalte fluxuri există rute alternative, mai ocolitoare, dar care pot conduce cu succes datele. Faptul că protocolul studiat dirijează toate pachetele prin ruta compusă din nodurile x, y și z face ca acestea să piardă rapid energie și să devină neutilizabilă. În acest caz fluxul de date ce avea singura posibilitate de a ajunge la destinație prin intermediul acestei rute va fi întrerupt. Dacă protocolul de rutare ar cunoaște și ar ține cont de aceste aspecte ar putea dirija pachetele ce au rute alternative prin alte căi fără să consume toată energia unei rute. Protocolul AOMDV ce oricum identifică mai multe rute pentru un flux de date va obține o creștere a performantelor. Verificarea faptului că pentru un flux de date există doar o rută se poate face prin numărarea rutelor din tabela de rutare. În acest mod s-ar obține o creștere a duratei de viață a rețelei, un throughput mai mare fără creșterea încărcăturii de rutare normalizate.

O altă metodă prin care se poate uniformiza traficul este de a trimite pachete prin diversele rute găsite de protocolul nostru ținând cont de nivelul energetic al nodurilor ce o compun. Totuși această variantă ar necesita un schimb continuu de informație cu privire la nivelul energetic și ar supraîncărca rețeaua.

O altă idee de îmbunătățire ar fi adăugarea la pachetele de date a unui câmp care să precizeze nivelul de prioritate al fluxului respectiv. Astfel o informație ce ar trebui să ajungă foarte repede la destinație ar fi împărțită și trimisă prin mai multe căi de care dispune protocolul AOMDV. Prin această

tehnică s-ar micșora întârzierile capăt la capăt și ar creste throughpu-ul fără modificarea încărcăturii normalizate de rutare.

5.5.5. Sinteza rezultate experimentale și recomandări de proiectare

In această secțiune am realizat un sumar al rezultatelor simulărilor pentru a evidenția ce protocol este bine pentru a fi utilizat în funcție de aspectul ce ne interesează. Pentru fiecare scenariu am definit și am calculat un punctaj în funcție de locurile pe care le ocupă protocoalele din diverse puncte de vedere. Aceste punctaje este bine să fie cât mai mici pentru a putea spune că un protocol are un comportament bun într-un anumit scenariu. Se recomandă utilizarea unui anumit protocol atunci când ocupă locul 1 dintr-un anumit punct de vedere sau are un punctaj cât mai mic într-un anumit tip de rețea, marcat cu verde. Punctajele marcate cu verde reprezintă rezultate foarte bune, galben reprezintă rezultate bune, portocaliu reprezintă rezultate slabe, iar roșu rezultate foarte slabe.

Tabel 5.23 Schema de alegere a protocolului optim pentru scenariul cu variația vitezei de mișcare

Scenariul cu variația vitezei de mișcare	AODV	DSR	AOMDV
durata de viață	locul 3	locul 1	locul 2
timp de inițializare	locul 1	locul 3	locul 2
procent recepție	locul 3	locul 1	locul 2
număr de pachete recepționate	locul 2	locul 3	locul 1
throughput	locul 2	locul 3	locul 1
întârzieri capăt la capăt	locul 2	locul 3	locul 1
încărcătura de rutare normalizată	locul 2 pentru viteze de 1, 5, 10 și 20 m/s locul 1 pentru 15 m/s	locul 1 pentru 1, 5 și 10 m/s locul 3 pentru 15 și 20 m/s	locul 3 pentru viteze de 1, 5, 10 și 15 m/s locul 1 pentru 20 m/s
Punctaj viteză de 1 m/s	15	15	12
Punctaj viteză de 5 m/s	15	15	12
Punctaj viteză de 10 m/s	15	15	12
Punctaj viteză de 15 m/s	14	17	12
Punctaj viteză de 20 m/s	15	17	10
Punctaj total protocoale per scenariu	74	79	58

Tabel 5.24 Schema de alegere a protocolului optim pentru scenariul cu variația frecvenței de mișcare

Scenariul cu variația frecvenței de mișcare	AODV	DSR	AOMDV
durata de viață	locul 3	locul 2 pentru 1/200, 1/100 Hz locul 1 pentru 1/150, 1/50 și 1/10 Hz	locul 1 pentru 1/200 și 1/100 Hz locul 2 pentru 1/150, 1/50 și 1/10 Hz
timp de inițializare	locul 3 pentru 1/200 și 1/150 Hz locul 2 pentru 1/100, 1/50 și 1/10 Hz	locul 1	locul 2 pentru 1/200 și 1/150 Hz locul 3 pentru 1/100, 1/50 și 1/10 Hz
procent recepție	locul 3	locul 1	locul 2
număr de pachete recepționate	locul 1 pentru 1/200 Hz locul 3 pentru 1/150 și 1/100 Hz locul 2 pentru 1/50 și 1/10 Hz	locul 3 pentru 1/200 Hz, 1/50 și 1/10 Hz locul 1 pentru 1/150 și 1/100 Hz	locul 2 pentru 1/200, 1/150 și 1/100 Hz locul 1 pentru 1/50 și 1/10 Hz
throughput	locul 3 pentru 1/150 și 1/100 Hz locul 2 pentru 1/200, 1/50 și 1/10 Hz	locul 3 pentru 1/200, 1/50 și 1/10 Hz locul 1 pentru 1/150 și 1/100 Hz	locul 2 pentru 1/150 și 1/100 Hz locul 1 pentru 1/200, 1/50 și 1/10 Hz
întârzieri capăt la capăt	locul 2	locul 3	locul 1
încărcătura de rutare normalizată	locul 1 pentru 1/200, 1/150, 1/100 și 1/50 Hz locul 2 pentru 1/10 Hz	locul 2 pentru 1/200, 1/150 și 1/100 Hz locul 3 pentru 1/50 și 1/10 Hz	locul 3 pentru 1/200, 1/150 și 1/100 Hz locul 2 pentru 1/50 Hz locul 1 pentru 1/10 Hz
Punctaj frecvență de 1/200 Hz	15	15	12
Punctaj frecvență de 1/150 Hz	18	10	14
Punctaj frecvență de 1/100 Hz	17	11	14
Punctaj frecvență de 1/50 Hz	15	15	12
Punctaj frecvență de 1/10 Hz	16	15	11
Punctaj total protocoale per scenariu	81	66	63

Tabel 5.25 Schema de alegere a protocolului optim pentru scenariul cu variația numărului de clustere

Scenariul cu variația numărului de clustere	DSDV	AODV	DSR	AOMDV
durata de viață	locul 4	locul 3	locul 2	locul 1
timp de inițializare	locul 4	locul 2 pentru 1 și 2 clustere locul 1 pentru 3 și 4 clustere	locul 3	locul 1 pentru 1 și 2 clustere locul 2 pentru 3 și 4 clustere
procent recepție	locul 2 pentru 1, 2 și 4 clustere locul 3 pentru 3 clustere	locul 3 pentru 1, 2 și 4 clustere locul 2 pentru 3 clustere	locul 1	locul 4
număr de pachete recepționate	locul 4 pentru 1 și 2 clustere locul 2 pentru 3 și 4 clustere	locul 1	locul 2 pentru 1 și 2 clustere locul 4 pentru 3 și 4 clustere	locul 3
throughput	locul 2 pentru 1 și 2 clustere locul 1 pentru 3 și 4 clustere	locul 1 pentru 1 și 2 clustere locul 2 pentru 3 și 4 clustere	locul 3 pentru 1 și 2 clustere locul 4 pentru 3 și 4 clustere	locul 4 pentru 1 și 2 clustere locul 3 pentru 3 și 4 clustere
întârzieri capăt la capăt	locul 2 pentru 1, 2 și 3 clustere locul 3 pentru 4 clustere	locul 3 pentru 1, 2 și 3 clustere locul 2 pentru 4 clustere	locul 4	locul 1
încărcătura de rutare normalizată	locul 1 pentru 2, 3 și 4 clustere locul 2 pentru 1 cluster	locul 2 pentru 2, 3 și 4 clustere locul 1 pentru 1 cluster	locul 3	locul 4
Punctaj rețea având 1 cluster	20	14	18	18
Punctaj rețea având 2 clustere	19	15	18	18
Punctaj rețea având 3 clustere	17	14	21	18
Punctaj rețea având 4 clustere	17	14	21	18
Punctaj total protocoale per scenariu	73	57	78	72

Tabel 5.26 Schema de decizie a protocolului optim pentru scenariul cu variația numărului de noduri

Scenariul cu variația numărului de noduri	DSDV	AODV	DSR	AOMDV
durata de viață	locul 4 pentru 25 și 64 de noduri locul 3 pentru 100 și 196 de noduri	locul 1	locul 2 pentru 25 de noduri locul 3 pentru 64 de noduri	locul 3 pentru 25 de noduri locul 2 pentru 64, 100 și 196 de noduri
timp de inițializare	locul 4	locul 1	locul 3 pentru 25 și 64 de noduri	locul 2
procent recepție	locul 4 pentru 25 de noduri locul 3 pentru 64, 100 și 196 de noduri	locul 2 pentru 25 de noduri locul 4 pentru 64, 100 și 196 de noduri	locul 1 pentru 25 și 64 de noduri	locul 3 pentru 25 de noduri locul 2 pentru 64, 100 și 196 de noduri
număr de pachete recepționate	locul 2 pentru 25 de noduri locul 4 pentru 64, 100 și 196 de noduri	locul 3 pentru 25 și 64 de noduri locul 2 pentru 100 și 196 de noduri	locul 4 pentru 25 de noduri locul 1 pentru 64 de noduri	locul 1 pentru 25, 100 și 196 de noduri locul 2 pentru 64 de noduri
throughput	locul 2 pentru 25, 100 și 196 de noduri locul 3 pentru 64 de noduri	locul 4 pentru 25 și 64 de noduri locul 3 pentru 100 și 196 de noduri	locul 2 pentru 25 de noduri locul 1 pentru 64 de noduri	locul 1 pentru 25, 100 și 196 de noduri locul 2 pentru 64 de noduri
întârzieri capăt la capăt	locul 3	locul 1 pentru 25 și 196 de noduri locul 2 pentru 64 și 100 de noduri	locul 4	locul 2 pentru 25 și 196 de noduri locul 1 pentru 64 și 100 de noduri
încărcătura de rutare normalizată	locul 2 pentru 25, 64 și 100 de noduri locul 4 pentru 196 de noduri	locul 3 pentru 25 și 64 de noduri locul 1 pentru 100 și 196 de noduri	locul 1 pentru 25 și 64 de noduri	locul 4 pentru 25, 64 și 100 de noduri locul 2 pentru 196 de noduri
Punctaj rețea de 25 de noduri	21	15	17	16
Punctaj rețea de 64 de noduri	23	18	14	15
Punctaj rețea de 100 de noduri	21	14	28	13
Punctaj rețea de 196 de noduri	23	13	28	12
Punctaj total protocoale per scenariu	88	60	87	56

In cazul scenariului cu variația vitezei de mișcare protocolul AOMDV obține rezultatele cele mai bune indiferent de viteza de mișcare a nodurilor în timp ce protocolul DSR obține cele mai slabe rezultate. Protocolul AODV are rezultate apropiate de protocolul DSR.

In cazul scenariului cu variația frecvenței de mișcare cele mai slabe rezultate indiferent de frecvența de mișcare a nodurilor sunt obținute de protocolul AODV. Protocolul DSR chiar dacă are rezultate foarte bune pentru frecvente de 1/150 și 1/100 Hz, are rezultate foarte slabe pentru frecvente de 1/200 și 1/50Hz, iar per total are rezultate bune. Cel mai bun protocol, chiar dacă nu la toate frecventele de mișcare este protocolul AOMDV. Pentru frecvente mai reduse uneori căutarea de rute multiple se dovedește a fi inutilă.

In cazul scenariului cu variația numărului de clustere protocolul cu cele mai bune rezultate indiferent de numărul de clustere este protocolul AODV. Cele mai slabe rezultate sunt înregistrate de protocolul DSR. Protocolul DSDV împreună cu AOMDV obțin rezultate apropiate și destul de slabe demonstrând că nu excelează în condiții de trafic intens.

In cazul scenariului cu variația numărului de noduri cele mai slabe rezultate sunt obținute de protocolul DSDV ce demonstrează că nu face față atunci când există un număr mare de noduri în rețea. Rezultate apropiate și la fel de slabe sunt înregistrate și de protocolul DSR care nu reușește să găsească rute atunci când există un număr mai mare de hop-uri între sursă și destinație, lucru specificat și în teorie. Protocolul AOMDV reușește să obțină cele mai bune rezultate per total chiar dacă pentru rețele de dimensiuni reduse căutarea de rute multiple nu se justifica. Protocolul AODV are rezultate apropiate și suficient de bune fiind un protocol stabil în acest scenariu.

6. Concluzii finale

Rețelele de senzori wireless au o vastă gamă de aplicații datorită flexibilității acestora. Încă de pe acum, dar mai ales în anii următori evoluția acestor rețele va fi una importantă. Astfel de rețele sunt folosite pentru eficientizarea multor domenii de activitate datorită faptului că oferă posibilitatea identificării aspectelor ce trebuie optimizate. Pe lângă aceasta vor deschide noi domenii de activitate și cercetare.

Caracteristică de bază a acestor rețele este flexibilitatea. Aceasta se referă atât la domeniile vaste în care se pot utiliza astfel de rețele dar și în modul de lucru al acestui tip de rețele. Rețelele de senzori wireless sunt rețele ce se auto-organizează și se auto-configurează făcând față cu succes unei dinamicități crescute și unei modificări frecvente a topologiei. Chiar dacă ideea de rețele de senzori wireless a apărut de ceva timp, studiile și implementarea acestor rețele nu au reușit să ofere soluții la toate problemele existente. Interesul în creștere pentru astfel de rețele face ca dorința de performanță să fie cât mai mare. Rutarea este cel mai important aspect întrucât prin aceasta se poate obține eficiența, rapiditatea, consumul redus de energie, siguranța, etc., caracteristici importante pentru rețelele de senzori.

In ceea ce privește rutarea, dintre algoritmii clasici, protocolul AODV are rezultate foarte bune în scenariul cu variația numărului de clustere fiind protocolul optim pentru acesta. Rezultate bune obține și în scenariul de variație al numărului de noduri, fiind un protocol cu rezultate constante. În schimb în scenariile de mobilitate obține rezultate slabe. Varianta optimizată a acestui protocol, AOMDV prezintă rezultatele cele mai bune, chiar dacă nu în toate scenariile și nu la toate capitolele. Protocolul AOMDV face față foarte bine scenariilor de mobilitate și este un protocol scalabil. În scenariul în care este variat numărul de clustere acesta nu obține cele mai bune rezultate fiind pe locul doi după AODV.

In final, pot spune că protocolul AOMDV are rezultate generale mai bune decât celelalte protocoale. Aceste rezultate pot fi îmbunătățite prin diverse variante de optimizare, o parte din acestea fiind prezentate în această lucrare, în funcție de aplicația în care se dorește utilizarea rețelelor de senzori wireless.

Anexa 1. Formatul programelor folosite

Exemplu de fişier.tcl

```
#==================================

# Parametri iniţiali simulare

#==================================

set val(chan)    Channel/WirelessChannel ; definire canal
set val(prop)    Propagation/TwoRayGround  ; modelul radio de propagare
set val(netif)   Phy/WirelessPhy ; tipul de interfaţa al reţelei
set val(mac)     Mac/802_11 ;# tipul MAC
set val(ifq) Queue/DropTail/PriQueue; #tipul cozii a nodurilor
set val(ll)            LL ;#tipul nivelului de legătura
set val(ant)     Antenna/OmniAntenna ;#tipul antenei
set val(x)       400   ;# dimensiunea X a topografiei
set val(y)       400   ;# dimensiunea Y a topografiei
set val(ifqlen)  10    ;# numarul maxim de pachete în coada
set val(nn)      25    ;# numarul de noduri
set val(stop)    800.0   ;# timpul de simulare
set val(rp)      AODV     ;# protocol de rutare
set val(agent)   Agent/AODV
set val(energymodel)  EnergyModel ;#modelul de energie
set val(radiomodel)    RadioModel ;#modelul radio
set val(initialenergy)  500       ;# Energia iniţiala în Joules

Phy/WirelessPhy set RXThresh_ 3.5672e-9 ;# setare raza transmisie


#======================================
# Iniţializare variabile globale
#======================================

set ns_[new Simulator]

set tracefd     [open simple.tr w]
set namfd       [open wireless.nam w]
set f0 [open wireless-tcp.data w]
$ns_ use-newtrace
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namfd 20 20

# set up topography object
set topo[new Topography]

$topo load_flatgrid $val(x) $val(y)
```

```
    #======================================
    # Setarea parametrilor nodurilor mobile
    #======================================

     $ns_ node-config -adhocRouting $val(rp) \
                      -llType $val(ll) \
                      -macType $val(mac) \
                      -ifqType $val(ifq) \
                      -ifqLen $val(ifqlen) \
                      -antType $val(ant) \
                      -propType $val(prop) \
                      -phyType $val(netif) \
                      -channelType $val(chan) \
                      -topoInstance $topo \
                      -agentTrace ON \
                      -routerTrace ON \
                      -macTrace ON \
                      -energyModel $val(energymodel) \
                      -idlePower 0.5 \
                      -rxPower 1.0 \
                      -txPower 3.0 \
                      -sleepPower 0.001 \
                      -transitionPower 0.6 \
                      -transitionTime 0.005 \
                      -initialEnergy $val(initialenergy)

    #======================================
    # Definirea nodurilor mobile
    #======================================


    for {set i 0} {$i < $val(nn) } {incr i} {
          set node_($i) [$ns_ node]
                }

$node_(0) set X_ 1.0
$node_(0) set Y_ 1.0

$node_(1) set X_ 1.0
$node_(1) set Y_ 100.0

$node_(2) set X_ 1.0
$node_(2) set Y_ 200.0

$node_(3) set X_ 1.0
$node_(3) set Y_ 300.0

$node_(4) set X_ 1.0
$node_(4) set Y_ 400.0

$node_(5) set X_ 100.0
$node_(5) set Y_ 1.0

$node_(6) set X_ 100.0
$node_(6) set Y_ 100.0
```

```
$node_(7) set X_ 100.0
$node_(7) set Y_ 200.0

$node_(8) set X_ 100.0
$node_(8) set Y_ 300.0

$node_(9) set X_ 100.0
$node_(9) set Y_ 400.0

$node_(10) set X_ 200.0
$node_(10) set Y_ 1.0

$node_(11) set X_ 200.0
$node_(11) set Y_ 100.0

$node_(12) set X_ 200.0
$node_(12) set Y_ 200.0

$node_(13) set X_ 200.0
$node_(13) set Y_ 300.0

$node_(14) set X_ 200.0
$node_(14) set Y_ 400.0

$node_(15) set X_ 300.0
$node_(15) set Y_ 1.0

$node_(16) set X_ 300.0
$node_(16) set Y_ 100.0

$node_(17) set X_ 300.0
$node_(17) set Y_ 200.0

$node_(18) set X_ 300.0
$node_(18) set Y_ 300.0

$node_(19) set X_ 300.0
$node_(19) set Y_ 400.0

$node_(20) set X_ 400.0
$node_(20) set Y_ 1.0

$node_(21) set X_ 400.0
$node_(21) set Y_ 100.0

$node_(22) set X_ 400.0
$node_(22) set Y_ 200.0

$node_(23) set X_ 400.0
$node_(23) set Y_ 300.0

$node_(24) set X_ 400.0
$node_(24) set Y_ 400.0
```

```
    #=======================================
    # Generare mișcare
    #=======================================
$ns_ at 15.0 "$node_(4) setdest 1.0 1.0 15.0"
#=================================


#    Definirea agenților
#=================================


#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n12 $tcp0
set sink2 [new Agent/TCPSink]
$ns attach-agent $n10 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500



#=================================


#    Definirea aplicației
#=================================


#Setup  a  FTP  Application  over  TCP
connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 0.0 "$ftp0 start"
$ns at 240.0 "$ftp0 stop"
```

```
#=====================================
#     Setare final simulare
#=====================================


$ns_ at $val(stop) "stop"
$ns_ at $val(stop) "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd namfd f0
    $ns_ flush-trace
      close $namfd
    close $tracefd
exec nam wireless.nam &
    $ns_ halt
}
puts "Starting Simulation..."
$ns_ run
```

Exemplu de fişier.tr (trace)

```
s 10.000000000 _0_ AGT  --- 0 tcp 40 [0 0 0 0] [energy 500.000000 ei 0.000 es
0.000 et 0.000 er 0.000] ------- [0:0 99:0 32 0] [0 0] 0 0
r 10.000000000 _0_ RTR  --- 0 tcp 40 [0 0 0 0] [energy 500.000000 ei 0.000 es
0.000 et 0.000 er 0.000] ------- [0:0 99:0 32 0] [0 0] 0 0
s 10.004989470 _0_ RTR  --- 1 DSR 32 [0 0 0 0] [energy 500.000000 ei 0.000 es
0.000 et 0.000 er 0.000] ------- [0:255 99:255 32 0] 1 [1 1] [0 1 0 0->0] [0 0
0 0->0]
s 10.005124470 _0_ MAC  --- 1 DSR 90 [0 ffffffff 0 800] [energy 500.000000 ei
0.000 es 0.000 et 0.000 er 0.000] ------- [0:255 99:255 32 0] 1 [1 1] [0 1 0 0-
>0] [0 0 0 0->0]
r 10.005844800 _1_ MAC  --- 1 DSR 32 [0 ffffffff 0 800] [energy 494.996718 ei
5.003 es 0.000 et 0.000 er 0.001] ------- [0:255 99:255 32 0] 1 [1 1] [0 1 0 0-
>0] [0 0 0 0->0]
r 10.005844800 _10_ MAC  --- 1 DSR 32 [0 ffffffff 0 800] [energy 494.996718 ei
5.003 es 0.000 et 0.000 er 0.001] ------- [0:255 99:255 32 0] 1 [1 1] [0 1 0 0-
>0] [0 0 0 0->0]
r 10.005844937 _11_ MAC  --- 1 DSR 32 [0 ffffffff 0 800] [energy 494.996718 ei
5.003 es 0.000 et 0.000 er 0.001] ------- [0:255 99:255 32 0] 1 [1 1] [0 1 0 0-
>0] [0 0 0 0->0]
r 10.005869800 _1_ RTR  --- 1 DSR 32 [0 ffffffff 0 800] [energy 494.996718 ei
5.003 es 0.000 et 0.000 er 0.001] ------- [0:255 99:255 32 0] 1 [1 1] [0 1 0 0-
>0] [0 0 0 0->0]
r 10.005869800 _10_ RTR  --- 1 DSR 32 [0 ffffffff 0 800] [energy 494.996718 ei
5.003 es 0.000 et 0.000 er 0.001] ------- [0:255 99:255 32 0] 1 [1 1] [0 1 0 0-
>0] [0 0 0 0->0]
r 10.005869937 _11_ RTR  --- 1 DSR 32 [0 ffffffff 0 800] [energy 494.996718 ei
5.003 es 0.000 et 0.000 er 0.001] ------- [0:255 99:255 32 0] 1 [1 1] [0 1 0 0-
>0] [0 0 0 0->0]
s 10.058618418 _0_ RTR  --- 2 DSR 32 [0 0 0 0] [energy 494.995278 ei 5.003 es
0.000 et 0.002 er 0.000] ------- [0:255 99:255 32 0] 1 [1 2] [0 2 0 0->0] [0 0
0 0->0]
```

Exemplu de fişier.awk

```
BEGIN{

recvd = 0;  # numărul pachetelor de date recepţionate

rt_pkts = 0;# numărul pachetelor de rutare recepţionate

}

{

##### verificare dacă este pachet de date

if (( $1 == "r") && ( $7 == "cbr" || $7 =="tcp" ) && ( $4=="AGT" )) recvd++;


##### verificare dacă este pachet de rutare

if (($1 == "s" || $1 == "f") && $4 == "RTR" && ($7 =="AODV" || $7 =="message" || $7
=="DSR" || $7 =="AOMDV")) rt_pkts++;

}

END {  printf("%.3f\n", rt_pkts/recvd); }
```

Bibliografie:

[1]- Andrew S. Tanenbaum – Computer networks pag. 241-243;

[2]- Aruna Jayasuriya, Sylvie Perreau, Arek Dadej, Steven Gordon, Hidden vs. Exposed Terminal Problem în Ad hoc Networks, 2004;

[3]- K. Holger, A. Willig "Protocols and Architectures for Wireless Sensor Networks", JohnWiley & Sons 2005

[4]- Mani Potnuru, Phanindra Gandi, Wireless Sensor Network: Issues, Challenges and Survey of Solutions, 2003;

[5]- Al-Sakib Khan Pathan, Hyung-Woo Lee, Choong Seon Hong, Security în Wireless Sensor Networks: Issues and Challenges, 2006;

[6]- Chris Karlof, David Wagner, Secure Routing inWireless Sensor Networks: Attacks and Countermeasures, 2003;

[7]- Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar, SPINS: Security Protocols for Sensor Networks, 2001;

[8]- Yih-Chun Hu, David B. Johnson, Adrian Perrig, SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks, in: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, New York, June 2002;

[9]- M.G. Zapata, N. Asokan, Securing ad hoc routing protocols, in: Proceedings of ACM Workshop on Wireless Security (WiSe), Atlanta, September 2002;

[10]- S. Giordano, M. Hamdi, Mobility management: the virtual home region, Technical Report No. SSC/1999/037, EPFL, October 1999;

[11]- S. Basagni, I. Chlamtac, V. Syrotiuk, B. Woodward, A distance routing effect algorithm for mobility (DREAM), Dallas, TX, 1998;

[12]- Reinhardt Karnapke, Unidirectional Links în Wireless Sensor Networks, 2012;

[13]- Stephan Mank, Reinhardt Karnapke, Jorg Nolte, MAC Protocols for Wireless Sensor Networks: Tackling the Problem of Unidirectional Links, 2009;

[14]- Kemal Akkaya, Mohamed Younis, An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks, 2003;

[15]- Zheng-Yu Wu, Han-Tao Song, Ant-based Energy-aware Disjoint Multipath Routing Algorithm for MANETs Department of Computer Science and Technology, College of Information Science and Technology, Beijing Forestry University, 2010

[16]- W Yoon, N Vaidya, Routing exploiting multiple heterogeneous wireless interfaces: A TCP performance study. Comput Commun. 33(1), 23–34 (2010);

[17]- L Brakmo, S O‟Malley, L Peterson, TCP vegas: new techniques for congestion detection and avoidance. ACM SIGCOMM Comput Commun Rev. 24(4), 24–35 (1994);

[18]- Ewa Romanowicz, TCP with Explicit Link Failure Notification, 2008;

[19]- Kemal Akkaya and Mohamed Younis, A Survey on Routing Protocols for Wireless Sensor Networks, 2004;

[20]- Alexandros Koliousis and Joseph Sventek, Proactive vs reactive routing for wireless sensor networks, 2008;

[21]- Narasimha Rao Velagaleti, Attacks and Countermeasures on Routing Protocols în Wireless Networks, 2008;

[22]- Charles E. Perkins, Pravin Bhagwat Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, Computer Science Department University of Maryland College Park, 1994;

[23]- Guoyou He, Destination-Sequenced Distance Vector (DSDV) Protocol, Networking Laboratory Helsinki University of Technology, 1999;

[24]- http://www.nsnam.org/doxygen/classns3_1_1dsdv_1_1_dsdv_header.html;

[25]- C.E Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, 1999;

[26]- http://www.nsnam.org/doxygen/group__aodv.html;

[27]- David B. Johnson, David A. Maltz, Josh Broch, DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, Computer Science Department Carnegie Mellon University Pittsburgh;

[28]- http://www.nsnam.org/doxygen/group__dsr.html;

[29]- Mingliu Zhang, Routing Protocols for Mobile Wireless Network, 2004;

[30]- Ye Ming Lu, Multipath Routing Algorithm for Wireless Sensor Networks, 2005;

[31]- Asis Nasipuri, Robert Castaneda, Samir R., Performance of Multipath Routing for On-Demand Protocols în Mobile Ad Hoc Networks, 2001;

[32]- Mahesh K. Marina and Samir R. Das, Ad hoc on-demand multipath distance vector routing, 2006;

[33]- C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust

communication paradigm for sensor networks, in: Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom_00), Boston, MA, August 2000;

[34]- Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva, Directed Diffusion for Wireless Sensor Networking, în Proceedings IEEE/ACM, Vol. 11, 2003;

[35]- http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf;

[36]- RFC 3561;

[37]- RFC 4728;

[38] C. K. L. Lee, X. H. Lin, and Y. K. Kwok, "A Multipath Ad Hoc Routing Approach to Combat Wireless Link Insecurity", Proc. ICC 2003, vol. 1, pp.448–452, May 2003.

[39] Marc Greis. *Ns Tutorial*. http://www.isi.edu/nsnam/ns/tutorial/index.html

[40] S. McCanne and S. Floyd. *Network Simulator*. http://www.isi.edu/nsnam/ns/

[41] TCL Tutorial. http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html

[42] Tracegraph http://www.tracegraph.com/download.html